

MDSSEF - A Federated Architecture for Product Procurement

Jaspreet Singh Pahwa

Ph.D. 2009

UMI Number: U585233

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



UMI U585233

Published by ProQuest LLC 2013. Copyright in the Dissertation held by the Author.
Microform Edition © ProQuest LLC.

All rights reserved. This work is protected against
unauthorized copying under Title 17, United States Code.



ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

MDSSEF - A Federated Architecture for Product Procurement

Jaspreet Singh Pahwa

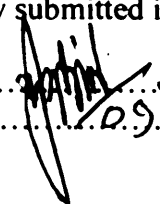
**School of Computer Science
Cardiff University**

**This thesis is submitted in partial fulfillment of the
requirement for the degree of Doctor of Philosophy**

July 2009

DECLARATION

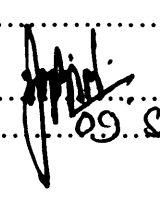
This work has not previously been accepted in substance for any degree and is not concurrently submitted in candidature for any degree.

Signed  (candidate)

Date 09 September 2009 .

STATEMENT 1

This thesis is being submitted in partial fulfillment of the requirements for the degree of *Ph.D* (insert MCh, MD, MPhil, PhD etc, as appropriate)

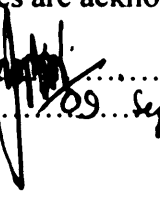
Signed  (candidate)

Date 09 September 2009 .

STATEMENT 2

This thesis is the result of my own independent work/investigation, except where otherwise stated.

Other sources are acknowledged by explicit references.

Signed  (candidate)

Date 09 September 2009 .

STATEMENT 3

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed  (candidate)

Date 09 September 2009 .

Summary

In the AEC (Architecture / Engineering / Construction) industry, large construction projects are tackled by consortia of companies and individuals, who work collaboratively for the duration of the project. The consortia include design teams, product suppliers, contractors and inspection teams who must collaborate and conform to predefined scheduling constraints and standards. These projects are unique, complex and involve many participants from a number of organisations.

Construction projects require consortia to procure supplies such as building materials and furniture from product suppliers. In large AEC projects, procurement of products, services and construction materials is an important and time consuming activity. Materials are sourced on a global basis from a large number of suppliers. The scale of the purchases made in large projects show that their procurement is a non-trivial exercise. Therefore, consortia members or the contractors require access to a large body of information about products or material information to aid procurement decision making.

Web based communication and network technologies play an increasingly important role in supporting collaboration in AEC projects. However collaborative working in the construction industry is still restricted by the current limitations of network and communication technologies and their system architectures which are usually client/server based. The construction industry has been examining how the advancements in distributed computing technologies such as the Grid computing can remove some of the existing limitations and enhance collaboration.

This research investigated how the procurement challenges such as accessing up-to-date product information available from a large number of products suppliers in an integrated manner using standard means could be addressed. A novel solution to the procurement challenges in the form of a distributed information sharing architecture is presented. The architecture uses the concepts of federated databases such as distribution of data and autonomy of databases and couples it with Grid computing to facilitate information exchange in a collaborative, coherent and integrated way to address the product procurement challenges.

Acknowledgements

I am deeply grateful to everyone who provided support and encouragement during the time I have been studying for my Ph.D. I am particularly grateful to:

Professor Alex Gray and Professor John Miles – this work would not have been possible without their unflinching support. They helped, supported and encouraged me in all possible ways and provided all the needed resources to help me achieve my aim. I am also thankful to the Head of Department Professor Nick Fiddian for his support.

My colleagues at the COVITE project Liviu Joita, Pete Burnap and Professor Omer Rana with whom I enjoyed working and for their invaluable support, feedback and motivation.

I would like to give special thanks to Wendy Ivins and Omnia Allam with whom I had numerous discussions about my research and they provided me with guidance and support whenever needed.

Finally, I would like to thank my parents and family, and particularly my wife Neeta for her unwavering support, encouragement and patience during my years of study.

*To my daughter Kirpa and nephew Veer
who were born during the time I was
studying for my Ph.D. They have
brought immense happiness in our lives.*

Acronyms

ACL	Agent Communication Language
AEC	Architecture / Engineering / Construction
ANSI	American National Standards Institute
APSL	ActivePlan Solutions Limited
BASIS	Biology of Ageing e-Science Integration and Simulation system
B2B	Business to Business
BBQ	Blended Browsing and Querying
BDW	Biodiversity World
CAD	Computer-aided Design
CAS	Common Access System
CDM	Common Data Model
CIS	Cooperative Information System
COIN	COntext INterchange
CORBA	Common Object Request Broker Architecture
COVITE	Collaborative Virtual Teams
CSCW	Computer Supported Cooperative Work
DBI	Database Implementer
DBS	Database System
DDL	Data Definition Language
DBMS	Database Management System
DDXMI	Distributed Database Xml Metadata Interface
DIKE	Database Intensional Knowledge Extractor
DIOM	Distributed Interoperable Object Model
DISCO	Distributed Information Search COmponent
DSS	Database Search Service
DTD	Document Type Definition
EDI	Electronic Data Interchange
EFIS	Engineering Federated Information Systems
EPC	Engineer-Procure-Construct
ESS	Extensible Services Switch
FDBMS	Federated Database Management System
FDBS	Federated Database System
FIS	Federated Information Systems
FM	Facilities Management
GAM	GeneGrid Application Manager
GARM	General AEC Reference Model
GSH	Grid Service Handle
GSI	Grid Security Infrastructure
GUI	Graphical User Interface
GUID	Globally Unique Identifier
HDDBMS	Heterogeneous Distributed Database Management System
HERMES	HEterogeneous Reasoning and MEdiator System
HOSQL	Heterogeneous Object Structured Query Language
HTML	Hyper Text Mark-up Language
HTTP	Hyper Text Transfer Protocol
IAI	International Alliance for Interoperability
ID	Identification

IDL	Interface Definition Language
IFCs	Industry Foundation Classes
IIS	Internet Information Server
IQL	Interface query Language
IRO-DB	Interoperable Relational and Object DataBases
IRMA	Information Reference Model for Architecture, Engineering, and Construction
ISO	International Standards Organisation
IT	Information Technology
IT/IS	Information Technology/ Information Systems
JDBC	Java Database Connectivity
KB	Knowledge Base
KQML	Knowledge Query and Manipulation Language
KRBL	Knowledge Representation Base Language
LDL	Logical Data Language
LIM	Loom Interface Module
MASS	Mirroring Application Support Scheme
MBIS	Mediator-based Information Systems
MDB	Multidatabase
MDBS	Multidatabase System
MDSS	Multiple Database Search Service
MDSSF	Multiple Database Search Service Federation
MGS	Master Grid Service
MIST	Model Integration Semantic Tool
MRDS	Multics Relational Data Store
MVDB	MultiViewDataBase
MVDL	Multiple View Definition Language
ODBC	Open Database Connectivity
ODL	Object Definition Language
OEM	Object Exchange Model
OGSA	Open Grid Services Architecture
OGSA-DAI	Open Grid Services Architecture-Data Access and Integration
OGSI	Open Grid Services Infrastructure
OML	Object Manipulation Language
OQL	Object Query Language
PCD	Product Class Database
PDF	Portable Document Format
PKI	Public Key Infrastructure
PSCD	Product Supplier Catalogue Database
PSE	Problem solving environment
RDBMS	Relational Database Management System
SD	Supplier Database
SDAI	Standard Data Access Interface
SEEK	Science Environment for Ecological Knowledge
SME	Small-to-medium Enterprise
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SPCD	Supplier-side Product Class Database
SQL	Structured Query Language
STEP	STandard for Exchange of Product data

SWQL	Semantic Web Query Language
T-SQL	Transact-SQL or Transact – Structured Query Language
TSIMMIS	The Stanford-IBM Manager of Multiple Information Sources
UK	United Kingdom
UML	Unified Modelling Language
VDD	Virtual Distributed Database
VO	Virtual Organisation
VR	Virtual Reality
webDDL	Web Data Definition Language
WSQL	WebSemantics Query Language
WSRF	Web Services Resource Framework
WWD-QL	World Wide Database Query Language
WWW	World Wide Web
XMF	XML-based Mediation Framework
XMAS	XML Matching and Structuring Language
XML	eXtensible Markup Language

Table of Contents

Declaration/Statements.....	i
Summary.....	ii
Acknowledgements.....	iii
Acronyms.....	v
1. Introduction.....	1
1.1 Background and Context for the Research.....	1
1.2 Procurement Challenges.....	3
1.3 The PSCD Application.....	4
1.4 Research Hypothesis and Objectives.....	7
1.4.1 Objective 1.....	7
1.4.2 Objective 2.....	8
1.4.3 Objective 3.....	10
1.4.4 Objective 4.....	10
1.5 Chapter Summary.....	10
1.6 Thesis Contents.....	11
1.7 Chapter Conclusions.....	13
2. The MDSSF Information Sharing Architecture.....	14
2.1 Introduction.....	14
2.2 The Functional Areas of the COVITE Research.....	15
2.2.1 Data Management and Grid-enabled Distributed Database Search	16
2.3 The MDSSF Information Sharing Architecture	17
2.3.1 The Product Class	19
2.3.2 The PCD System.....	20
2.3.3 The SPCD and the SD systems.....	20
2.3.4 The Multiple Database Search Service (MDSS) System.....	21
2.4 Novelty of the Proposed Information Sharing Architecture.....	22
2.5 Chapter Conclusions.....	23
3. Construction Procurement.....	24
3.1 Introduction.....	24
3.2 Background.....	26
3.2.1 The Construction Supply Chain.....	26
3.2.2 The Engineer-Procure-Construct (EPC) Projects.....	27
3.2.2.1 Engineering/Design Phase.....	28
3.2.2.2 The Construction Phase.....	28
3.2.3 The Need for Improvement in the Construction Industry.....	29
3.3 Procurement.....	29
3.3.1 Sub-Contracting Arrangements.....	31
3.3.2 A Brief Overview of Building Procurement Systems.....	33
3.4 The Role of Information in the Construction Supply Chain.....	34

3.4.1 Information is a Strategic Resource.....	35
3.4.2 Flow of Goods and Services.....	35
3.4.3 Linking Actors in Construction Supply Chain.....	36
3.4.4 Coordination and Collaboration.....	37
3.4.5 Sellers and Purchasers.....	37
3.4.6 Role of Information in Other Areas of Construction.....	38
3.5 Role of Technology in Construction Supply Chain.....	38
3.5.1 Use of IT is Widespread in the Construction Industry.....	39
3.5.2 IT enables Cooperation and Information Sharing and Communication.....	40
3.5.3 Technology Accelerates the Rate of Information Flow.....	41
3.5.4 Competitive and Strategic Advantage.....	41
3.5.5 Inter-Organisational and Organisational Efficiency.....	42
3.5.6 Small Firms.....	43
3.5.7 Web Technologies.....	44
3.6 How MDSSF Can Benefit Construction Procurement.....	45
3.6.1 Organisations Depend on External Companies for Procurement.....	46
3.6.2 Communication is Important for Procurement.....	47
3.6.3 Suppliers in Different Supply Chains.....	47
3.6.4 Full Knowledge of Equipment Specification.....	47
3.6.5 Each Construction Project is Unique.....	48
3.6.6 Data Sharing in e-Business Models.....	49
3.6.7 Partnerships and Collaborative Working.....	50
3.6.8 Relationship Management.....	51
3.6.9 Small Firms.....	52
3.6.10 Supply Chain Operations and Logistics.....	53
3.6.11 Changing Environments and Globalisation.....	54
3.6.12 Decision Making in Strategic, Tactical and Operational Levels.....	55
3.6.13 The World Wide Web and Associated Technologies.....	56
3.6.14 Marketing of Supplier Products.....	56
3.7 Product Procurement.....	57
3.7.1 The Traditional Paper Based System.....	57
3.7.2 Supply Chain and Critical Chain Project Management.....	58
3.7.3 An Approach of Smarter Selection for Procurement.....	59
3.7.4 Procurement using E-Commerce Systems.....	63
3.7.4.1 Limitations of E-Commerce.....	64
3.7.5 The E-Union Framework.....	65
3.7.6 Keyword Searching for Products.....	66
3.8 Chapter Conclusions.....	66
4. Information Sharing in Distributed Environments.....	68
4.1 Introduction.....	68
4.2 Sharing of Information in Autonomous Environments.....	69
4.3 Information Mediators.....	70

4.3.1 Information Integration Using Mediators.....	71
4.3.2 Resource Wrappers.....	72
4.3.3 The Data Model and Query Language.....	73
4.3.4 Ontologies.....	74
4.3.5 Knowledge Based Information Sharing Systems.....	75
4.3.6 Section Summary.....	76
4.4 Database Interoperability and Schema Integration.....	79
4.4.1 Mediator vs. Schema Integration Based Systems.....	83
4.5 Federated Database Systems and Federated Information Systems.....	83
4.5.1 Federated Information Systems.....	87
4.6 Grid-based Systems.....	90
4.6.1. USECA Properties.....	93
4.7 Other Information Sharing Systems.....	94
4.8 Chapter Conclusions.....	95
5. The Product Classes.....	97
5.1 Introduction.....	97
5.2 Product Classes.....	97
5.3 Composition of Product Class.....	99
5.3.1 Unit Specification.....	100
5.3.2 Specification Group.....	100
5.3.3 List Specification.....	101
5.3.4 Table Specification.....	101
5.3.5 Sub-Product Class Specification.....	102
5.4 Versioning of Product Classes.....	103
5.5 AEC Industry Information Modelling/Management System/Standards.....	103
5.6 Chapter Conclusions.....	105
6. The MDSSF System Architecture.....	106
6.1 Introduction.....	106
6.2 Product Data Definition and Management in the PSCD Application.....	107
6.2.1 The Product Class Database (PCD) System.....	108
6.2.1.1 Modular Approach of PCD.....	110
6.2.1.2 Product Class and Product Class Version Entities.....	111
6.2.1.3 Versioning Support in PCD.....	112
6.2.1.4 Table Specification and Associated Entities.....	113
6.2.1.5 Default Values and Measurement Units.....	114
6.2.1.6 PCD Category Management.....	115
6.2.1.7 The Other Features of the PCD System.....	116
6.2.2 Supplier Product Data Management.....	116
6.2.2.1 SPCD and Subscription of Product Classes.....	117
6.2.2.2 The SD System.....	119

6.2.3 Section Summary.....	122
6.3 MDSS – a Virtual Distributed (VDD) of MDSSF.....	123
6.3.1 MDSS System Architecture.....	126
6.3.2 Distributed Database Search Using MDSS.....	128
6.3.2.1 Search Criteria.....	129
6.3.2.2 Search Space.....	129
6.3.2.3 Available Grid Resources for a Search.....	130
6.3.2.4 Data Aggregation.....	132
6.3.3 SD Web Interface.....	133
6.3.4 MGS and DSS Architectures.....	133
6.3.4.1 MGS.....	133
6.3.4.2 DSS.....	136
6.3.5 Section Summary.....	137
6.4 Software Development Process.....	138
6.5 Chapter Conclusions.....	139
7. System Testing, Verification and Validation of the MDSSF Architecture.....	140
7.1 Introduction.....	140
7.2 System Testing of Prototype.....	140
7.2.1 Test Objective 1.....	141
7.2.1.1 Identification of machines.....	142
7.2.1.2 Installation of MDSSF System Components	143
7.2.1.2.1 Set-up Grid enabled MDSS.....	144
7.2.1.2.2 Set up MDSSF Databases and Supplier Web Service Interface.....	146
7.2.2 Test Objective 2.....	147
7.2.2.1 Creating a Product Class.....	147
7.2.2.2 Assigning a Product Class.....	150
7.2.2.3 Creating a New Specification.....	150
7.2.2.4 Versioning Support in MDSSF Databases.....	151
7.2.2.5 Product Class Subscription in SPCD.....	153
7.2.2.6 Creation of Product Information in SD.....	153
7.2.3 Test Objective 3.....	154
7.2.3.1 Testing MGS.....	155
7.2.3.2 Testing of DSS.....	160
7.2.3.3 Testing the MDSS as a Single System.....	161
7.2.4 Test Objective 4.....	164
7.2.5 Section Summary.....	166
7.3 MDSSF Demonstration.....	167
7.4 Verification and Validation of the MDSSF Architecture.....	168
7.4.1 Benefits to the Construction Industry.....	168
7.4.2 MDSSF Information Sharing Architecture.....	170
7.4.2.1 Grid Support.....	170
7.4.2.2 Federated Architecture Supporting Competing Product Suppliers...171	171

7.4.2.3 MDSSF's Cooperation Model.....	171
7.4.2.4 Schema Integration.....	171
7.4.2.5 Subscription-based Approach.....	172
7.4.2.6 A Standard Method of Information Storage and Exchange.....	172
7.4.3 The Software Components of the MDSSF.....	173
7.4.3.1 Software Components at Supplier Side.....	174
7.4.3.2 MDSSF Supports a Single RDBMS and Schema.....	175
7.4.3.3 Linking of Databases at the Supplier Side.....	176
7.4.3.4 Consistency Checks and Data Constraints.....	176
7.4.3.5 Need for the Support of Additional Database Operations.....	177
7.4.3.6 The Grid-based Search.....	178
7.4.3.7 The Grid Middleware.....	179
7.4.4 System Testing and Evaluation of the MDSSF.....	179
7.5 Chapter Conclusions.....	180
8. Conclusions.....	181
8.1 Introduction.....	181
8.2 Achievement of Research Objectives.....	181
8.3 Research Contributions.....	185
8.3.1 Research Publications.....	185
8.3.2 Novelty of the Proposed Approach.....	185
8.4 Applicability of MDSSF in Other Domains.....	187
8.5 Future Work.....	188
Appendix 1: Performance Criteria of the Hong Kong based Study.....	191
Appendix 2: Representation of Product Information in MDSSF.....	192
Appendix 3: Information System Comparison Tables.....	218
Appendix 4: Setting-up MDSSF in a Distributed Environment.....	240
Appendix 5: MDSSF System Code.....	248
References.....	441

Table of Figures

Figure	Description	Page
2.1	The conceptual view of the distributed architecture of the new Grid-based PSCD application.	15
2.2	The conceptual view of the MDSSF.	17
2.3	Relationship between the SPCD and the SD systems at the product supplier end.	21
3.1	Key phases of the construction supply chain.	27
3.2	Procurement.	30
3.3	Proposed model of decision support system for optimising procurement protocols and complementary operational sub-systems as identified by Kumaraswamy et al. [Kum00].	62
4.1	The global schema integration process identified by Parent and Spaccapietra [Par98].	80
5.1	The Product class and its various specification types.	100
5.2	An example showing a Specification Group.	101
5.3	An example showing product attributes arranged in rows and columns.	102
6.1	Database diagram of the PCD system.	110
6.2	The reduced version of the PCD System identifying the three level hierarchy implemented in the system.	113
6.3	Relationship between the SPCD and the SD systems at the product supplier end.	120
6.4	Database diagram of the SD System.	121
6.5	The conceptual view of the MDSSF.	124
6.6	The MDSSF system architecture.	128
6.7	An example search criteria identifying the ID of the product class.	129
6.8	A snapshot of the XML document identifying the product supplier databases to search.	130
6.9	XML document identifying a subset of available Grid services which can be used to perform distributed database search.	131
6.10	A snapshot of the product data returned from an SD System.	132
6.11	UML class diagram of the MGS system component of the MDSS.	134
6.12	UML class diagram of the DSS system component of the MDSS.	136
7.1	The MDSSF system architecture.	142
7.2	The MDSSF system components and the machines in which they were installed.	144
7.3	The Grid Service Handle (GSH) of a DSS Grid service deployed in machine bouscat.cs.cf.ac.uk.	145
7.4	An example stored procedure execution code showing how to execute <i>proc CreateNewProductClass</i> stored procedure in PCD.	149
7.5	The stored procedure execution code showing how a new specification can be created and assigned to a product class.	151

Figure	Description	Page
7.6	A snapshot of Electric Bed version 1.0 product class showing two versions of <i>Type List Specification</i> .	152
7.7	An example search criteria identifying the ID of the product class.	156
7.8	A snapshot of <i>supplierString</i> XML document identifying the product supplier databases (SD systems) to search.	156
7.9	A snapshot of XML document identifying a subset of available Grid services via their GSH which can be used to perform distributed database search.	157
7.10	This figure shows how the <i>executeJob</i> method of the <i>MasterGridImpl</i> class in the MGS distributes database search jobs to DSS nodes.	158
7.11	A snapshot of code from <i>executeJob</i> method of the <i>MasterGridImpl</i> class, which shows how the method allocates database search jobs by creating <i>jobExecution</i> threads in a <i>for</i> loop.	159
7.12	A snapshot of product data returned from SD systems.	160
7.13	A snapshot of <i>supplierString</i> XML document identifying the product supplier databases (SD systems) to search.	162
7.14	An exception generated by <i>DatabaseSearchImpl</i> class of DSS system when it fails to retrieve product data from an SD system.	166

MDSSF System Code (Appendix 5)

Table of Contents

1. Product Class Database System (PCD) Code.....	248
2. Supplier's Product Class Database (SPCD) System Code.....	326
3. Supplier Database (SD) System Code.....	369
4. Master Grid Service (MGS) System Code.....	416
5. Database Search Service (DSS) System Code.....	431

1. Introduction

1.1 Background and Context for the Research

In the AEC (Architecture / Engineering / Construction) industry, large projects are tackled by consortia of companies and individuals, who work collaboratively for the duration of the project. These projects are usually unique, very complex and involve many participants from a number of organisations acting collaboratively. In order to tackle the complexity, consortia members provide a range of skills to the project from its inception to completion. The consortia include design teams, product suppliers, contractors and inspection teams who must collaborate and conform to predefined scheduling constraints and standards. These participants also work concurrently, thus requiring real time collaboration between the geographically remote participants. A typical consortium member is often providing similar services to multiple projects simultaneously involving different partners.

Construction projects range in size from design and construction of a single building, to the creation of a large national infrastructure such as airports, dams, and highways. The planning, implementation and running of these AEC industry projects thus requires the formation of virtual organisations (VOs)-computing infrastructures, which enable collaboration between its geographically dispersed members by sharing project information and resources. The VOs formed to enable the members of a consortium to work together for the duration of a project are electronically networked organisations, where technology plays an important role in coordinating the various activities [Bur99]. An important feature of the consortia is that they are dynamic in nature and are formed for the lifetime of the project only. Members can participate in several consortia at the same time and can join or leave a consortium as the project evolves.

Web based communication and network technology and distributed computing is beginning to play an increasingly important role in supporting collaboration in AEC projects, particularly by enabling the project management (or team) to identify the current state of a project, its activities, and the constraints on these activities and their schedules. However collaborative working in the construction industry is still restricted by the current limitations of network and communication technologies and their system

architectures which are usually client/server based. The construction industry has been examining how the advancements in network and distributed computing technologies can remove some of the existing limitations and improve the management of these projects [Mil02]. This will involve researching and creating new software tools.

In the construction industry a consortia procures supplies, such as building materials and furniture from suppliers, who specialise in manufacturing or retailing these products. In large AEC projects, procurement of products, services and construction materials is an important and time consuming activity. Materials are sourced on a global basis from a large number of product suppliers. The procurement process involves obtaining desired products from a wide range of products available from a large number of product suppliers. In large projects a large quantity of various kinds of construction material is required. For example a typical UK hospital has 8000 rooms. Each room needs items such as a light, a door, floor and ceiling, floor and ceiling coverings, furniture, power socket, some form of ventilation such as windows (possibly optional), walls and wall covering. Multiplying these requirements by 8000 rooms shows that a huge amount of product purchasing is necessary to build a hospital [Bur05]. The scale of the purchases required for building large artefacts shows that their procurement process is a non-trivial exercise. Hence the consortia members or the contractors require access to a large body of information about products and material information to aid procurement decision making.

In order to address the procurement challenges and improve the efficiency of the construction processes, the School of Computer Science and the School of Engineering, Cardiff University along with an industrial partner ActivePlan Solutions Limited (APSL) [Aps09] initiated a research project called COVITE (COllaborative Virtual TEams) [Mil02]. An important aim of the COVITE project was to investigate how recent advances in network and distributed computing technology and in particular the Grid-based distributed computing technology could be used to address procurement challenges. APSL made contributions to the project by providing their expertise in relation to the procurement processes in the construction industry. APSL also provided the COVITE research team with a software system called the Product Supplier Catalogue Database (PSCD) application (see Section 1.3) which was developed to

support product procurement. The author got an opportunity to be a member of the project team and to address the challenges and requirements identified in the project proposal. Although the COVITE project proposal [Mil02] identified a number challenges in areas such as security, user and application management, there are three procurement challenges, which are important to this research.

1.2 Procurement Challenges

- Procurement of supplies for construction projects has now become a global phenomenon where materials are sourced from suppliers operating in different parts of the world. Product suppliers usually deal with a wide range of products and for a given product or a range of products and there are a large number of competing suppliers. These suppliers operate in both regional and international markets, but there is no single integrated means of accessing the product information available from these different sources. Because of this lack of an infrastructure, consortia members have a limited possibility of reaching the large number of potential suppliers, let alone searching their databases for required products. Hence there is a requirement for an integrated information sharing system, because *“None of the existing construction industry information services have an adequately comprehensive database, largely due to the fact that they are information intermediaries and the number of individual products available in the construction means that the scale of this task is huge.”* [Mil02].
- The consortia members require up-to-date information about products which can be acquired from the external product suppliers, so that information, such as product specifications, availability, delivery time and cost can be taken into account in procurement planning. Providing up-to-date product information to the consortia members during the early stages (i.e. the conceptual and design stages of the construction project life cycle) of the construction processes is crucial, because 80% of the construction costs are fixed during this stage [Mil02]. However providing up-to-date product information at later stages is also crucial, as most of the actual procurement takes place during the construction phase. The need for up-to-date product information is also important from another perspective, namely to aid contractors in their decision

making processes through out the project lifecycle. For example a contractor in the middle of a construction activity may discover that a given product has a revised delivery date, which will cause significant delays to the project. Hence, it is necessary to quickly source an equivalent product from an alternate supplier within the given cost and with a suitable delivery date.

- Different product suppliers use different ways of managing product information. Although a large amount of product information is available from several suppliers, their responses have to be coordinated so that the information is presented in a coherent fashion to the consortium members to enable easy comparison of the competing products. Some suppliers have complex product information IT systems in place, whereas others store information as PDF files. Thus, there is heterogeneity in the way suppliers manage information about their products. Unless a standard mechanism is adopted by all the product suppliers to store information about their products, it is not possible to provide this information to contractors or consortia members in a standard way. Providing information in a standard way is important for integrated access because it aids comparison, when information about the same or similar products is supplied by different product suppliers. This then helps contractors to identify the suppliers, who are in the best position to meet the needs of a consortium given the constraints of availability, cost and delivery time.

1.3 The PSCD Application

The client/server based PSCD (Product Supplier Catalogue Database) application is an APSL software product, which supports collaborative working of consortia members, such as product suppliers and contractors. It uses a web-based technology and is concerned with making available to the members of a consortium information about products, which have to be acquired from external suppliers so that availability, delivery and cost can be taken into account in the procurement planning. This application provides a product database, which stores information about the products available from different suppliers. It allows products to be grouped under different categories according to their nature, and provides a mechanism to define simple product specifications. The application, via its front-end web based interface, makes product information (stored in

the central back-end database) available to buyers and contractors. The functionality of the PSCD application (provided to the project team by APSL at the start of the COVITE project [Mil02]) is limited. This restricted its ability to address the procurement challenges identified in Section 1.2. The limitations of the PSCD application important to this research are summarised here.

- The PSCD application provides a centralised database for storing and managing product information. A central database cannot overcome the challenges of Section 1.2 because the application domain is inherently distributed and consists of independent, autonomous and far flung industry actors who are continually updating this information. There are a large number of product suppliers operating in different parts of the world and dealing in a huge variety of different types of products. Secondly, the local autonomy of product suppliers must also be maintained, as it allows suppliers to manage their product information without any external influence. Product suppliers usually do not share their product related sensitive information, such as costs, specifications and delivery time with their competitors. They only share this information with potential buyers or contractors. Therefore they will not be willing to provide this information in a database which is not controlled by them because there is a risk of such information being accessed by a competitor. Additionally, product suppliers are also competitors and are competing with each other when they provide product information to contractors. There is a possibility that information about the same or similar products can be supplied by more than one product supplier. Therefore it should not be possible for a supplier to view the sensitive information of a competitor, as it may give an advantage to this supplier over another in bidding for orders. The centralised database of the PSCD application does not provide adequate mechanisms to restrict sharing of product information between product suppliers.
- The database of the PSCD application does not provide mechanisms to manage complex product information. It only provides mechanisms to define products having simple specifications (such as *height*, *width* and *weight* of a product) and allow them to be grouped according to a given criteria. However, products used

in the construction industry are usually complex, having a large number of different types of specifications. The high degree of complexity is a feature of construction industry products, where bespoke products are built from standard items, which can be assembled in several different ways. The degree of product complexity is far higher than the degree of complexity found in the automotive industry, and the problems are further aggravated because there is no mass production in the construction industry [Bur05]. Each solution designed is unique and is made in accordance with a customer's requirement or its anticipated use, although it is usually constructed from simpler parts.

- Managing a large amount of product information from different product suppliers is a difficult problem. This problem is further aggravated when there is heterogeneity, due to different product suppliers using different ways of managing product information. This makes it difficult to collaborate at the information level. Heterogeneity is an obstacle to information sharing, which also makes it hard to present the information in an integrated way using a standard means for comparison purposes or for matching a project's constraints with the supplier conditions. Additionally, new products or a new range of existing products are introduced by suppliers on a regular basis, as they enhance features and functionality. Often, the supplier also provides different versions of the same or a similar product providing different or additional features. The centralised database of the supplied PSCD application does not provide mechanisms to describe products as they evolve to give enhanced features and functionality. Hence in the PSCD application, it is not possible to distinguish different versions of a product according to the features they support and represent them using standard methods. This is because the schema of the database is built to a fixed structure, which does not support product evolution.

These challenges and the need to address the limitations of the PSCD application, led to the research hypothesis and research objectives presented in the next section.

1.4 Research Hypothesis and Objectives

It is possible to create an information sharing architecture which addresses the procurement challenges of construction consortia in a Grid-based distributed computing environment.

In order to address the procurement challenges and limitations of the PSCD application several research objectives can be outlined. They are:

1.4.1 Objective 1:

To identify how the new information sharing architecture will benefit construction industry actors.

The aim of this research objective is to identify a solution to the procurement challenges identified in Section 1.2 via an information sharing architecture which uses technological options to enable information sharing and enhance collaborative working. The requirements of the new information sharing architecture were set by industrial collaborators APSL and project supervisor Professor John Miles who have considerable experience and expertise in the area of construction procurement. These requirements were to address procurement challenges and limitations of the PSCD application described above via a new information sharing architecture. However, new models of information sharing cannot be brought into existence without understanding the present role of information and technology in the construction industry. From this perspective and for the purpose of adaptability, looking at the existing role of information, technology and practices that drive construction procurement is vital. It is important to look into these methods and practices to identify how a new information sharing architecture will benefit construction industry actors in their desire to make better procurement decisions.

1.4.2 Objective 2

To develop a distributed information sharing architecture meeting the procurement challenges.

As identified in Section 1.3, client/server based applications such as the PSCD application with its present limitations cannot address the procurement challenges, because the domain is inherently distributed consisting of a large number of actors operating in different parts of the world. Therefore, a distributed system architecture is needed to address these preset limitations. Secondly, a distributed architecture is also needed from another perspective, to enable product suppliers to manage their product information autonomously and protect this information from being viewed by a competitor. A suitable distributed architecture should ensure that a product supplier is able to share information only with those contractors who are responsible for procurement planning, such as buyers and contractors. Additionally, from the contractors' perspective the distributed architecture should present information on similar products but available from several suppliers in an integrated way. This will allow suppliers to compare product information available from several sources and identify the most appropriate source of procurement by considering factors such as project needs, product specifications, cost and delivery dates.

In a distributed information sharing architecture, information sharing in a coherent manner cannot take place unless standard methods of information sharing between different industry actors, such as product supplier and contractors are adopted. This means that common mechanisms are needed, which can be adopted by all suppliers, so that their products are described in a standard way. The rationale behind this approach is to promote standardisation, right from the beginning at the data exchange level, so that the inconsistencies, which heterogeneity brings, can be avoided in the first place and thus lead to improved collaboration in the consortium. The importance of collaboration between different consortia members, whilst managing construction activities cannot be overlooked. The consortia include design teams, product suppliers, contractors and inspection teams, who must collaborate and conform to predefined and pre-scheduled constraints and standards for efficient management and timely delivery of construction projects. Collaboration at the systems level leads to efficient collaboration of actors at

the user level, for example it can aid contractors to identify procurement sources and collaborate with only those suppliers, who are able to meet the project's constraints. Secondly, standardisation is important from the perspective of small-to-medium enterprises (SMEs), who form a large proportion of the construction workforce, to enable them to contribute their expertise and provide specialised input to construction projects whilst competing with their larger counterparts.

One of the principal aims of the COVITE project was to investigate how the advances in distributed computing and particularly Grid computing can be used in the domain of the construction industry to address some of its procurement challenges and aid the procurement processes [Mil02]. An important cause of limitations of PSCD application is its client/server based architecture, which is not able to take advantage of newer technology, such as the Grid's advanced distributed computing infrastructure, which provides greater support for collaborative working. The planning, implementation and running of construction projects and managing procurement processes is a complex task, in which distributed computing and Grid technology has been identified as potentially an important infrastructure in the future. The Grid is perceived as providing additional functions which will enhance the existing functionalities of the internet. It offers features such as enhanced security infrastructure including single sign-on capability, security between consortia, distribution of computationally intensive jobs across multiple distributed processors and resource information sharing. The COVITE project aimed to provide a Grid-based solution to the construction procurement challenges by using the Grid middleware Globus Toolkit available from the Globus project [Glo09] – the de facto standard for open source Grid computing infrastructure [Glo09a]. Therefore an important aim of the COVITE project was to investigate “*how best to re-implement*” the PSCD application so that it can utilise the advanced features of the Grid infrastructure and be deployed in a Grid environment by using Grid middleware [Mil02].

1.4.3 Objective 3

To show empirically the viability of the new distributed information sharing architecture by designing and implementing its various software components which address the procurement challenges at the data management and sharing level.

Research and development efforts are required to bring together autonomous contractors and suppliers by developing a distributed information sharing system which is based on the new architecture. The information sharing system should provide software components to manage product information at the suppliers' side and allow access to it using a distributed database search mechanism which searches all the relevant product supplier databases to retrieve product information. This product information (retrieved from several supplier databases) should then be presented to contractors in an integrated manner. The aim of this objective is to demonstrate that the present limitations of the PSCD application can be addressed and the procurement challenges met by means of a prototype system.

1.4.4 Objective 4

To verify and validate the new information sharing system and its architecture against the research hypothesis, research objectives and determine its strengths and weaknesses.

The software components of the new information sharing system will be tested to ensure that it enables product data management meeting the requirements of the contractors and suppliers. The new information sharing architecture will also be verified and validated for its ability to address the procurement challenges and achievement of research objectives.

1.5 Chapter Summary

The construction industry will benefit from an information sharing architecture, which allows members of consortia such as contractors to view up-to-date product information in a coherent presentation, where this information is provided by suppliers using a common structure. The research will investigate, whether it is possible to address the

procurement challenges by creating a new distributed information sharing architecture, which allows contractors to access information from a large number of suppliers, and to identify what products and services best suit the project requirements taking into consideration information, such as product specifications, material costs and delivery time. This information sharing architecture should bring together construction industry actors, such as contractors and product suppliers and facilitate information exchange in a collaborative, coherent and integrated way to address the product procurement challenges. The architecture should allow sharing of product information by product suppliers using standard means with potential buyers and contractors. The architecture should also allow product suppliers to manage their product information independently in their own databases without any external influence and to protect their local autonomy. It should allow contractors to search the databases of a large number of product suppliers and present the information (which is retrieved from several supplier databases) in an integrated way to allow contractors to judge competing products based on the project requirements. This will create a new model of information sharing using this architecture and a prototype system will be created to evaluate the approach. The new distributed information sharing architecture developed in this research is called the MDSSF (Multiple Database Search Service Federation) and is introduced in Chapter 2.

1.6 Thesis Contents

The new information sharing architecture created in this research uses the concepts of federated databases linked with distributed computing features provided by Grid technology. Chapter 2 introduces this new information sharing architecture and its components which address procurement challenges. It also shows how this new information sharing architecture can be used for client/server based applications such as the PSCD application.

Chapter 3 provides background information on construction procurement. The chapter provides an overview of the construction supply chain, introduces construction procurement and its importance and the role of information and technology in modern information-driven construction environments. The chapter also identifies how the information sharing architecture proposed as part of this research can benefit

organisations in different organisational scenarios when making procurement related decisions. It describes the present approaches to construction procurement.

In order to highlight the novel features of the new distributed information sharing architecture, Chapter 4 provides a review of different information sharing systems/architectures and schema integration methodologies. It reviews the key features of information sharing and integration approaches to identify their scope and functionality against the domain specific requirements of the new architecture.

Chapter 5 describes the concept of product classes in greater detail which is introduced in Chapter 2. Product classes provide a mechanism for defining and managing products in a standard way. The chapter identifies the composition of product classes and presents a description of version support needed to support product class evolution. The chapter also provides a brief summary and critique of some of the important and competing product modelling/management systems and reference architectures currently used in the AEC industry for the management of product data.

Chapter 6 describes the architecture of MDSSF in detail. This explains how the data management and Grid enabled distributed database search issues of the PSCD application were used to address the challenges of product procurement through the MDSSF architectural components. In this chapter the architecture of the MDSSF is described through its architectural components.

Chapter 7 is dedicated to the evaluation of the project and the testing of the MDSSF architectural components. It identifies the distinctive features of the MDSSF architecture, which provide a novel means of information sharing between construction industry actors such as product suppliers and contractors. The chapter critically evaluates the MDSSF architecture for its ability to address the procurement challenges for which it is designed.

Chapter 8 presents conclusions. The chapter provides a review of the research and highlights the research objectives achieved against the research hypothesis. The chapter also provides a summary of the contribution to research and learning and confirms the

research hypothesis. The future work section of the chapter identifies potential areas for further research.

1.7 Chapter Conclusions

The main achievement of this research has been the development of a novel information sharing architecture which addresses the product procurement challenges through the criteria established in the research and thereby increases collaboration between construction industry actors, such as product suppliers and contractors. The new distributed information sharing architecture was used to build a new prototype PSCD application which enabled it to use federated database concepts and distributed computing features provided by Grid technology. These features benefited the application by giving autonomy to product suppliers and providing scalability support. The second achievement of this research is the design and development of the MDSSF architecture's software components which address procurement challenges at the data management level in order to empirically test the viability of such an architecture. The third achievement has been the evaluation of the MDSSF information sharing architecture in terms of the research hypothesis and research objectives.

2. The MDSSF Information Sharing Architecture

2.1 Introduction

Information access and its management is an important area of research when creating new models of information retrieval and sharing to meet information needs of business organisations. Network technologies such as the Grid and Web Services together with federated database architectures provide a new means of collaboration and information exchange between the actors within a given industry. One of the features of the Grid is that it provides middleware to enable distributed computing in a particular domain to achieve high-end computational capabilities and high-throughput computing [Fos99]. Web Services is a paradigm for enabling computation in distributed and heterogeneous environments [Fos02]. Federated database architectures provide mechanisms to protect local autonomy and at the same time allow data to be shared with external users [She90]. Research in the area of Engineering Federated Information Systems (EFIS) has recognised Grid computing as an emerging area for building new models of data exchange [Wys03].

An important research objective of the COVITE project was to identify how Grid technology can be used to improve the collaboration between different industry actors by addressing the procurement challenges identified in section 1.2. The Grid enablement process investigated the applicability and advances in the distributed computing area in recent years to identify how such advances can be used in the sphere of the construction industry, and particularly in procurement planning. The test bed for this investigation and research was the prototype PSCD application (and the expertise of the industrial collaborator APSL) which provided the COVITE team with the necessary knowledge of construction procurement processes. The client/server based PSCD application and particularly its product database were found to be inadequate for the planned Grid enablement and therefore the application and its database were redesigned to take advantage of advanced distributed computing features provided by Grid middleware and to enable a standard mechanism of product data management and exchange. The client/server based PSCD application was transformed into a distributed application to effectively utilise the Grid middleware to provide scalability support

2. The MDSSF Information Sharing Architecture

when accessing a large number of supplier databases for required products. Figure 2.1 shows the conceptual view of the new Grid-enabled version of the PSCD application.

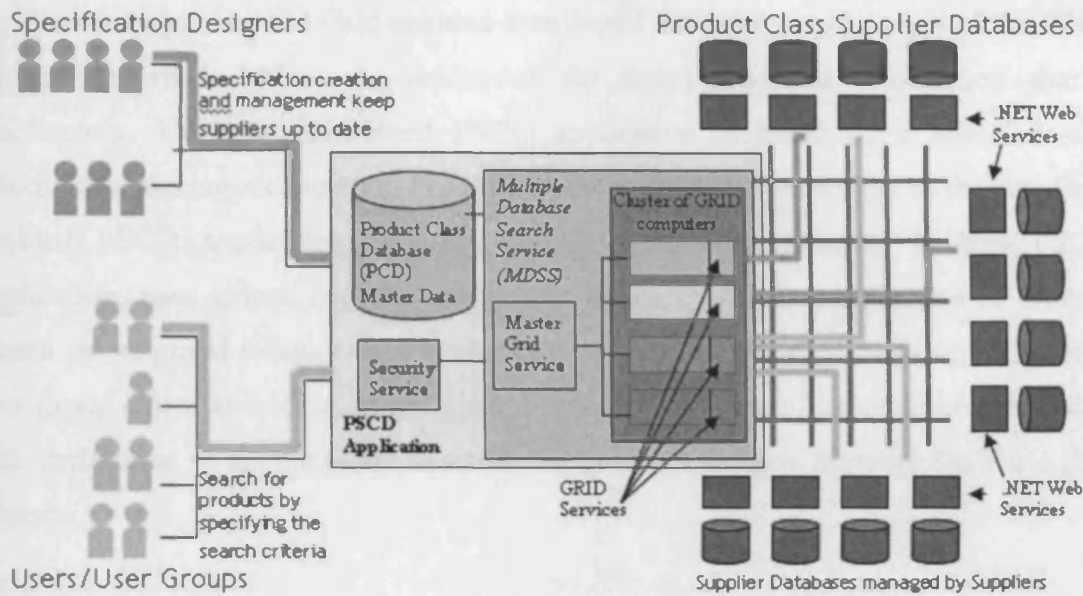


Figure 2.1 The conceptual view of the distributed architecture of the new Grid-based PSCD application.

Source: Burnap et al. [Bur05]

2.2 The Functional Areas of the COVITE Research

In the COVITE project, the research work for the Grid enablement of the PSCD application had been broadly split into three different functional areas which were also inter-related and interdependent. The functional area of security management investigated the security aspects of the application in order to provide secure access to the PSCD application via a user friendly web interface. Research in this area was led by the COVITE team member Liviu Joita. The security architecture designed in the project is explained in greater detail in [Joi04a], [Ran05]. The second functional area of user management was led by Pete Burnap. This functional area investigated what happens to a user once they have accessed the system and it determines what resources and services they are offered. Research in the user management functional area is presented in [Bur03], [Bur04], [Joi04a], [Bur05]. The author's research efforts were mainly concentrated in the third functional area of data management and the Grid enablement of the distributed database search of the new Grid-based PSCD application. This is a research contribution in the area of federated information sharing architectures and Grid-based architectures. The author's research is introduced in section 2.2.1.

2. The MDSSF Information Sharing Architecture

2.2.1 Data Management and Grid-enabled Distributed Database Search

The author's contribution to research in the area of federated information sharing systems occurred in the enhancement of the PSCD application. The author addressed the data management and Grid-enabled distributed database search issues of the PSCD application which led to the design of the novel federated information sharing architecture. The new distributed PSCD application is based on a new federated information sharing architecture. Figure 2.1 shows the conceptual view of the new Grid-enabled PSCD application and its various system components. In this PSCD¹ application, user groups such as contractors interact with the application in order to search for required product data available from several supplier databases. The other user group called specification designers interact with the application to create product data definitions to aid suppliers describe the products in their Supplier Database (SD) systems.

An important component of the PSCD application is the Multiple Database Search Service (MDSS) which lies in the core of the PSCD application and provides the functionality to federate a large number of supplier databases in a Grid environment. The MDSS is introduced in section 2.3. Since the MDSS federates a large number of supplier databases via a Grid based search, the novel federated information sharing architecture is termed the MDSS Federation (MDSSF). The author developed the MDSSF architecture as part of this research to Grid-enable the PSCD application and address data management challenges. The architecture was developed to support the need to provide product information to consortia for procurement of products and supplies, where this information is coming from several different product suppliers. A consortium continually needs up-to-date product information from several different suppliers so that they can make informed decisions about which product to use in construction projects. A new approach is needed if this information is to be provided in an integrated way so that several different suppliers can supply their product related data to contractors and potential buyers using a common standard via a single system. The approach must protect the autonomy of product suppliers, allow competing products to be located, compared and judged on the basis of their specifications against the project requirements. It must also protect the confidential information of different

¹ The PSCD application from this point onwards refers to the new Grid-based PSCD application.

2. The MDSSF Information Sharing Architecture

product suppliers. The MDSSF information sharing architecture exists through its components, section 2.3 therefore introduces the architecture through its architectural components.

2.3 The MDSSF Information Sharing Architecture

The MDSSF information architecture brings together autonomous contractors and suppliers. Figure 2.2 provides a conceptual view of the architecture. The architecture enables the creation of a Virtual Distributed Database (VDD) of product information where product information is supplied by a large number of suppliers using a standard data representation for each type of product. The VDD allows suppliers to provide product information, whilst holding this information locally in their local autonomous databases. A local product database system managed by a supplier supplies information in response to a request from a virtual database user such as a contractor. This local supplying of data will protect it and also ensure that it is up-to-date.

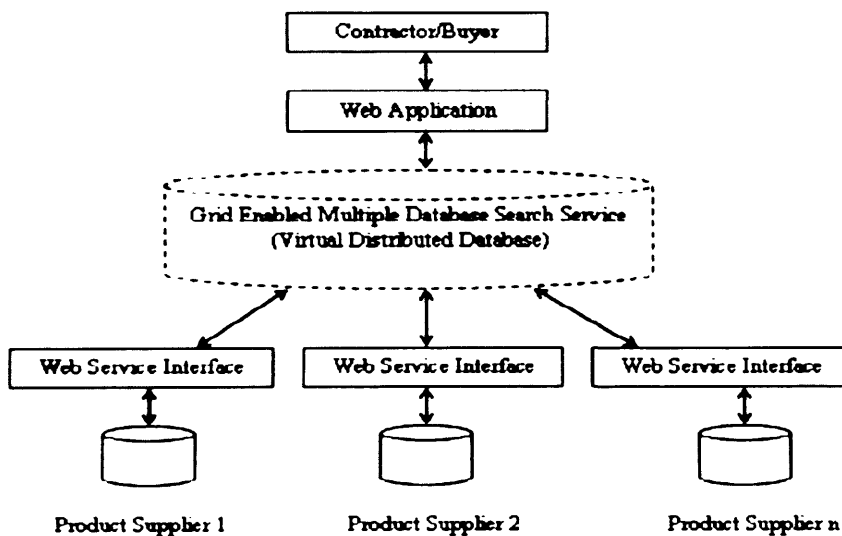


Figure 2.2 The conceptual view of the MDSSF

The architecture of the MDSSF is developed by utilising the features of federated database architectures, such as distribution of data and autonomy of local database systems (DBS) [She90] and coupling them with Grid technology to provide scalability support. In the VDD the federation of a large number of autonomous product supplier databases is achieved via a Grid based search to provide a scalable solution to contractors searching a large number of supplier databases. It provides a mechanism to perform a distributed database search of supplier databases in a Grid environment to

2. The MDSSF Information Sharing Architecture

retrieve product information of interest to contractors. The MDSSF federation model adopts a service oriented architecture for flexibility in retrieving data from the suppliers' databases and sharing it with contractors.

The information sharing architecture also supports a subscription based approach to address heterogeneity issues by providing product suppliers with access to a standard product data model. The subscription based approach can be defined as a mechanism using which product suppliers can gain access to standard product definitions and a product database. The product definitions, which are based on the schema of the standard product data model identify a set of criteria for product data management in local product supplier databases. A product supplier can use these product definitions by subscribing to them (i.e. downloading product definition criteria) and using them to store information about products in the local product database in a standard way. The subscription based information sharing architecture allows autonomous product suppliers to manage their product information but also make this information available to federation users such as contractors via the VDD. The subscription based approach is required for two important reasons:

- First, it tackles the issue of heterogeneity by providing product suppliers with a pre-defined and pre-designed product database and standard product definitions which can be readily used by them to provide product related information in their database by describing the product's features/specifications in terms of product definition criteria.
- Second, the VDD of the federation can contain a large number of product definitions to support different types of products – all of them may not be required by an individual product supplier. For example a furniture supplier may require only those product definitions which allow the supplier to describe furniture equipment in its local database. A product supplier therefore, based on need subscribes to a certain number of product definitions only and then uses these to create and publish product information in the VDD of the federation.

The above means that publication of product information by product suppliers in the VDD enables the creation of a real time product catalogue system which contains

2. The MDSSF Information Sharing Architecture

product data provided by a large number of autonomous product suppliers. This creates an environment for product suppliers to compete with each other in a virtual market place based on the product information provided by them. This VDD can only be built using federated database concepts as the autonomy of the product suppliers must be protected. The aim of this research is not to build a product catalogue system but to create a new information sharing architecture based on federated database architecture concepts (which enables such product catalogues to be built) in order to support product procurement processes in the construction industry.

To address the procurement challenges, the development of various system components occurred as part of the MDSSF architecture. These components provide mechanisms for product data definition by industry experts, management of product data by suppliers, and for making it available via a Grid enabled distributed database search to contractors. Thus the components of the novel federated information sharing architecture address the requirements of the three fundamental areas of the PSCD application such as data definition, data management and data search. This section introduces these components.

2.3.1 The Product Class

A mechanism is required to create standard product definitions which can be used by the product suppliers to describe products in their databases. These product definitions are called product classes. Product classes provide a set of criteria for product data management by product suppliers. A product class is made up of several different specifications to allow description of different product attributes in a supplier's SD System. The concept of product class acts as the fundamental means and provides the logical basis to address the heterogeneity problem and a standard means to exchange product data in MDSSF. A product class can be used by a large number of product suppliers to describe product features from the specifications in their databases. The concept of product class and its specification types is described in greater detail in Chapter 5. Section 7.2.2 describes in detail how a product class is created by means of examples.

2. The MDSSF Information Sharing Architecture

2.3.2 The PCD System

The product classes are created in the PCD (Product Class Database) system. The PCD system is a database centric tool which allows its users, the independent and knowledgeable industry specification designers and/or industry experts to create different types of specifications as part of creating a product class by using the constructs of the PSCD front-end web application and the stored procedures of the back-end PCD system. These specifications provide a mechanism to define different attributes which a product may have. For example a product class corresponding to furniture item, such as a chair, can have a number of specifications such as *width*, *height*, *weight*, *chair description*, *wood type* (in case of a wooden chair). A product supplier of furniture equipment can then use these specifications to describe the features of a product and so populate the predefined specifications with values describing the actual product. Hence the PCD system enables specification designers to create new product classes or new versions of existing product classes. The PCD system stores information regarding product classes, product categories and product specifications and uses these to facilitate the description of actual products by product suppliers. The PCD system supports the creation of different types of product classes to describe different types of construction industry products with the aim that these product classes will be used by product suppliers to describe products in their databases and allow search for these products by contractors. In the MDSSF it is a requirement that all product suppliers adhere to the schema of the PCD system to describe products in their databases. In this respect the schema of the PCD system acts as the common data model (CDM) of the MDSSF. The architecture of the PCD system is described in greater detail in Section 6.2.1.

2.3.3 The SPCD and the SD systems

The PCD System enables the creation of product classes which are subscribed by the product suppliers in their locally installed PCD system. This PCD system is called the Supplier-side PCD (SPCD) system. Subscribing to a product class means downloading the specifications and its values from the central PCD system and populating these in the local SPCD system which is managed and controlled by the autonomous supplier. Once a product class is downloaded into the SPCD system, product suppliers can describe the features or specifications of their products in their Supplier Database (SD)

2. The MDSSF Information Sharing Architecture

systems by referring to the product class specifications downloaded in the SPCD system. It is a requirement of the PCD System that all product suppliers use databases which are identical to the PCD system i.e. they conform to the schema of the PCD system. This occurs because product classes can only be downloaded to the database system if it has a schema which is identical to the schema of the PCD System. Therefore standard SPCD and SD systems were designed for this purpose with the aim of providing suppliers with a readily available means to describe their product data in a structured and standard way. The schema of the SD system is also similar to the schema of the PCD system for the reasons given above. Hence in order to describe products, product suppliers require two databases, the SPCD system and the SD system. The relationship between the SPCD and the SD systems is illustrated in Figure 2.3 and these systems are described in greater detail in Section 6.2.2.

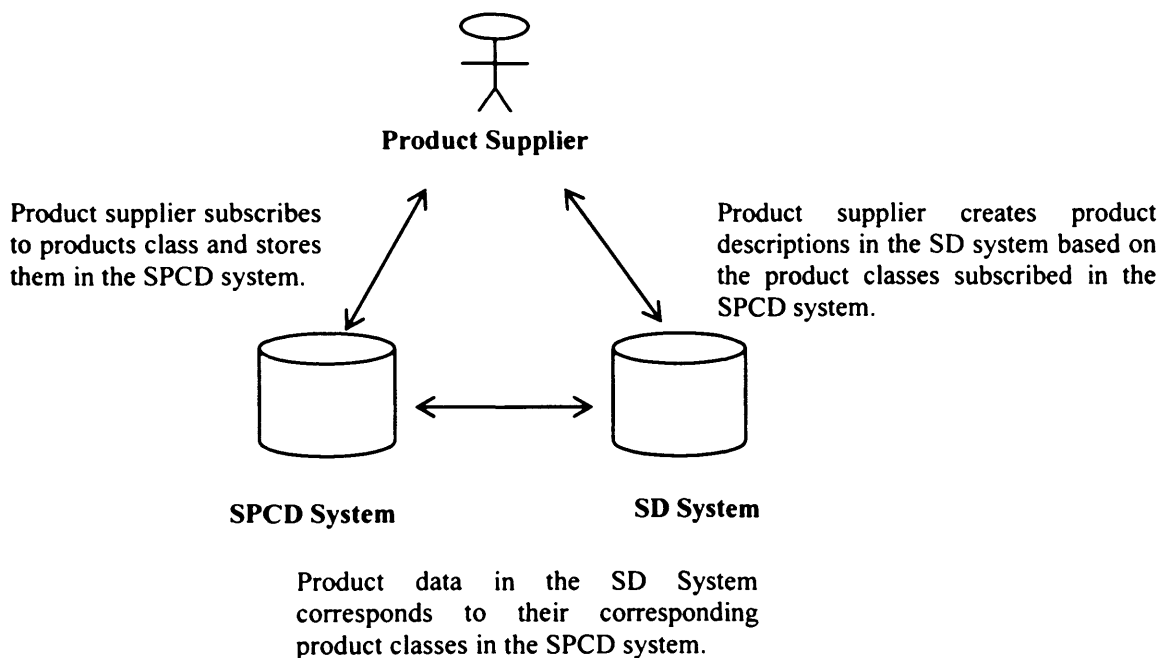


Figure 2.3 Relationship between the SPCD and the SD systems at the product supplier end

2.3.4 The Multiple Database Search Service (MDSS) System

The MDSS System retrieves product data from a large number of autonomously managed SD systems belonging to individual organisations. The MDSS provides a Grid service solution for processing large amount of data by utilising the Grid middleware Globus Toolkit [Fos98] 3.0.2 (Core) which is based on the Open Grid Services Architecture (OGSA) [Fos02]. It invokes a dynamic selection of relevant supplier

2. The MDSSF Information Sharing Architecture

databases to extract, in real time, detailed information about the products which the user wishes to acquire. Figure 2.2 provides a conceptual view of the MDSSF where the Grid-enabled MDSS at the core (middle layer) enables formation of a virtual distributed database (VDD) to provide product data from the product suppliers to the contractors in the standard schema of the PCD system. The VDD is so named because it is a search facility and does not store any product data but provides such data when requested by the contractors by querying the SD systems in real time. An important requirement of the PSCD application is that product suppliers retain full control of their data so it cannot be replicated outside their domain. The architecture of the MDSS system is described in greater detail in Section 6.3.

2.4 Novelty of the Proposed Information Sharing Architecture

The MDSSF is made up of different system components. These system components gives certain distinctive features to the architecture, which are part of its novelty and provide a new federated information sharing model enabling a novel type of collaboration between construction industry actors. MDSSF is designed to provide Grid technology support for retrieving product information and sharing it with contractors in real time. It provides an integrated way to access product information from a large number of product suppliers using a single system. MDSSF adopts a different model of cooperation from traditional FDBS architectures, which enables data sharing between the component DBSs participating in the federation. The MDSSF model of cooperation does not allow sharing of data between component DBSs supplying the data (i.e. between the SD systems of the product suppliers). In the MDSSF architecture product suppliers share their data with the contractors only. This is because product suppliers are business organisations, who do not wish to disclose their product related sensitive data to their competitors, who are participating in the same federation. They only cooperate with the centralised MDSS so that appropriate information about their products is retrieved and sent to the contractors in the standard data structure of the VDD. The MDSSF information sharing model is different from other federated database or mediator based architectures with respect to schema integration. In MDSSF, unlike other information sharing systems (see Section 4.4) the sharing of data takes place between contractors and the supplier databases without the need to create federated, integrated or external schemas. This schema integration is not required in the

2. The MDSSF Information Sharing Architecture

MDSSF approach, because of the homogeneous nature of the product data which is exchanged in MDSSF. The MDSSF is also a novel information sharing architecture from the perspective of its data model and data subscription. The architecture (via its architectural components) allows subscription to the data model of the federation and product classes by the product suppliers (see Section 6.2.2.1). This subscription based approach provides product suppliers with readily available mechanisms to manage their product data in a structured way by using the standard specifications of product classes to describe product features and in this manner tackle the issue of heterogeneity. The novel features of the MDSSF are described in greater detail in Section 7.4.2.

2.5 Chapter conclusions

This chapter introduced the MDSSF information sharing architecture through its architectural components. The MDSSF information sharing architecture is a key contribution of this research. The MDSSF federates a large number of supplier databases and allows them to be searched via a Grid-based distributed database search. The aim of the MDSSF is to address the procurement challenges identified in Section 1.2. For this purpose it provides components for data definition, data management, and data search in a Grid-based distributed environment. The next chapter provides background information on construction procurement, the role of information and technology in modern information-driven construction environments and some of the potential benefits of this research.

3. Construction Procurement

3.1 Introduction

Construction procurement plays an important role in this research. Construction procurement is considered as an exemplar study area to identify how new models of information sharing can bring about increased benefits to actors who are associated with procurement related activities. The construction industry has made significant advances in recent years to embrace technological options which enable information sharing. This sharing of information has not only led to increase in the efficiency and physical output of the industry as a whole, but has also paved the way to bring about changes in work practices and ways of collaboration. New ways of information sharing cannot be created or brought into practice without understanding the existing and well-established information related modes and practices prevalent in the industry. From this perspective, understanding the role of information in the construction industry is fundamental to this research.

It is also important to understand the role technology plays in the construction industry because it enables information sharing. Modern technology provides advanced means of sharing and managing information via electronic means, which enable its quick dissemination. The volume of information generated by construction projects is so large, that using IT to handle it is highly beneficial. Technological options available via computer-based systems enable creation of information systems to meet the information needs of the organisations. Information systems provide a means of managing a large amount of information and sharing it with appropriate people. Therefore, to fully appreciate the role of information in the construction industry, it is important to understand the role technology plays in making information available at the right place and time to serve the crucial purpose of aiding decision making.

Last, but not least, the role of various information system based techniques, mechanisms, approaches that are currently used for procurement as part of the different traditional and modern procurement systems used in the construction industry also cannot be ignored or underestimated. The applications developed, based on the information sharing architecture proposed as part of this research, will have to find their

3. Construction Procurement

place alongside these information systems to aid the existing procurement systems in the construction industry, if this research is to be put to use. From this perspective and for the purpose of adaptability, looking at the existing information systems and practices that drive construction procurement is vital. This chapter therefore has two objectives. The first is to identify, how the existing procurement systems could benefit from an integrated mechanism for sharing product information from different suppliers. This includes identifying the technological options and information systems designed and used for procurement and the information tools which will be used to create the proposed information sharing architecture. The application developed based on the proposed architecture, should use similar technological options for the purpose of compliance, and should integrate with the existing systems with minimum effort. This will enable existing applications or procurement systems to use the services of the information sharing architecture proposed as part of this research in order to retrieve product data from a large number of supplier databases. The second objective which is more crucial is to establish, whether there is a real need for such data sharing architectures. As part of this objective we address how the proposed data sharing architecture can benefit contractors and procurement managers in their desire to make better procurement decisions.

In this chapter a review of the current material procurement strategies used for E-Commerce systems is provided. It will look at the present procurement methods and techniques and identify how the proposed procurement mechanism relates to the existing procurement methods. To improve the performance of Engineer-Procure-Construct (EPC) projects, a number of engineering models can be used. Some of the models identified by Yeo and Ning [Yeo02] include: fast-track, concurrent engineering, JIT Logistics Management, Business Process Re-engineering and Partnering. In this research we looked at the product procurement models that are used in the construction industry. The aim is not to introduce a new model of the procurement process but to aid the existing models by proposing a new model of information sharing and collaboration.

The chapter is organised as follows. Section 3.2 provides an overview of the construction supply chain. Section 3.3 focuses on procurement, and identifies the importance of procurement and sub-contracting in construction. It provides a brief

3. Construction Procurement

overview of different procurement systems. Section 3.4 identifies the role of information in the construction supply chain and in Section 3.5 we learn the role technology plays in modern information-driven construction environments. An important role of technology is to interconnect various processes of the supply chain to provide its actors with information about the flow of goods and services so that decisions can be made about various aspects of the project. In Section 3.6 we identify how the MDSSF information sharing architecture proposed in this research can benefit organisations in different scenarios when making procurement-related decisions. Section 3.7 describes different approaches for product procurement, which are used by construction industry practitioners or proposed by researchers in the literature. In this section, how the MDSSF data sharing architecture can fill some of the gaps in the procurement systems and approaches is described. Chapter conclusions are presented in Section 3.8.

3.2 Background

3.2.1 The Construction Supply Chain

Supply Chain Management is a process managing the movement of resources from the source of supply to the point of use [Chr98]. Baily et al. [Bai05] define supply chain as:

“The supply chain includes all those involved in organising and converting materials through the input stages (raw materials), conversion phase (work-in-progress), and outputs (finished products).”

A construction supply chain links all the parties that are participating in a construction project [Che01]. The construction process involves all activities, whether technical, managerial or strategic which interact to bring about the realisation of the project, where a physical facility actually appears [Row99]. Within the construction industry, there are different types of supply chains for each of its specialised areas, and these overlap with each other. Some of the major supply chains include construction integration, professional services, materials, equipment and labour [Cox02]. Figure 3.1 shows the key phases of a construction supply chain as identified by Edum-Fotwe [Edu01]. The phases define the lifecycle of a facility from its conception to decommissioning and give a high level view of the activities that may take place in these phases. This research

3. Construction Procurement

does not take into account all these phases of a construction supply chain. It only focuses on the procurements aspects of the construction supply chain to identify how a new model of collaboration for data sharing can provide benefit to the construction industry actors who are involved in procurement related activities.

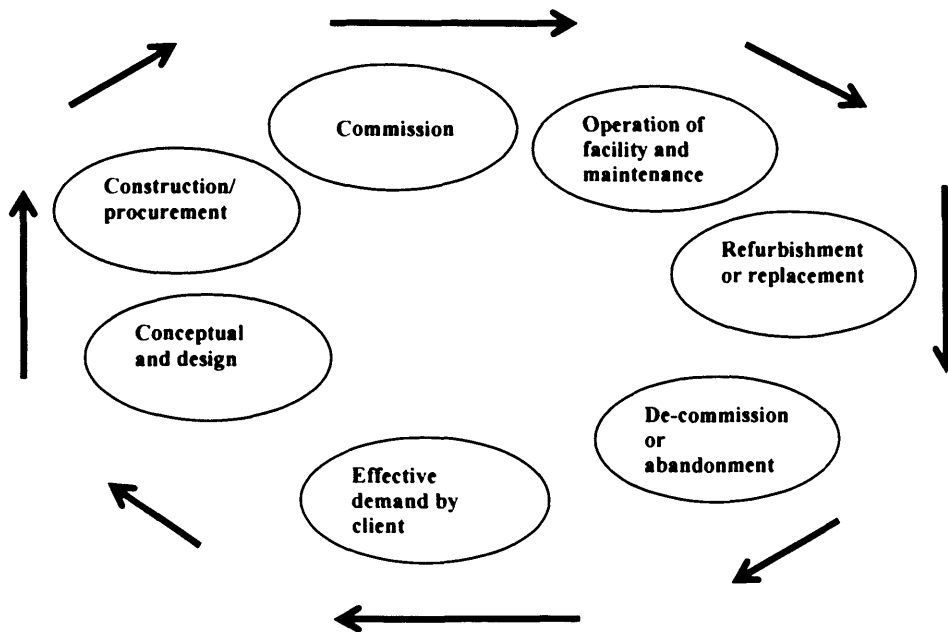


Figure 3.1 Key phases of the construction supply chain
Source: Edum-Fotwe et al. [Edu01]

3.2.2 The Engineer-Procure-Construct (EPC) Projects

In the construction supply chain, EPC projects are complex undertakings, which consist of a large number of interconnected subsystems and components requiring considerable human efforts and financial commitments [Yeo02]. An EPC project is unique; complex in nature because of involvement of players from different institutions; generally with a long duration period; and is contractual, where a contract is made between a client and a contractor which identifies delay, cost and other project specifications [Mah04]. It is the increasing magnitude, complexities and risks associated with major construction projects, which bring together organisations having diverse strengths and weaknesses to form a joint venture or consortia to collectively bid for and tackle these large scale projects which, cannot be handled individually [Kum00]. These large scale projects require new technologies, knowledge of local practices, financial strengths, specialist or experienced staff, and integrated procurement arrangements [Kum00]. In order to bring together all this expertise and knowledge requires different organisations who are

3. Construction Procurement

specialist in their area to work together by forming consortia. The activities of EPC projects are time-phased according to a specified precedence, resource requirements and constraints. A brief description of the engineering/design and construction phase of EPC projects follows, while the construction procurement and related activities are described in Section 3.3.

3.2.2.1 Engineering/Design Phase

In the Engineering/Design phase, the needs of an owner or a developer are defined, quantified and qualified into clear requirements, which are later communicated to builders or contractors [Yeo02]. Detailed engineering designs and plans are made by designers and architects. It progresses in stages, namely client briefing, conceptual design, preliminary design and detail design [Bla92]. As this phase has the highest level of influence on the project, key decisions regarding the design of the building or a facility are made at this stage. Commitment of funds and resources required for carrying the project to a successful completion are made at this stage. Early cost commitment may be made on the basis of incomplete documents which will go through redesign and design iterations based on concepts such as value engineering and constructability analysis until the contractor and sub-contractors have a cost-effective construction document [Pie97]. The construction activities may start before the design is completed. The contractor is selected before the completion of working drawings and specifications, and some speciality contractors can also be hired before the final agreement about price is reached [Pie97]. At this stage, in addition to having access to cost related information of other project aspects it is important for the contractor and sub-contractors to have access to the cost of materials and products, which need to be procured and which will be used for constructing the required facility.

3.2.2.2 The Construction Phase

The construction phase starts when the contractor begins to construct specified facilities according to the work packages prepared during the engineering phase and use equipment and materials obtained during the procurement phase. The outcome of this phase is a completed building or facilities which is fit for a given purpose and meets the client's requirements identified before the start of the construction phase or while the construction is in progress.

3.2.3 The Need for Improvement in the Construction Industry

Although the UK construction industry has been described as leading-edge in its ability to deliver challenging and pioneering projects, it is still perceived as under achieving [Ega98]. It has low profit margins and there is little investment in terms of research, development, training and capital [Ega98]. Agapiou et al. [Aga98] identify that the profit margin on construction work is just 1-2% of the construction price. There is still considerable room for improvement of the processes and operations of the industry as a whole, which needs to improve its productivity and increase its efficiency and solve its problems such as budget overruns, delays, poor quality work and failure in meeting client's requirements. The Egan report [Ega98] proposed that the UK construction industry should reduce its annual construction cost and construction time by 10%. The Latham report [Lat94] called for an improvement in productivity by as much as 30%. Although many of the challenges identified in the Egan and Latham reports have been addressed or are in the process of being addressed what has not changed is the complexity of construction projects and procurement planning and there is always a need to increase efficiency. A good proportion of procurement is still paper-based and the use of catalogues, referrals and personal contacts is widespread. In order to reduce costs, increase productivity and improve efficiency the role of IT is recognised as essential [Ega98]. Considerable benefits can be gained by using IT effectively. This will help in eliminating waste and rework, and enable rapid exchange of information [Ega98]. There has already been a widespread adoption of technological options and e-Business within the construction industry to replace the traditional methods of working and relationships between construction partners [Sfc02]. These technological options can bring about efficiency; economy and speed; improved business relationships; and improvements in the products, processes and operations [Sfc02].

3.3 Procurement

In Section 3.2.1 we identified that within the construction industry there are different types of supply chains such as construction integration, professional services, materials, equipment, and labour. In all these supply chains, procurement plays an important part, as it is a channel for sourcing all the needed input to transform the architectural plans and drawings into a physical product based on a client's requirements. In simple terms procurement can be defined as an activity that deals with acquisition of project

3. Construction Procurement

resources for the realisation of a constructed facility [Row99]. An important objective of procurement is to ensure that all the resources are acquired effectively [Har01]. Procurement can be conceptually illustrated (Figure 3.2) [International Labor Office [Aus84]]. The figure illustrates that a construction project can require different types of resources which can include human and physical resources in the form of skills, expertise, construction equipment, materials brought by different members of the consortium or construction industry participants who come together to undertake the project.

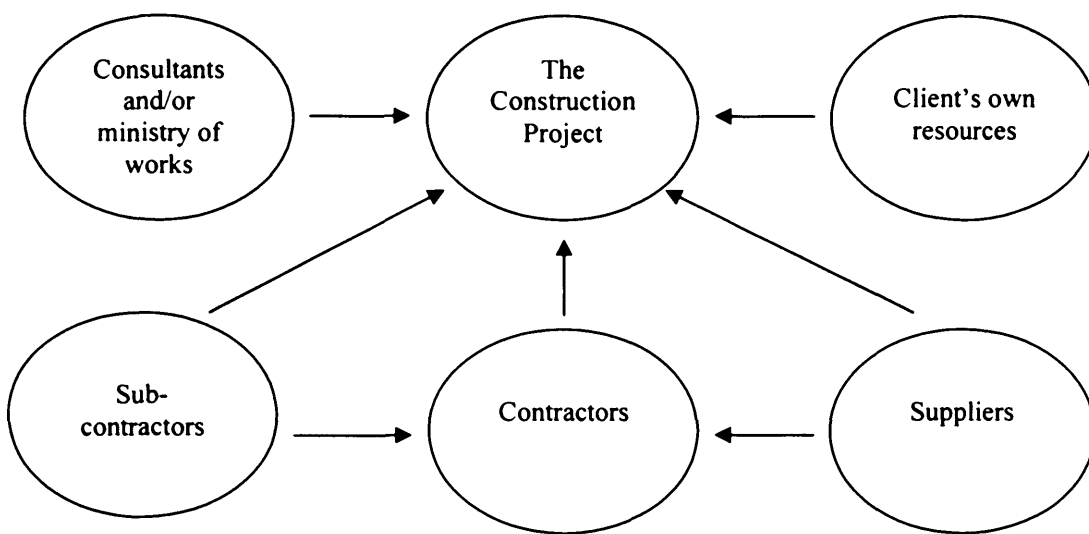


Figure 3.2 Procurement
Source: International Labor Office [Aus84]

This shows that procurement can play an important role in the construction supply chain as it provides a mechanism to assemble required resources from various sources for delivering the industry's physical output. However acquisition of resources for constructing a specified facility forms only part of the entire procurement process. Important activities of procurement also include sourcing, purchasing, contracting, on-site material management and a host of other activities. These activities are identified by Barrie and Paulson [Bar92] in their elaborate definition of procurement:

3. Construction Procurement

“Procurement includes purchasing of equipment, materials, supplies, labour and services required for construction and implementation of a project. It also includes related activities of tracking and expediting, routing and shipment, materials and equipment handling, accountability and warehousing, final acceptance documentation, and ultimate disposal of surplus items at job end.”

Important aspects of procurement also include selecting professional teams and contractors to undertake construction projects, and dealing with contractual attributes which impact construction time, cost and quality [Har01]. A contractor procures project equipment and construction materials during the procurement phase after the receipt of engineering drawings, specifications and other relevant documents [Yeo02]. Depending on the needs of the construction project, the procurement related activities are carried out in all phases of a project starting from the design phase through to the completion phase, during which the constructed facility is ready for operation and is handed over to the client [Bar92]. In some projects, procurement related activities such as procurement of project specific equipment, products and materials are carried out after the completion of the design phase and before the start of construction phase. Procurement activities may also be carried out by owners after the completion phase for facility maintenance and upgrade. Most of the procurement occurs during the construction phase whilst the construction is taking place. In certain circumstances, procurement takes place at the last possible moment so that an accurate quantity of required materials and products is sourced. This eliminates wastage of resources and reduces warehousing or inventory costs. In any construction project there are many stakeholders such as owner, designer, contractor and sub contractor. Major procurement for construction can be handled by one competent stakeholder or can be split between the stakeholders.

3.3.1 Sub-Contracting Arrangements

A high degree of specialisation has taken place in the construction industry for the provision of various goods and services which has led to a network of supply chains that include multiple layers of sub-contractors and interlinked suppliers [Kum00]. Edum-Fotwe et al. [Edu99] also identify that the use of subcontracting arrangements in the construction industry is quite extensive. Organisations focus on their core competencies to achieve higher productivity gains and value added niches in their area and outsource

3. Construction Procurement

all other subsidiary functions to other organisations, which are better in performing these functions. This has also led to the inter-linking of different organisations in the construction supply chain, when undertaking a construction project.

In sub-contracting arrangements, the main contractor appoints sub-contractors to deliver certain “*packages*” in a given time frame, which are then integrated in the “*solution*” [Cox02]. Sub-contracting is vital because the work that takes place in the industry is very diverse in nature and structure and requires the expertise of many specialists who come together for the duration of a construction project. It is not possible for the main contractor to have all the expertise and knowledge of the many aspects of a construction project, therefore the majority of the work is outsourced to different small-size firms through subcontracting arrangements and the main contractor only retains a small portion of the work [Edu99]. The main contractor tends to keep the staff employed for construction projects to a minimum because the workload can fluctuate greatly from one project to another. Additionally construction projects are not confined to a specific geographic area; movement of labour and other resources to new sites is required. Since all these factors can add to the cost, subcontracting arrangements with local sub-contractors or suppliers can help to keep the costs down. Multiple layers of sub-contracting have evolved in many countries which means that even sub-contractors can allocate or outsource portions of work packages allocated to them to external specialists. Product suppliers who usually feed these sub-contractors or sub-sub-contractors may themselves be customers of other specialist suppliers hence extending the construction supply chain even further.

Sub-contracting arrangements are an important part of the procurement to be performed at several levels. Major equipment such as capital equipment from national/international suppliers which has a longer lead time may be ordered in advance by the owners in order to ensure timely arrival. The items of a less critical nature which do not require heavy capital investment, such as furniture items and electrical fittings are ordered by the contractor or sub-contractors involved in the program. Specialist sub-contractors involved in the project, such as lift installers, air conditioning experts and electricians use their own channels for procurement. The field office at the construction site normally procures supplies, incidental rentals and other requirements as required for

3. Construction Procurement

day-to-day construction work [Bar92]. Construction materials typically account for 40-45% of all costs of construction work [Aga98]. Therefore in order to make the best decisions different actors use their knowledge and expertise to procure products and services from the wide range of available suppliers. Hence the responsibility of procurement is split between various actors depending on the role and the level in which they participate in the construction supply chain.

Procurement is usually made from the best available sources. One way of selecting the best available suppliers is via the process of bidding. Whilst procuring supplies for a construction project a general contractor may receive bids from subcontractors, material suppliers and equipment manufacturers who refer to the completed plans and specifications before bidding and are in a position to supply the products needed according to these specifications [Bar92]. Requests for quotation can also be made by the owners to a potential contractor or sub contractors. This can include advertising in the local trade journal to request potential contractors/sub-contractors to submit their bid package. For this the owner or designers acting on behalf of the owner may provide plans, specifications and other contract documents fully identifying all aspects of the proposed purchase [Bar92]. A contract may be awarded to a supplier or a consortium of suppliers who make the most competitive bid and are able to meet project requirements such as costs and delivery time. Sub-contracts may also be awarded by the contractor or manager to undertake on-site work [Bar92].

3.3.2 A Brief Overview of Building Procurement Systems

There are different types of procurement used in the construction industry. Harris and McCaffer [Har01] and Masterman [Mas02] have defined four different types of building procurement systems in current use: *separated*, *integrated*, *management-oriented* and *discretionary systems*. Franks [Fra98] has also identified building procurement systems which are similar to these. In the *separated* procurement systems separate organisations such as design consultants, architects, quantity surveyors, contractors, structural engineers and planning supervisors are responsible for the design and construction work. The client has the overall responsibility of funding and managing the project and is also responsible for the operation of the facility after it is completed [Mas02]. Traditional contracts come under this category [Har01]. In *integrated* procurement

3. Construction Procurement

systems there is one organisation with the entire responsibility for the design and construction of a facility and therefore the client only has to deal with one organisation for its needs. Some of the systems in this category include design and build, develop and construct, and package deal [Mas02] [Har01]. In *management-oriented* procurement systems, the construction manager or managing contractor gets involved with the professional team at early stages before the start of the construction work and helps in the overall construction programme and its work packages. The management takes an active part in the project, right from its inception to ensure that the project finishes on time, within budget and is of acceptable quality and serviceability [Har01]. Examples in this category include: construction management, management contract, and design and management [Har01]. In *discretionary* systems the client has a greater involvement in the administration of the project and also has the discretion to choose the most suitable procurement system from the three categories identified above [Mas02]. The aim is to ensure that clients are able to define and express their needs right from the start and that the building or facility which is developed matches a client's expectation and is fit for purpose. Partnering, alliances and joint venture are some examples which are in this category [Har01].

Edum-Fotwe et al. [Edu99] identified that all the procurement systems described above are dependent on sub-contracting arrangements. These sub-contracting arrangements are made between the principal contractor and a number of sub-contractors, who are responsible for delivering the work packages in their area of expertise. In the construction supply chain these contractors and sub contractors are ultimately chained to the suppliers (directly or via other intermediaries such as sub-sub-contractors and procurement agents) for the procurement of products and services. Therefore the sharing of information via the MDSSF architecture proposed, as part of this research can be beneficial to all the building procurement systems identified by Harris and McCaffer [Har01] and Masterman [Mas02] by providing contractors or sub-contractors with access to product related information from suppliers.

3.4 The Role of Information in the Construction Supply Chain

Information plays a vital role in all the processes within the construction industry. In this section we will identify from various perspectives the role information plays in

3. Construction Procurement

bringing together the construction industry processes and enabling smooth functioning of its operations. The aim of providing this information here is to show the current role of information in the construction supply chain. It is only by understanding the current role of information that the case for new ways of presenting information can be justified, which bring increased benefits to industry participants. Another important aim is to identify how the MDSSF data sharing architecture can provide benefit to contractors, procurement agents and other actors in the construction industry who wish to procure products from a wide range of product suppliers. From the arguments presented below it will be established that information plays a vital role in the construction industry. Hence a new way of providing information using an integrated approach such as the MDSSF can benefit construction industry actors performing procurement-related activities, by providing relevant and timely information.

3.4.1 Information is a Strategic Resource

The role information plays in the construction supply chains has shifted from a passive function in decision making to a strategic resource, the effective utilisation of which enables the smooth operation of various processes in a supply chain and has a direct impact on the competitiveness of an organisation [Edu01] [Har01]. The elevation of the role of information to a resource from a mere enabler of processes in a supply chain took place over the last two decades [Edu01]. McCreadie and Rice [McC99] view information as a resource which can be “*produced, purchased, replicated, distributed, manipulated, passed along, controlled, traded, and sold*”. Information is a resource within the construction industry because it exists in diverse forms such as drawings; specifications; and in a communication mode which provides conditions, explanations and clarifications between parties [Edu01]. [Edu01] further identifies that it forms the bedrock of the construction industry’s production activity. If up-to-date information from different sources is presented in an integrated way it will become a strategic resource for practitioners to refer to and use to assist in decision making.

3.4.2 Flow of Goods and Services

Construction is an information transaction process [Edu01]. It is not just a material conversion process where input materials are converted into a final product such as a constructed facility or a building but it is also a process in which information as a

3. Construction Procurement

resource is combined with other resources to generate physical products [Edu01]. In this respect the flow of information within the various processes of the construction industry and between its key players at various stages of a supply chain is equally important as the flow of physical goods and services. It plays an important role in procuring the right kind of materials for a given construction project from across a large number of suppliers. Saunders [Sau97] identifies that personnel in purchasing and supply management are, from one perspective information processors as they receive, analyse, make decisions and distribute information in order to manage the flow of goods and services in a supply chain.

“At the heart of the transactions that take place in supply chains within construction is information.” - Edum-Fotwe et al. [Edu01].

“Information allows, forbids and directs the physical flows, and also enables the checking and confirming as well as provides proof and audit trails for transactions.”
- Edum-Fotwe et al. [Edu01].

“Executives in the industry implicitly accept that information is a key management resource and underpins the processes and operations of every construction company.”
- Haris and McCaffer [Har01].

3.4.3 Linking Actors in Construction Supply Chain

In the Architecture, Engineering and Construction (AEC) industry, a typical large project requires the direct and indirect participation of many specialised firms, each in charge of a functional task in the design or construction process [Pie97]. Hence a construction supply chain brings together different stakeholders known as actors who are working in its different phases. Actors such as contractors, project managers, suppliers, financial institutions and regulatory bodies participate in the construction phase [Edu01]. These actors are connected to each other via the information links and communication channels for disseminating information about various aspects of a project. A list of actors participating in various phases of a construction supply chain is identified by [Edu01] as information actors who *“generate and provide or acquire and process information to facilitate the activities of the particular phase in which they*

3. Construction Procurement

participate". These actors also use information available from several external sources such as trade journals, newspapers and supplier catalogues to learn more about the market conditions and remain competitive.

3.4.4 Coordination and Collaboration

Sharing of information in the construction industry pertaining to different aspects of projects is vital to coordinate its development. This sharing takes place between the organisations which come together and undertake a shared responsibility to develop a facility. The information which was previously considered sensitive and confidential is also increasingly shared through arrangements such as partnering to achieve improvements both at the company level and throughout the supply chain [Edu01]. The importance of various types of information is further reflected by the growth of information sources in the past decade. There has been an exponential growth of information sources available to technical and managerial executives, which reflects the growing importance of information to the operational success of organisations [Edu01].

"Emphasis these days is on information sharing between systems of different organizations. It is expected that enabling information sharing between different parties in the construction material procurement process can facilitate improved information communication and coordination, better strategic planning and decision making, and rapid and flexible supply chain management." – Kong et al. [Kon04].

Information sharing enables buyers and suppliers to conduct their relationship in a better way thereby developing mutual trust and collaboration in an objective manner in the face of uncertain circumstances and no knowledge of the motives of the party at the other side [Cox02].

3.4.5 Sellers and Purchasers

Information sharing brings together sellers and purchasers. Kong et al. [Kon04] identify that both purchasers and sellers can benefit from information sharing as purchasers can get more comprehensive information about the materials they wish to buy and the sellers can have information on the current market situation. The current market environment actually promotes and encourages different organisations to work together

3. Construction Procurement

to optimise the flow in a supply chain. Organisations along a supply chain are linking their information systems in order to form inter-organisational systems [Kon04]. These systems are networks of systems which allow organisations to share information between each other and interact via an electronic means across organisational boundaries.

“By enabling information sharing between different E-commerce systems for construction material trading, buyers and sellers may operate with lower levels of ambiguity and uncertainty due to the provision of greater volumes of timely and accurate information, thereby enabling them to make more efficient and effective decisions” – Kong et al. [Kon04].

3.4.6 Role of Information in Other Areas of Construction

In addition to the areas identified above, information also plays an important role in other areas of construction such as in the design and construction phases. At the design phase, exchange of information enables interaction between clients and construction industry actors such as architects, structural engineers, and quantity surveyors to capture the scope and requirements of the project and outline the plans, which then become the basis for design documents [Har01]. Similarly, at the construction stage, information transactions take place between various actors to transform the designs into a constructed facility. For example, information transaction between contractor and a sub-contractor; a sub-contractor and material suppliers, etc. [Har01]. These information transactions enable coordination of various aspects of the construction project and aid in the timely delivery of the construction phase [Har01]. Even in these areas involvement of suppliers and access to the information pertaining to their products and services is essential so that right from the conceptual stage, accurate decisions can be made by contractor and sub-contractors, by taking into account external factors such as availability of products and services, their specifications, cost, quality, and delivery time.

3.5 Role of Technology in the Construction Supply Chain

IT and e-business have already *“radically transformed”* many operations in the construction sector, but there is still a vast scope for improvements [Sfc02].

3. Construction Procurement

Technological options enable getting the right kind of information for making crucial decisions, which impact the progress and performance of the processes of the construction supply chain. IT has now become a technology of strategic significance in modern business from a position of being a support technology [Bet99]. In the construction sector, like any other sector, IT is now being applied in all application areas for a variety of functions and for communication between organisations [Bet99]. Understanding the role technology plays to manage different types of information in the construction industry is vital to this research, because the MDSSF information sharing architecture presents a technology based solution which brings together and provides an integrated view of product data from several supplier databases. New technology-based solutions cannot be designed without understanding how modern technology is helping construction practitioners achieve their objectives. Understanding the role of technology is also important from the perspective of identifying new ways of addressing current challenges, such as dealing with the large amount of information which is available from several different autonomous sources. In this section, we present an overview of the role technology plays in managing construction industry information from several different perspectives.

3.5.1 Use of IT is Widespread in the Construction Industry

Pietroforte [Pie97] identifies that the introduction of IT in the AEC industry has significantly shortened the development of new products by allowing the simultaneous implementation of different functional designs and engineering tasks. It is one of the reasons behind the widespread use of IT for several different purposes. Construction-related documents, spread sheets, image files, CAD drawings, blue-prints, specification documents, database files are widely created, used and shared between project members, through the use of different types of software packages. Sharing of information means communication with project members various types of information such as drawings, design documents, specifications, calculations, cost estimates and schedules. There has also been a widespread increase in the use of technology for data related functions such as data storage, data mining, data archiving and data analysis and communication to inform various processes of the construction supply chain. The use of internet or intranet to provide access to electronic notice boards gives the potential to effectively provide up-to-date project status information to project participants

3. Construction Procurement

[Wal99a]. Completed projects also leave a trail of enormous amounts of information in the form of written, graphic and numeric documents, which must be preserved for future reference and use through IT systems. Additionally, technologies for Electronic Data Interchange (EDI), bar coding, visualisation (including CAD, Virtual reality (VR) and Augmented Reality), communication (including data/video conferencing, intranets, electronic mail, file transfer, telnet) and integration technologies (for project document management, data warehousing, and development of industry wide information repositories) are widely used across many spheres of construction activities [Har01]. IT is also being applied actively in automation of contract administration [Pie97].

3.5.2 IT enables Cooperation and Information Sharing and Communication

Pietroforte [Pie97] recommends that technology should be used for establishing electronic links to enable inter-activity, simultaneous two-way information exchange and flexibility of communication formats. These should network all the project participants despite their geographic dispersion and time limitations. Increasingly large investments are being made by major construction firms in IT and engineering technology. The use of IT gives the potential of closer cooperation between buyers and suppliers in the product design and development process, of sharing logistics information, such as demand and stock levels in order to develop more responsive supply capabilities in meeting customer requirements [Sau97]. The use of technology by various actors in a construction supply chain to share information is unprecedented. Widespread use of technology has enabled availability of information and resource sourcing on a global level [Edu01]. Acquiring information from websites has become vital for contractors as more and more procurement websites are available on the internet [Dze05]. Pietroforte [Pie97] further identifies that the focus of new information technologies should be broadened from controlling contractual compliance to facilitating communication and interaction among project participants.

“...communication with other organisations is an essential requirement for purchasing and supply management. Scope for adopting IT approaches in this external role is increasing through techniques that have come to be known as ‘electronic data interchange’ – with EDI as the accepted acronym.” - Saunders [Sau97].

3.5.3 Technology Accelerates the Rate of Information Flow

An important essence of supply chain management is to employ IT/IS to accelerate the information flowing in both intra and inter organisations [Han99]. Fast flow of information between processes in a supply chain can provide rapid or real time information on market conditions, availability of materials from certain suppliers and reduces decision making time. Yeo and Ning [Yeo02] identify that information system management is one of the components of supply chain management which enables real time information sharing. An important component of the advanced level of supply chain systems is information links which bring together the critical competencies of supply chain partners [Yeo02]. These information links enable an accelerated pace of information flow.

3.5.4 Competitive and Strategic Advantage

Technological developments have also contributed to enhance the importance of information and make it a strategic resource. In modern business organisations, business planning is not only about market share and returns on investments but increasingly business organisations also take consideration of concepts such as competitive advantage and strategic positioning by relating them to their business goals [Bet99]. According to Betts [Bet99], competitive advantage can be gained by using techniques in the areas of human resource management; marketing; product design; services; distribution methods; research and development; use of advanced technology; and information and information (knowledge) management. Enabling the use of techniques in these areas requires the application of IT, which then enables an organisation to gain competitive advantage. Lysons [Lys00] identifies the benefits of using IT to gain competitive advantage in terms of reduction of costs when producing goods and service. IT enhances an organisation's capability by creating new linkages between the activities it performs both within and outside the company and also allows coordination of actions more closely with their buyers and suppliers to gain strategic advantages over competitors. Hence IT should be viewed as part of a long-term strategy rather being for short-term financial returns and therefore appropriate IT strategies are vital for rapid and effective communication of ideas, plans and data [Wal99].

3.5.5 Inter-Organisational and Organisational Efficiency

Use of IT enables inter-organisational efficiency by reducing costs, an enterprise incurs in dealing with external organisations, for example suppliers and subcontractors. Lysons [Lys00] has identified the following examples of a gain in inter-organisational efficiency due to IT:

- a) Allowing purchasers to “shop” and check the status of potential suppliers electronically.
- b) Facilitating the inexpensive electronic transmission of purchase orders.
- c) Improving control and co-ordination of suppliers with vendors e.g. arranging deliveries in more economical lifts and at time required by purchaser.
- d) Monitoring supplier performance.

The concept of improved operational efficiency and functional effectiveness can be extended beyond the boundaries of a single firm via use of inter-organisational information systems [Bak86]. Inter-organisational systems provide opportunities for better coordination between customers and suppliers in order to make operations more efficient for the benefit of all participants. They allow firms to integrate information related activities without disturbing the legal boundaries of the entities involved [Bak86]. According to Bakos and Treacy [Bak86] one example is to couple the product planning system of a firm with the order entry system of suppliers to lower the amount of inventory in process and the turnaround time for new orders. Another area which is part of this research is linking an organisation’s systems with an external system which provides up-to-data information on product availability by accessing the information systems of several different suppliers in real time. In the construction industry, such efficiency gains can only be achieved by linking the information systems of various stakeholders to perform a set of related activities which requires information exchange between the parties involved. One way of linking different organisations to achieve inter-organisational efficiency is via project extranets. Project extranets provide access to information repositories set up by the organisations via the web. This allows project participants to view or update information. Links inside an organisation of the information systems of various departments and sections can be set up via the intranet,

which allow members of an organisation to access information available from different sources inside an organisation [Har01].

An interesting scenario is presented by Walker and Betts in their paper [Wal97] published in 1997, which identified how IT will be used in global construction in the year 2001 to improve organisational efficiency. The scenario identifies how video conferencing available via the World Wide Web will link project participants to conduct project meetings, which will save time and cost. Using the web, databases could be updated on the spot and various options affecting the time could be investigated using the project planning system. By setting up video links on-site construction problems could be investigated by project participants remotely. GUI based systems will assist in easy understanding of project control documents by graphically representing various aspects of the project, for example representing project milestones achieved against agreed time in the form of a bar chart. The web will enable monitoring of work activities via permanent point cameras to provide a quick review of the state of affairs without having to walk to the construction site. The web will also provide access to daily supervisor reports and automatically catalogue them and provide the facility to retrieve them when required by searching the information bank. As more and more companies use the internet, it will be easier to link various stake holders' web resources for effective communication and making valuable information available. This scenario is presented in greater detail in [Wal97].

3.5.6 Small Firms

Although technology has an important role to play in larger firms to manage complex organisational operations, it has an equally important role to play in smaller construction firms, who also are exposed to market forces, such as increasing competition from local and overseas firms, rising costs, enlarged markets and threats of being taken over by larger firms. In order to ensure their survival in this competitive environment, smaller firms have also made increasing use of IT. This not only enables them to remain competitive and profitable, but also take on projects which until the recent past were run and managed by their larger counterparts [Edu99]. Tools and techniques available through IT infrastructure are enabling them to compete directly with the larger firms. Edum-Fotwe et al. [Edu99] have further identified that increasing

3. Construction Procurement

availability of IT at lower costs and the low operating costs of the smaller firms due to their smaller size is changing the nature of competition in the industry in a way which is yet to be understood. Larger firms are also increasingly contracting out their non-core operations to small firms by forming strategic alliances [Edu99].

The construction industry in the UK is highly fragmented [Ega98]. In the presence of a large number of construction firms in the UK, most of which are small and medium scale enterprises, integration of teams specialising in different areas from different organisations is very crucial for efficient project implementation. An important strategic target set in the Strategic Forum for Construction report [Sfc02] is that 20% of construction projects by value should be undertaken by integrated teams and supply chains by the end of 2004. This figure should rise to 50% by the end of 2007. To create such integrated teams for the required work the role of an integrated IT approach is very crucial [Sfc02].

3.5.7 Web Technologies

Computer-based technologies such as the internet and web technologies enable linking of business information to a global network and provide a common standard for transmitting and displaying information in a cost effective way [Kon04]. There is potential to “*drastically reduce*” the cost of infrastructure by adopting and making a wider use of technological options such as internet and e-Procurement [Sfc02]. Web technologies can also be used for effective communication. For example, web-based email systems provide a reliable and efficient mechanism for sending messages to different members of the team very quickly. The web also provides access to resources such as electronic libraries and search tools which can be effectively used for finding information. By using web technologies such as blogs, wikis, podcasts or by using website creating tools, information can be easily and quickly published on the web and can then be accessed from almost anywhere in the world. Walker and Rowlinson [Wal99a] identify that construction professionals can use the web to gather estimating and forecasting information. The web can also be used for undertaking research activities by accessing various resources available online. For example, links can be established with supplier data files for identifying cost information, to bureau of statistics data for marketing plans, or to the bureau of meteorology for finding weather

information [Wal99a]. Construction professionals can also use the web for promotional purposes to make potential clients aware of the company's profile, capabilities and services, which are offered in the same way as currently marketing and promotional materials are offered.

3.6 How MDSSF Can Benefit Construction Procurement

The importance of advancing the use of technology to achieve greater benefits is well identified in the literature. This research proposes a data sharing architecture for managing product data, which allows buyers and contractors to access product data made available by several different suppliers in an integrated way. The aim is to aid practitioners, when they make procurement related decisions and to provide product information. However, procurement is a very wide subject area, and therefore proposing a data sharing architecture is simply not sufficient. As identified in Section 3.3 procurement is not just about meeting on-site construction requirements by sourcing equipment, raw materials, products and services from suppliers. In order to carry out procurement related activities effectively, organisations also consider many other factors having wider influences, scope and implications on themselves and on the construction industry as a whole. Organisations spend considerable resources and time, and form different types of collaborations and partnerships, not only to ensure timely delivery of project outputs, but also to make further improvements, for example by introducing innovative techniques, by making changes based on lessons learned from past projects, by seeking greater specialisation to reduce costs and outsourcing non-core functions. Therefore it is important for this research to look into several different factors, which influence procurement from the perspective of identifying, how the MDSSF data sharing architecture can be of benefit to organisations in different organisational scenarios. In this section the author identifies some of the important factors and scenarios, which influence procurement and how the MDSSF can be of benefit in those scenarios by providing an integrated access to product data from several autonomous supplier databases, which ensures the authenticity of the information source.

3.6.1 Organisations Depend on External Companies for Procurement

Yeo and Ning [Yeo02] identify that the function of procurement is highly dependent on external companies such as suppliers and sub-contractors. In this respect, it is important for a contractor to have access to product information held by suppliers and sub-contractors in order to select the appropriate materials. The construction industry is fragmented and there is a diverse supply market in which clients can perform procurement [Cox02]. Cox and Ireland [Cox02] further identify that technological advances and advances in the field of construction products and services, now provide a wide range of different sourcing options. Due to these advances, the construction firms now use the expertise of external suppliers to provide construction related services, previously managed by firms internally.

Having access to product information from a large number of suppliers not only enables selection of appropriate materials from a vendor to meet the constraints of the project, but it also accelerates the decision making process. The MDSSF information sharing architecture can be used to provide practitioners with external information about potential suppliers, and products they can offer in an integrated way. An important objective in providing information in an integrated way is to achieve cost reduction and selection of the best available supplier, by taking into consideration the project constraints. Because the UK construction industry operates on a low margin [Cox02], there is increasing pressure on the industry as a whole to reduce costs. Procurement can be made for high value and low value items. Cox and Ireland [Cox02] identify that purchasing large value items regularly from a supplier to whom an organisation is a key customer can aid cost saving. However purchasing low value items from a “*powerful supplier*” can sometimes result in a waste of time and effort. In such circumstances finding an appropriate supplier with fewer overheads, such as administration costs and transportation can help in cost savings. In order to identify such suppliers comprehensive information is required which is collected from different sources and presented to decision makers. The MDSSF architecture has the potential to provide such information in a convenient fashion.

3.6.2 Communication is Important for Procurement

In procurement, there is a greater degree of communication and negotiation with suppliers and sub-contractors [Yeo02]. The ability of a supplier to meet given requirements also leads to a higher degree of communication, as the contractor then negotiates with the supplier regarding details such as pricing, product specifications, bulk discount, delivery and payment method. We believe that an online system built using the MDSSF architecture can play a significant role in this negotiation process by incorporating negotiation options alongside product information.

3.6.3 Suppliers in Different Supply Chains

Since there are different types of supply chains in the construction industry such as labour, material, equipment and professional services, procurement professionals face challenges and difficulties in sourcing from these supply chains [Cox02]. One of the challenges is the availability of large amounts of scattered information, available from diverse sources, such as individual suppliers in these supply chains. This makes it increasingly difficult for procurement professionals to keep themselves abreast of the available information. In this respect, a system built using the MDSSF architecture can provide a mechanism to present up-to-date information in an integrated way to aid procurement professionals with their selection process. The MDSSF architecture has the potential to serve as a channel providing effective information to construction firms, which allows them to choose their suppliers. It can aid in finding the right suppliers by providing information about them and the products and services they supply to the construction industry. This will allow contractors or sub-contractors to judge the competency of suppliers by their ability to supply the required products as per the project requirements.

3.6.4 Full Knowledge of Equipment Specification

A buyer or contractor may require full knowledge of equipment specification before making a purchase. It is necessary to know about equipment cost, quality and delivery time, but also to know about the detailed technical specifications of equipment, which a contractor may wish to acquire. For example capital equipment are costly and takes a long lead time to manufacture and procure [Yeo02]. Technical specifications of capital equipment may interrelate with the technical specifications of other equipment sourced

3. Construction Procurement

from different suppliers [Yeo02]. Information on technical specifications can help in planning, provisioning, and checking compatibility, if they are sourced from more than one supplier. By using an online system, the right kind of equipment can be identified and correlated with the technical specifications of other equipment. Applications built using the MDSSF data sharing architecture can provide suppliers with options for specifying product specification in detail, which can help practitioners to take into account such information, when making procurement decisions.

“It is also important that firms are fully aware of the products and services that they purchase. Research by the authors in industries ranging from construction and financial services has demonstrated that practitioners are not always in possession of the information that will allow them to act in a professional and effective manner.” – Cox and Ireland [Cox02].

3.6.5 Each Construction Project is Unique

Cox and Ireland [Cox02] identify that there is no generic answers to problems and situations in the construction industry. Each engineering project is unique and therefore procurement planning is also unique for each project [Yeo02]. Also the circumstances under which the construction firms operate keep changing. Social, economic and political changes all have an impact on the construction business and the construction practitioners have to take account of these changes in order to remain competitive. They also have to take into consideration market conditions, which change and evolve swiftly, and adopt different strategies based on the changing environment [Cox02]. Not only is every project and its environment unique, but different actors in the supply chain may need to be brought together to create *“each individual solution”* [Cox02]. Therefore a single business strategy or a single solution cannot be used in different scenarios. Industry practitioners need information from various sources to help them devise new strategies to meet these changes. The requirements for each project may differ depending upon the site, client’s demands, and how the constructed facility will be used in the future. Taking into consideration project requirements, different suppliers may be chosen, who have the ability to supply the right kind of materials for the project. In this respect having access to a resource such as the MDSSF, which provides

comprehensive information about different suppliers and the products they can supply is very crucial for the contractor and the project.

3.6.6 Data Sharing in e-Business Models

Cheng et al. [Che01] propose an e-business model for improving communication and coordination, and encouraging mutual sharing of inter-organisational resources and competencies in the construction supply chain. In the e-business model, the importance of communication and coordination is emphasised and therefore in this respect the role of IT is pivotal. The e-business model proposed by Cheng et al. enables “*improved flow of information and other resources in terms of speed, quantity and the level of confidentiality across the boundaries of all parties, and ensures a cooperative relationship between construction parties*”. The e-business model provides a comprehensive infrastructure and considers various aspects in the construction industry. The components of the e-business model are described by Cheng et al. as:

- 1. A cooperative virtual network structure that is associated with a hierarchical contracting structure, resulting in a value added construction supply chain infrastructure.*
- 2. A supply chain infrastructure. This structure consists of six core functional elements – resource planning, teamwork, tools and techniques, information management, training and development, and performance measures.*
- 3. Change management (which will require the implementation of human, organisational, and cultural enablers) so that the organisations’ employees can adapt to the e-business environment.*
- 4. Organisational adaptation, which is the ability of management and employees to learn and respond to changes in the external environment.*

The author believes that the MDSSF data sharing architecture can aid such e-business models by providing the actors in the core functional elements identified, by Cheng et al., with supplier related information. The prototype MDSSF application uses open communication standards for data communication, using XML based Web Services for interoperability and therefore can become a part of a virtual cooperative network of

organisations seeking information from external suppliers when making crucial decisions.

3.6.7 Partnerships and Collaborative Working

In a partnering relationship, two or more organisations work together to improve performance with the aim of achieving mutual objectives and in this process devise plans for dispute resolution, continuous improvement, progress measurement and sharing the gains [Ega98]. The need for forming partnering relationship collaborations between organisations undertaking large construction projects is strongly emphasised in the literature. Construction projects bring together architects, designers, project managers, surveyors, contractors, builders and others to form partnering relationships. Kornelius and Wamelink [Kor98] identify that co-operation and inter-organisational coordination is a common practice in the construction industry. High levels of performance can be achieved, when organisations bring together their resources and use networking techniques [Pat99]. Lamming [Lam93] identifies that individual companies cannot face challenges, when working globally without forming strong collaborations. Edum-Fotwe et al. [Edu99] also call for a closer partnership of organisations, who are involved in the construction supply chain beyond existing practice by taking advantage of IT to face up to the increasing challenges in the construction business at the global level. This is because such collaborations lead to effective utilisation of resources and eliminate duplication of efforts [Lam93]. Collaboration between buyers and suppliers is an important requirement when creating an agile supply chain, which is responsive to market demands [Cox02]. The importance of partnering in construction has also been strongly recognised in the Egan [Ega98] and Latham [Lat94] reports. From the above it is clear that a strong partnering relationship is very important for success in construction projects. The MDSSF architecture can aid the process of building partnership relationships for procurement by providing information on available suppliers in the market or region. Once partnership relationships are established the MDSSF architecture can facilitate sharing of product data between project partners.

Edum-Fotwe et al. [Edu99] further identify that because of the increasing globalisation and emerging enlarged markets and the resulting competitive forces which these trends

create, there is a greater need for collaboration and co-operation in the construction industry to ensure a firm's continued survival. A new approach to procurement, and a greater degree of coordination is also required between organisations and the alliances they form via subcontracting arrangements, if the project supply chain is to be managed effectively [Edu99]. This collaboration can be achieved by the integration of processes in the construction industry by making effective use of IT [Cox02]. A key technology identified by Walker and Rowlinson [Wal99a], which has potential to change procurement systems completely is the concept of virtual collaboration where designs may be posted and manipulated in cyber space by the collaborating groups of designers, consultants and contractors. The MDSSF architecture has the potential to create virtual collaborations and process integration between contractors and suppliers, which will allow contractors to retrieve product information from supplier databases when desired. Interfaces provided by the MDSSF application can be hooked to the information systems of contractors using different IT tools available for different platforms. In this way, an integrated system can be created to access product data from supplier databases.

3.6.8 Relationship Management

In the construction business, forming alliances with different organisations to undertake a construction project is crucial. Therefore it is important for organisations, if they wish to succeed globally in an increasingly competitive environment to efficiently manage their relationship [Edu99] both at the management and operations level. In this respect information sharing, for example between contractors and suppliers by using information sharing architecture such as the MDSSF, plays important roles by bridging the information gap, thereby allowing the participating organisation to fully learn about the organisations with whom they wish to form alliances. In order to remain competitive companies are constantly faced with challenges to reduce time-to-market, improve product quality and slash production costs and lead times [Kum00]. Kumaraswamy et al. [Kum00] further identify that these challenges cannot be effectively met by merely changes within specific organisations or organisational units as they heavily depend on relationships and interdependencies between the different parts, that are internal or external to an organisation.

3.6.9 Small Firms

The role of small firms in the construction industry cannot be ignored. The profitability of larger firms is linked with the efficient performance of the smaller firms and therefore it is important that both these types of firm work together in the construction supply chain in a collaborative and cooperative fashion [Edu99]. According to Small Firms in Britain 1995 report [Dti95], more than 10 million people in the UK are employed in small firms. 60% of construction jobs are in organisations having less than 10 employees, and there are 1.2 million more small firms than in 1979. According to the Egan report [Ega98], published in 1998, there are 163000 construction companies listed on the Department of Environment, Transport and the Regions' (DETR) statistical register, and most of them employ fewer than eight people. The Institute of Employment Studies, in their survey of Small to Medium Enterprises (SMEs) of 2005 [Ies05] have identified that 10% of SMEs in the UK (in the survey sample of 8640) are in the construction industry.

The construction industry itself has made radical shifts in its strategies from 1990 onwards, in response to competition both at the regional and global level. The strategies before the 1990s, although growth oriented, also had features of rigidity which meant it took a long time for organisations to respond to the changing business environment [Edu99]. In order to take advantage of large-scale economies, organisations adopted strategies of vertical and horizontal integration [Edu99], which is an important cause of rigidity both at the organisational level and at the level of its many systems. However the construction industry has responded to increasing competition and the dawn of the information age by working towards achieving greater flexibility. In order to achieve flexibility firms have adopted vertical and horizontal disintegration and outsourced activities which do not form a core part of the business to external suppliers [Edu99], which has given rise to an increasing number of small and medium firms in the construction business. The larger firms now work with smaller or medium counterparts in construction projects where the work is sub-divided into packages or sub-packages and sub-contracted to individual organisations based on their expertise. Because of this disintegration, smaller or medium scale firms who supply products and services now play equally important roles. The construction industry presently has a large number of smaller organisations which provide services to larger counterparts via sub-contracting

arrangements [Edu99]. The presence of smaller firms in the construction industry makes the role of data sharing architectures all the more important, as it provides an opportunity to smaller firms to make themselves known to their larger counterparts by participating in data sharing federations such as the MDSSF. Smaller firms, because of their small budgets cannot often make large investment on IT systems, which prevents them gaining competitive advantage against their competitors. As we have seen, the use of IT is vital in the increasingly information driven business environments of modern construction. The MDSSF application can provide small and micro scale suppliers with information systems to manage their product data using a subscription based approach (see chapter 6). Hence the MDSSF application can serve two vital purposes for small and micro firms: manage product data, and share it with potential buyers and contractors.

3.6.10 Supply Chain Operations and Logistics

One of the important roles of an organisation's information system is to support the smooth and effective functioning of supply chain operations and logistics. According to a report by the Strategic Forum for Construction, much of the waste in the UK construction industry occurs due to poor logistics [Sfc02]. This report calls for a greater emphasis on improving the supply chain operations and logistics by facilitating integrated working. Porter and Millar [Por98] identify that the effective performance of an organisation is related to the effective management of its supply chain operations. Yeo and Ning [Yeo02] identify that real time information sharing, coordinated procurement processing in the whole chain and collaborative attitude among all chain members are important drivers of supply chain management. Since many organisations come together in a construction project, the efforts of all these organisations need to be coordinated to ensure effective collaboration. Within a given project the responsibility of procurement may lie with many project participants working in their specialised areas, as part of different supply chain operations, such as inventory management, sales and purchases. Therefore managing the flow of information across a supply chain and ensuring availability of up-to-date information on project status and supplier data to these different participants is vital. In this respect an application designed using the MDSSF data sharing architecture can provide up-to-date product data in real time from a large number of supplier databases for different project participants working in the

different areas of supply chain operations. The MDSSF architecture can also provide information on product availability and delivery time to improve supply logistics.

3.6.11 Changing Environments and Globalisation

Edum-Fotwe et al. [Edu99] have identified that the changes in the construction industry that have taken place from 1990s onwards are the result of the “*strategic factors*”, which have come together and affected the construction industry. The factors include changes in the clients and their requirements. Construction industry clients now have high expectations and demand more in return for their money. There has been a widespread increase and improvement in construction productivity with the application of concepts such as lean construction, benchmarking and total quality management [Edu99]. There have also been changes in the business of construction. It is no longer a process of transforming raw materials into a finished product. It has adopted a more service oriented approach, where some contractors only facilitate the process [Edu99], whereas the actual work is done by specialised groups to achieve cost savings by selecting the best resources available from a large number of available suppliers. Additionally, globalisation has also had a considerable impact on the industry as a whole. The contractors not only procure from domestic markets but also turn their attention to international markets for sourcing expertise and suppliers. The contractors also increasingly bid for international projects as part of a larger consortia and form partnerships with local (to the project) organisations to take advantage of local knowledge and keep the costs associated with labour transfer to a minimum [Edu99].

An important change brought about by globalisation and an enlarged market is that smaller firms are now exposed to a greater degree of competition [Edu99]. Previously they only had to compete with local firms, but now they are increasingly threatened by overseas suppliers who are in a position to supply similar or better quality products on very competitive terms by taking advantage of low labour costs and abundant natural resources. These firms also have to face up to increasing client demands [Edu99] to deliver better quality products and services at very competitive prices. In these circumstances it becomes very important for the smaller firms not only to reduce costs to remain profitable but also to find new markets for their products and services in order to increase their market share.

3. Construction Procurement

Developments in technology have also been an important factor in bringing change to the global construction business. It put pressure on the structure and processes of the construction industry to change and has helped in the development of a global market [Edu99]. Many firms whether large or small operating in different parts of the world can be quickly reached using web technologies and search tools and communication can be established by exchanging email messages. Use of IT has brought in virtual proximity and given rise to electronic commerce and tele-working in the construction industry [Edu99]. Edum-fotwe et al. [Edu99] have also identified that the impact of IT on procurement is expected to be considerable. This is because IT can facilitate electronic-bidding and negotiation for projects, allowing suppliers from all over the world to take part in the bidding process, and collaborative teams can be assembled which use the best resources available worldwide. Hence, in these changing environments and with increasing globalisation, the role of a data sharing architecture, such as the MDSSF, becomes more important. For example, before bidding for an international project, a consortium might need information on available local suppliers with whom they can form an alliance to enable calculation of cost estimates. This will not only allow them to submit a competitive bid proposal, but to assess the current market conditions and identify potential challenges, problems, and opportunities in advance. Although the MDSSF application only serves a role of providing supplier related information, such information can be used by organisations in several different ways to achieve their objectives.

3.6.12 Decision Making in Strategic, Tactical and Operational Levels

The integrated data sharing architecture proposed in this research will bring together product information from different suppliers. This is required to optimise supply chain performance decision making at strategic, tactical and operational levels in the construction industry [Kum00]. Kumaraswamy et al. [Kum00] further identify that at the tactical level the main emphasis is typically focussed on supplier evaluation, supplier selection and supplier negotiation, while at the operational level the tasks are more administrative in nature, pertaining to purchase requisitions, quotations and the issue of purchase orders. The strategic level is concerned with issues such as outsourcing, supplier base reduction, formation of collaborative relationships and supplier development [Kum00]. The aim of the data sharing architecture proposed in

this research is to aid construction industry practitioners in their decision making at all levels: strategic, tactical and operational, by providing information about the suppliers and the products they supply.

3.6.13 The World Wide Web and Associated Technologies

Walker and Betts [Wal97] identify that information technology is fundamentally changing the global construction business, and that the internet and World Wide Web (WWW) [WWW07] will be key to this change. The construction industry can use web technologies to gain strategic and competitive advantage [Wal99a]. The three ways, identified by Walker and Rowlinson [Wal99a], of achieving this are:

- a) Web and internet technologies can reduce communication transmission costs. This will bring productivity gains and reduce multiple-handling of information in the supply chain.
- b) Web technologies provide the ability to allow clients already using these technologies to communicate with the construction industry using common communication technologies.
- c) There is a possible gain in quality of service advantage, if the use of these technologies is well thought out. The good use of these technologies can offer online current-status information on projects.

The MDSSF application is fully compliant with prominent web technologies, such as Web Services [Gra02], and it uses industry standard communication formats, such as XML making it interoperable with other web-oriented and desktop-based applications available, across different platforms. In this respect, the MDSSF architecture has the potential to provide competitive advantage via the use of web technologies in the three areas identified by Walker and Rowlinson [Wal99a].

3.6.14 Marketing of Supplier Products

Organisations constantly seek new channels to market their products to potential buyers and contractors. Advertisement of products and services using media such as news papers, trade journals, magazines, product catalogues, television, radio, hoardings, banners, sponsorships and internet is a common practice. From a supplier's perspective,

3. Construction Procurement

MDSSF architecture provides a mechanism to share product information with potential buyers and contractors and it therefore gives suppliers an opportunity to market their products. By participating in the MDSSF federation, a supplier can disseminate product information, which is retrieved by the MDSSF application from the supplier's database when a search is made. Hence the MDSSF application has the potential to make buyers or contractors aware of the existence of suppliers whom they may contact.

3.7 Product Procurement

This section identifies different approaches for product procurement, which are used by construction industry practitioners or are proposed by researchers. The MDSSF information sharing approach cannot be compared directly with these procurement systems or the product procurement approaches presented in this section since it is only an information sharing architecture and not a complete solution for procurement. However it has the potential to be part of a procurement process, as implemented and demonstrated in the COVITE application. In this respect, MDSSF can aid existing procurement systems or approaches by providing product and supplier information. MDSSF can provide certain functionalities which can be used by buyers, contractors and suppliers in combination with existing procurement systems, but it cannot completely replace existing procurement systems. With this view, we present some of the important procurement approaches and systems and identify how the MDSSF data sharing architecture can fill some of the gaps in these existing approaches and systems.

3.7.1 The Traditional Paper Based System

The traditional method of procurement includes a paper based system for procuring materials, where paper based product catalogues provided by suppliers are searched for required materials and orders are placed using telephone and fax [Kon04]. It involves generation, copying and transfer of paper documents, such as material requisitions, purchase orders and quotations [Cal95].

Kong et al. [Kon04] have identified problems with traditional construction material procurement systems. Some of the problems identified by [Kon04] can be summarised, as:

- a) Paper based systems are costly, error prone and time consuming.

- b) Such systems can only operate during specified business hours.
- c) In such a system a purchaser can only work with a limited number of suppliers.
- d) Use of such systems limit the amount of information that can be collected about suppliers and their products.
- e) Paper based catalogues are cumbersome to use, become outdated very quickly and require a large storage area.
- f) It difficult to search for products and make comparisons between different products using catalogues.

These limitations make it difficult for a contractor to stay up-to-date with market conditions and products.

3.7.2 Supply Chain and Critical Chain Project Management

Yeo and Ning [Yeo02] present an approach to product procurement management which couples the concepts of Supply Chain Management and Critical Chain Project Management and draw ideas from constraint theory. Constraint theory is used to understand the systems performance by identifying the constraints that limit output. The critical chain method, when applied with constraint theory, offers an enhanced approach, which manages risks and uncertainty associated with the project in the project value chain. This enables improved performance in project time management [Yeo02]. The framework for improved engineering procurement also highlights the importance of the partnering relationship in selecting reliable materials and equipment suppliers. It also proposes a mechanism of *feeding buffers* (provisioning of extra time in case of delay in procurement) to deal with procurement uncertainties caused by logistic processes and risks associated with procurement of major items. In the framework proposed, *feeding buffers* are added to the procurement process for major materials and equipment procurement parallel chains. In their approach, Yeo and Ning aim to improve present practices in Engineer-Procure-Construct (EPC) projects. They identify the importance of partnering relationships for selection of reliable materials and equipment suppliers. In order to create new partnering relationships, it is vital to have information on available players. The paper does not address how new organisations can be found, based on products they can supply. The MDSSF architecture can fill this gap by providing information about available suppliers and products they supply. This can aid

the procurement approach of Yeo and Ning, by adding techniques to identify new partners.

3.7.3 An Approach of Smarter Selection for Procurement

Kumaraswamy et al. [Kum00] and Kumaraswamy and Dissanayaka [Kum01] identify the importance of an improved selection process for procurement of construction products for buildings or civil engineering infrastructure. We describe their work here to give the reader an overview of their procurement system. They conducted a Hong Kong based study, which confirmed the importance of 'smarter' project formulation in deciding factors, such as project scope and work packaging. The importance of more informed decision making in assembling appropriate procurement frameworks, which suit the needs of a particular project is discussed [Kum01]. They identify that intelligent choices in overall procurement system design should be related to the project scenario rather than being based on familiarity or convenience. In the study, five principal sub-systems of the procurement system were identified. They are: work packaging, functional grouping, payment modalities, contract conditions and selection methodologies. They identify that selection of these procurement subsystems are critical procurement aspects that should be tailored to match the current project objectives. Additionally, holistic approaches are also required in construction procurement, in order to achieve a synergistic balance between the procurement subsystems. The holistic model should extend from critical work packaging decisions at the outset, through appropriate risk, and functional allocations to selection methodologies adopted in choosing various procurement sub-systems [Kum00]. It should enable decision makers to make "*smarter*" choices (for example by choosing sub-systems and assembling appropriate project specific procurement systems). Therefore, in order to select appropriate procurement and operational systems and make optimal selection of project participants (such as joint venture partners, consultants and contractors), there is a need to make "*smarter*" choices and not use a purely price based selection. This study identified 11 key performance criteria against which a construction project performance can be assessed (see Appendix 1). They also identified that significant improvements in procurement processes in addition to improvements to managerial processes are necessary to achieve greater performance gains.

3. Construction Procurement

There is an underlying demand to access and analyse large amounts of industry data, past project information and expert knowledge in order to make the choice of an appropriate procurement system. Kumaraswamy et al. identify that rapid growth in IT/IS potential will make it possible to do this in the future. They identify that knowledge based decision tools are required to meet these demands in order to improve the selection processes in the construction industry supply chain which will lead to gains in productivity in the construction industry.

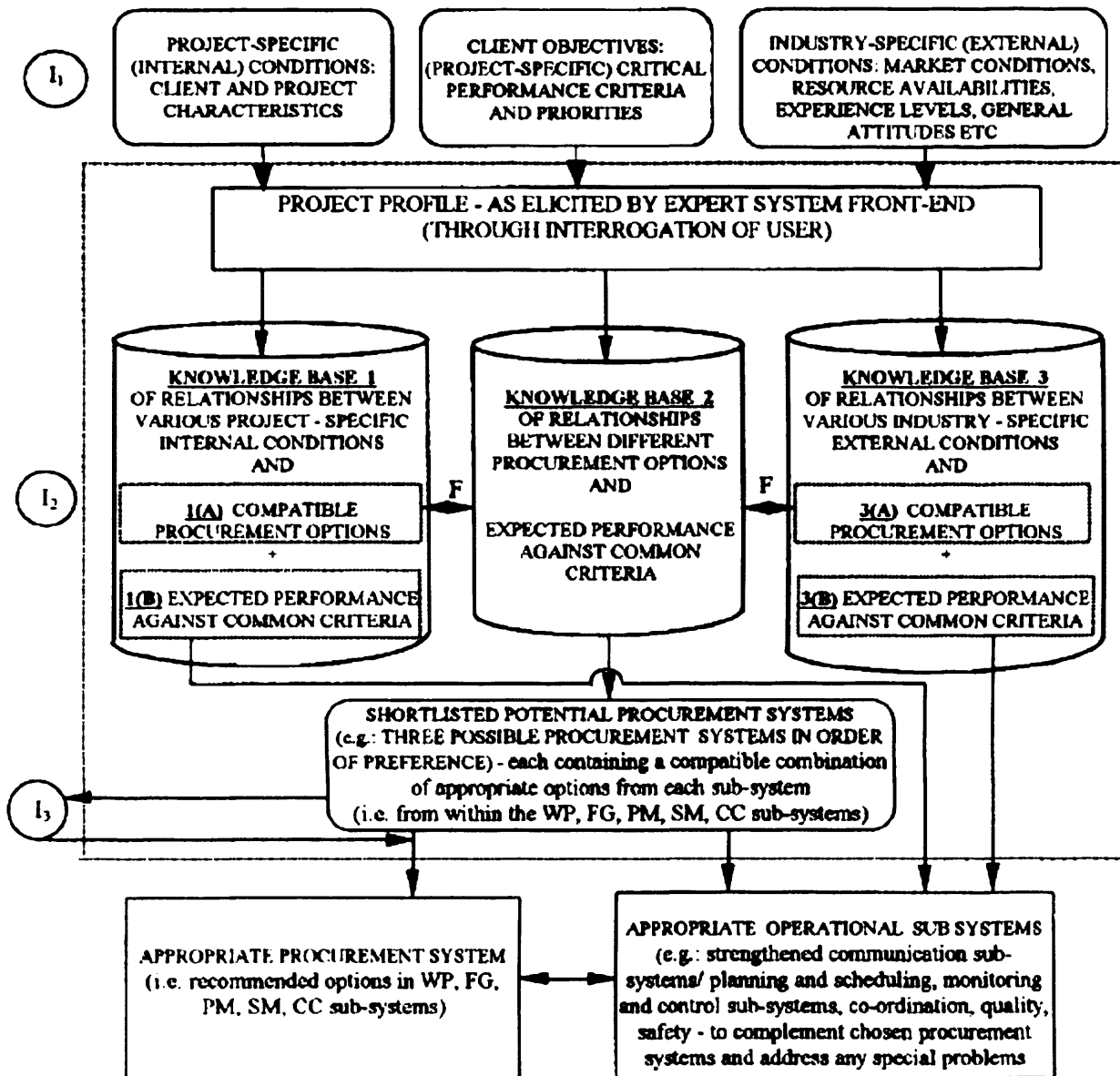
Kumaraswamy et al. [Kum00] propose a knowledge based decision support system that will help its users make more informed procurement decisions, and select appropriate supply chain structures and people, whilst simultaneously suggesting synergistic operational processes that would meet client priorities, project-specific (internal) conditions and industry-specific (external) conditions. The system will enable capturing and consolidating experiences from previous projects and will also provide the facility of harnessing decision rules used by experts having a broad overview of procurement systems. IT/IS and artificial neural network aids were used to capture and analyse available experiential knowledge, in order to make intelligent procurement and operational choices [Kum00]. These aids were needed to develop more structured approaches which use comprehensive databases and knowledge bases. The aim of the system is to provide “*advice*” pertaining to procurement matters, which takes into account project specific scenarios. The proposed model of this decision support system is illustrated in Figure 3.3. It consists of three knowledge bases in its core. The knowledge bases are designed to capture and incorporate any “*causal*” relationship which links project performance (which is measured using the 11 key performance criteria) against variables such as: project specific internal conditions, procurement options and external conditions.

The knowledge based system identified by Kumaraswamy et al. for making procurement decisions takes into account various aspects of a construction industry supply chain as identified in the figure to select an appropriate procurement system for a given project. An important distinction can be identified between the decision support system and the MDSSF architecture in the area of information sharing. The MDSSF provides an information sharing mechanism only and therefore does not take into

3. Construction Procurement

account the wider perspectives achieved by incorporating the several areas associated with procurement identified by [Kum00] and [Kum01]. The MDSSF architecture does not take into account all these construction industry aspects, because they are not relevant when providing product specific information to contractors from a large number of product suppliers. However we believe that the software systems developed using MDSSF can aid such knowledge based decision support systems by providing information on industry specific (external) conditions such as resource availability from a wide range of product suppliers to decision makers enabling better choices to be made.

3. Construction Procurement



ABBREVIATIONS

CC - CONTRACT CONDITIONS FG - FUNCTIONAL GROUPING
 PM - PAYMENT MODALITIES SM - SELECTION METHODOLOGIES
 WP - WORK PACKAGING

Key:

- I - USER INPUTS
- I₁ - MODELLING PRESENT PROJECT PARAMETERS
- I₂ - CHECKING WHETHER ANY PROJECT PARAMETERS (INTERNAL OR EXTERNAL CONDITIONS) COULD CRITICALLY AFFECT THE EFFECTIVENESS OF CERTAIN PROCUREMENT OPTIONS (These are temporary input requirements at the 'F' interfaces, which will be later replaced by additional knowledge base modules)
- I₃ - RE-CHECKING THE COMPATIBILITIES AND ANY POSSIBLE SIDE-EFFECTS OF THE SHORTLISTED PROCUREMENT OPTIONS
- - EXPERT SYSTEM BOUNDARIES
- F - FOR FUTURE KNOWLEDGE BASE DEVELOPMENT

Figure 3.3 Proposed model of decision support system for optimising procurement protocols and complementary operational sub-systems as identified by Kumaraswamy et al. [Kum00]

Source: Kumaraswamy et al. [Kum00]

3.7.4 Procurement using E-Commerce Systems

The E-commerce systems that have emerged in recent years for construction material procurement have become e-trading market places, through which manufacturers, suppliers, agents and purchasers can buy and sell construction materials [Kon04]. E-commerce technologies enable electronic trading of goods and services, online delivery of digital content, electronic fund transfer between bank accounts, invoicing, generation of receipts for goods bought or sold, confirmation of payments, online sourcing, and procurement [Har01].

These E-commerce systems are owned and managed by individual manufacturers, suppliers, agent companies and application service providers [Kon04]. However the construction material information in these individual information systems is limited and the information systems are isolated with no interaction between them [Kon04].

Using an E-commerce system for construction materials, different kinds of information can be shared such as material information, supplier information, manufacturer information, buyer information, agent information, information on the amount of sales for different materials, buying patterns, buyer's comments on products and services [Kon04]. Many organisations use e-Commerce systems when doing business [Kon04]. E-Commerce systems provide an environment for direct communication between buyers and suppliers and an expanded market place for doing business [Che01]. Online construction material trading markets are readily accessible to buyers and are not bound by geographical limitations. A buyer can quickly locate a desired product using a key word search or by browsing through the categories and sub-categories. E-Commerce usually provides a larger variety of products with different ranges, sizes and style and does not suffer limitations such as limited store space [Kon04]. Direct communication between a buyer and supplier eliminates middlemen and reduces procurement costs [Kon04]. Organisations such as *Home Depot* and *Lowe's* in the US construction industry utilise E-Commerce in their retail business. They are also developing B2B solutions so that information between customers, stores, suppliers and employees can be shared [Kon04].

3.7.4.1 Limitations of E-commerce

IT/IS is playing an important role in linking together different systems and sub-systems and enabling the flow of information to effectively manage the procurement process. However this network of information systems does not provide enough support for enabling new business models in which suppliers have more control of their data. These information systems also do not provide tools for managing information pertaining to product or material specification effectively. An important requirement of new or emerging business models is to enable virtual collaborations, where data models of the virtual environment can be shared between the participants. The existing technological options used in the construction industry enable linking of information systems to share data but the option of sharing data models by creating virtual environments is not widely used. According to Kong et al. [Kon04] E-commerce systems for construction material procurement serve two crucial functions: supplying construction material information and facilitation of transactions and trade. They further identify that some E-commerce systems can be owned and operated by individual suppliers, manufacturers or agents and a buyer usually has to visit more than one E-commerce system to procure all the necessary construction materials.

Different E-commerce systems usually adopt individual methods of publishing and managing information about construction materials. The web sites adopt different methods for searching and displaying information and use different attributes for construction materials [Kon02]. There is heterogeneity in the way a similar type of information is managed by different suppliers. This also creates a problem for buyers as they have to remember and maintain a list of web addresses and understand the semantics and navigation methods of each web site to find the right kind of material [Kon04]. This is a time consuming process for the buyers. From a seller's point of view, the E-commerce system they are using is closed, which means they cannot provide information from other E-commerce systems and therefore comprehensive market information for making key decisions on production and distribution cannot be made using a single system [Kon04].

Web based e-markets provide contractors with more business opportunities and a greater selection of suppliers, however, it has also created a challenge for contractors as

3. Construction Procurement

they have to manage a large amount of electronic information [Dze05]. Construction companies provide information on a large number of products via a single web site. But a contractor may not find all the required products in a single web site. For procuring all the materials required, a contractor may have to visit a number of web sites. Since product information is supplied individually by the construction companies on their web sites, it is difficult for a contractor to make a comparison of different products and choose the best product to meet project constraints.

“There are many limitations with the current use of IT in construction. The IT tools that are used are standalone systems, many of which were originally designed for the engineering design/production process.” – Betts and Clark [Bet99a].

MDSSF can address these issues, since it enables sharing of product data using a subscription based approach, which allows sharing of data models with the suppliers so that they can create product representations in their systems using a single, agreed format for data sharing. This addresses the issue of heterogeneity. By using a single data representation for data sharing a large amount of product information from different suppliers can be integrated and presented to the buyer or contractor in a coherent fashion. In this way all the information from several suppliers can be presented via a single system so that the contractor does not have to visit several different systems to find product information. The structure of MDSSF’s data model is described in Section 6.2.

3.7.5 The E-Union Framework

E-Union framework [Kon04] provides value-added services by enabling information sharing between E-commerce systems for construction material procurement. In this framework, for the purpose of inter-communication and information interchange, a number of different construction material trading sites are joined together by an application provided in their information system. The framework enables a buyer to access material information at other E-commerce sites by using one of the E-commerce websites [Kon04]. The framework links together all the participating E-commerce systems using an E-Union server which acts as a data centre for collecting information from its members and passing it to requesting members [Kon04]. The presence of

frameworks such as the E-Union demonstrates that there is a need for such data sharing systems in the construction industry. There are similarities in the challenges the E-Union and MDSSF frameworks are trying to address. Both systems address the issue of information sharing within the construction industry. Important features of the E-Union framework are explained in greater detail in chapter 4 where the two architectures are compared and contrasted in greater detail through their architectural features.

3.7.6 Keyword Searching for Products

Dzeng and Chang [Dze05] propose a learning model which provides a mechanism to improve search effectiveness, when searching for construction products using keywords on websites. This mechanism guides a user's search using three approaches: correction, specification and extension. The correction guide corrects misspelled or misused words. The specification guide constrains the search by adding words such as "AND" to keyword phrases or by replacing a keyword with a more specific term. The extension guide extends the search criteria by adding keywords such as "OR" or by suggesting keywords for subsequent searches. [Dze05]

3.8 Chapter Conclusions

This chapter identified the role information and technology plays in the construction supply chain to enable its various processes. We looked at the importance of procurement in the construction industry and how information systems can be used to improve procurement planning. Approaches of information sharing and product procurement, ranging from traditional paper based systems to modern systems which use computer-based technologies to provide users with quick access to information and perform procurement related activities were also identified. Based on the arguments presented in this chapter, the MDSSF data sharing architecture has the potential of providing benefits to construction industry practitioners by presenting up-to-date supplier data in an integrated way and in real time. In this the MDSSF can also act as an important support tool for existing systems and can be integrated with existing procurement systems. The MDSSF should not be seen as replacing existing procurement systems. With this objective, we believe that the scope of information sharing for product procurement can be improved by bringing about collaboration of the main actors involved in the process of product procurement via the MDSSF data sharing

3. Construction Procurement

architecture. The next chapter presents a review of information sharing models which provide information to users in an integrated way from different types of information sources which use different approaches.

4. Information Sharing in Distributed Environments

4.1 Introduction

There is an increasing requirement to make information more readily available for sharing from various sources via a single system using integration techniques in a structured way so that it is readily usable. The traditional approach is a federated database system with mediator based information integration systems or architectures, which provide mechanisms for information sharing but have a limited scope since the users need to have prior knowledge of the information sources they wish to access. The data sources which are integrated or federated usually have interoperability issues due to various types of heterogeneity. Alternatively the dynamic and open environments of the World Wide Web (WWW or simply web) provide access to a large number of heterogeneous information sources in different formats. Keyword based search mechanisms provided by internet search engines allow users to locate required information. However it is still a time consuming task even with the aid of search engines to retrieve related information from a given domain. We need a domain focussed search technique which enables its users to access relevant information from the available data providers and integrate the resulting data. This technique should take advantage of the traditional federated and other information integration approaches, the current information sharing approaches using the web (and its associated technologies and protocols), and provides information to user using an agreed format so that it can be readily analysed and used by users for decision making in the given domain. The application domain for which a solution is being designed has a vital impact on the choice of the federated architectures [Con99]. In this research the specific domain is provision of product information from a large number of product suppliers to contractors to aid procurement related decision making.

The MDSSF is a database-centric information sharing architecture which federates a large number of product supplier databases to present product information to contractors in an integrated way based on a common data model (CDM). In this chapter the author presents a review of information sharing models which provide information to their users in an

4. Information Sharing in Distributed Environments

integrated way from different types of information sources using different approaches. In order to highlight the novel features of MDSSF, 40 different information sharing systems covering different areas and several schema integration approaches or methodologies were reviewed. The features of the information sharing models researched are categorised into 14 different criteria and are presented in tabular format (see Appendix 3). This chapter reviews the key features of the information sharing and integration approaches to identify their scope and functionality in the light of the MDSSF's domain specific requirements. The information sharing models are analysed in the context of the author's present research only. The reader is encouraged to refer to the corresponding information sources (publications or websites) for further and fuller details.

The chapter is divided into a number of sections. Each section groups related information sharing models together and collectively identifies their strengths or key features from the perspective of the MDSSF information sharing model and also justifies why such features are applicable or not applicable to MDSSF based on the requirements it aims to address. We also identify key features in some models which strengthen this research. Section 4.2 provides a brief summary of the need for sharing information in autonomous environments. Section 4.3 describes the role of information mediators in integrating data from different sources. This section also identifies the role of resource wrappers, data model and query languages, ontologies and knowledge bases which address various information sharing issues. Section 4.4 provides a review of schema integration based approaches for information sharing and identifies some of the different schema integration methodologies described in the literature. Section 4.5 reviews federated database systems and federated information systems for information sharing. Section 4.6 identifies Grid based approaches to information sharing, and Section 4.7 reviews other systems not fitting into these categories of information sharing models. Finally in Section 4.8 conclusions are drawn.

4.2 Sharing of Information in Autonomous Environments

The past two decades have seen a phenomenal increase in availability of information from a variety of autonomous sources, because of its use in various information related activities in different industry sectors. The rise of the internet is also an important contributing factor

4. Information Sharing in Distributed Environments

which has led to the availability of information on a large scale. As identified in Chapter 3 the widespread use of information by different business organisations to gain competitive advantage has elevated the role of information from a mere enabler of process to a strategic resource – something which is indispensable to a modern organisation in an information economy. However availability of information from different sources also gives rise to a number of challenges. This increasing amount of information is available but in heterogeneous ways conforming to different formats and requires the support of different languages and/or tools for its access or manipulation. Different sources presenting information in different and isolated ways makes it difficult to analyse information using integrated and standard means.

An important challenge is how to view information available from different heterogeneous sources in an integrated way to derive benefits from it. This information integration challenge also brings other challenges with it such as how to resolve different types of structural, semantic and other forms of conflicts so that the information can be presented to the user in an integrated way when it comes from several different autonomous and heterogeneous information sources. Sharing of information in autonomous and distributed environments is a key element of this research. However, in this research an approach to information integration is not considered when providing an integrated view of information from several autonomous sources, due to the use of the common data model (CDM) by product suppliers. This provides product information in a standard way based on pre-defined criteria and therefore makes the MDSSF a federated information system for homogeneous databases. However understanding the information integration challenge is important to this research as it reveals the obstacles to information sharing and why some of these obstacles do not arise in the novel architecture of the MDSSF. Research efforts in this area have largely been focussed on establishing standard ways based on standard criteria, language and tools to enable interoperability in these heterogeneous environments.

4.3 Information Mediators

Mediators are software modules which enable “*intelligent and active*” use of information by presenting data from various sources in a useful way by using techniques such as

4. Information Sharing in Distributed Environments

abstractions, transformation and integration to aid decision making [Wie92]. They are used to integrate heterogeneous information sources [Pap95]. Wiederhold [Wie93] presents a three layer architecture for intelligent integration of information (I3) using mediators. The objective of the architecture is to assist end user applications “*with information obtained through selection, abstraction, fusion, caching, extrapolation, and pruning of data*” [Wie93]. A fundamental step to achieve intelligent integration of information is to address the issues of heterogeneity when bringing together information from several sources because “*joining heterogeneous data is essential when trying to generate information*” [Wie93]. Mediators together with wrappers address various types of heterogeneities including structural, semantic and other forms of conflicts to provide integrated access to heterogeneous information sources including those which are semi-structured and unstructured. Table 1 (see Appendix 3) describes various features of the 19 mediator and/or wrapper based systems reviewed as part of this research. Column 16 of Table 1 provides a short description of how these systems handle heterogeneity within their environments.

According to Tomasic et al. [Tom98] an important advantage of a mediator based architecture is that different specialised components handle different aspects of the system and user requirements independently [Tom98] by providing common standards. Additionally mediator based architectures also specialise in providing information from related data sources (for example, TSIMMIS [Gar97] and DISCO [Tom98]) and thus address the schematic and semantic issues of a given domain. They provide location transparency to information sources managed by different data providers [Pal03]. In order to access information from several sources mediators divide user queries into several sub queries – one for each information source and then integrate the query results from information sources before sending it back to the user. Mediators also process queries over integrated data representations [Tom98]. All these features lead to a flexible and extensible system.

4.3.1 Information Integration Using Mediators

From the perspective of this research understanding the concept of a mediator based architecture is important because this architecture provides a mechanism to access several

4. Information Sharing in Distributed Environments

autonomous information sources in an integrated way. The MDSSF approach also provides integrated access to several autonomous product supplier databases. However mediator based architectures also address certain issues which are not required for the MDSSF approach of information sharing. Mediators are used in information sharing environments, where intelligent integration of information is required because “*data is obtained from many diverse and heterogeneous resources*” [Wie93]. Therefore intelligent integration is the combination of heterogeneous data by resolving various types of conflicts and mismatch between data elements and presenting the results to the user in a readily usable form. For example, an important step in information integration could be fusion of objects. This process identifies semantically equivalent entities retrieved from different information sources for the purpose of semantic integration [Pap96a]. Section 4.4 discusses the problem of integrating information from heterogeneous sources (based on schema integration approaches) in detail and identifies some of the approaches described in the literature to address this issue. Out of 40 systems surveyed, 15 supported information integration or integrated access using mediators or provided “*mediator-like*” [Pal03] or mediated access to heterogeneous resources as identified in Table 1. Information mediators have been implemented in many different ways to suit different application domains and several different designs of information integration have been proposed. However the fundamental concept behind all these approaches is the same, namely integration of information or provision of integrated access to diverse sources and making heterogeneous systems interoperable in order to present information in a readily usable form. Table 1 identifies the key features of these information sharing approaches, their domain of operation, key implementation/middleware technologies, how they resolve different types of conflicts and other features of the mediator based information sharing approaches.

4.3.2 Resource Wrappers

In mediator based information systems, mediators accept a user query and decompose it into many sub-queries one for each information source [Gar99]. These sub-queries are then sent to individual information sources via their wrappers. The mediators access data through resource wrappers, which combine or integrate data retrieved through several wrappers and provide applications/users with an integrated view [Pap96]. Wrappers are

4. Information Sharing in Distributed Environments

commonly used to hide heterogeneity at the data source level and provide a standard interface. Wrappers provide a standard set of operations or a CDM to information from the underlying heterogeneous information sources so that they can be conveniently accessed in a uniform manner. Wrappers which are also known as translators [Pap96] also provide the functionality to convert data from different sources into the CDM of the system which corresponds to an agreed or desired format. One exception here is the resource wrappers of the BiodiversityWorld (BDW) [Pah06] [Pah06a] system which do not convert data into the CDM of the system but hide data heterogeneity by using specialised data types within its environment. The system provides tools at the client side to deal with heterogeneous data. Wrappers perform query and data transformation in order to deal with different query languages and individual data sources [Tom98]. In different systems wrappers are implemented or used in different ways. The TSIMMIS [Gar97] uses wrappers for converting user queries into data source specific queries. In order to address data source heterogeneity and describe the functionality of the underlying data sources the DISCO [Tom98] system provides a wrapper interface which supports an “*algebraic machine of logical operators*”. When implementing a new wrapper the DBI (Database Implementer) chooses a subset of algebra (capabilities of wrapper) to support. The DBI also registers the specifications of the capabilities of the wrapper such as the subset of the algebra it supports to aid mediators when queries are posed. The COIN [Bre97] framework also provides wrapper interfaces to information sources so that they can be accessed using a uniform protocol. Table 1 (Column 16) provides a short description of wrapper functions in several systems which use wrapper-based techniques.

4.3.3 The Data model and Query language

Almost all of the 40 systems or architectures reviewed and presented in Table 1, 2 and 4 (except the Grid based systems in Table 3) provide access to heterogeneous data sources via a CDM. The Grid based systems provide a standard mechanism for accessing heterogeneous data sources and not necessarily a CDM. For example the workflow based BDW [Pah06] [Pah06a] system uses a datatype approach to encapsulate heterogeneous data. The OGSA-DAI [Atk05] [Ogs07] framework provides standard *perform documents* operations to access different relational databases available in the Grid environment. On the

4. Information Sharing in Distributed Environments

other hand the XML [XML07] based systems reviewed (in Table 1) such as MIX [Bar99], XML Media [Gar99] and XML Data Integration System [Alm04] use XML as the CDM and XML based query languages. The mediator based systems use a mediator language and its associated data model to provide data from heterogeneous sources based on the given structure and semantics of the middleware layer. The CDM and the associated query language provide a standard mechanism and an interface to users to view information from different sources without the need to know the native data model and query language of the individual information sources. The CDM of mediator architectures helps to achieve integration of information from several heterogeneous information sources by resolving different types of heterogeneity. In some cases the CDM approach also serves additional purposes. For example the CDM and query language of TSIMMIS [Gar97] supports a rich collection of structures, handles missing information in an information source and also provides meta information about the heterogeneous information sources it supports. The common query language, in addition to querying data sources also enables joining of different mediators to provide additional functionality. In the case of the COIN [Bre97] framework its logical inference based context mediation approach provides a common vocabulary and a mapping which the individual data elements of different sources refer to. The DISCO [Tom98] system uses common standards at the middleware layer to provide query optimisation facilities to solve mismatches between the DISCO system and underlying data sources and a mechanism for defining cost of operation in terms of time, tuples produced, size of data returned in bytes and the number of distinct attribute values. The SIMS [Are93], [Are97] system uses a semantic data model to describe the application domain and to help users determine which information sources contain relevant data. Column 14 of Tables 1, 2, 3 and 4 identifies the CDM of the various information sharing models reviewed as part of this research.

4.3.4 Ontologies

Ontologies represent concepts in the form of classes, relations and functions which have agreed definition and relationships among them in a given domain [Gru91], [Gru93]. A number of approaches reviewed use ontologies to aid information retrieval, processing and sharing. For example the SIMS [Are93], [Are97] system provides an application domain

4. Information Sharing in Distributed Environments

ontology which specifies the terms and relations users can use to query information sources. The InfoSleuth [Bay97] system is designed to address information sharing in open and dynamic environments where there is no centralised management and a request for information can be represented in generic terms. Therefore the system uses ontologies to provide a concise, uniform and declarative description of semantic information about information sources which is used by broker agents to semantically match a user's information needs with the available resources. The Carnot system [Woe93], [Sin97] which is identified as one of the pioneers of *an ontology-based approach to interoperability* relates different database schemas to a common ontology to establish semantic relations between schemas for the purpose of interoperability. The XML data integration approach proposed by Lehti and Frankhauser [Leh04] uses the Web Ontology Language OWL [Bec04] as a global schema and mapping language for integrating multiple heterogeneous data sources. The approach identifies how the semantic relationship features provided by OWL can be used to map heterogeneous data sources to the common global schema and how the reasoning facilities of OWL can be used to check the consistency of these mappings.

4.3.5 Knowledge Based Information Sharing Systems

Knowledge based systems differ from traditional DBMS, in that they provide rule-based processing services which are triggered when users invoke certain methods [Su95]. They are characterised by rich representational structures, inference and reasoning mechanisms, deductive rules and integrity constraints for efficient management and access to knowledge bases and concept descriptions [Myl96]. Out of 40 systems surveyed, 8 use knowledge bases or knowledge based or reasoning techniques to aid information management for information sharing. The Infomaster [Gen97] system uses a knowledge base for storing rules and constraints to describe information sources from which data is retrieved and to perform data translation to effectively answer user queries. HERMES [Sub95] is a system for semantic integration of information from heterogeneous data sources and reasoning systems. DIKE [Pal03] provides support for building inter-schema knowledge by providing support for extracting knowledge patterns. This then aids the derivation of useful and complex relationships between concepts in two or more input databases to establish data

4. Information Sharing in Distributed Environments

semantics. In the SPICE [Jon00], [Xu02] system the Common Access System (CAS) knowledge repository provides information on which databases to query based on a user's requests. In the COIN [Bre97] framework the context mediators resolve semantic conflicts among heterogeneous systems through comparison of contexts which are associated with data sources engaged in data exchange. This context based mediation uses logical inferences which are stored as context knowledge and provide a common knowledge structure for the COIN queries and this aids resolution of semantic conflicts. SIMS [Are93], [Are97] uses a knowledge representation system to describe domains holding information in information sources. This covers structure and contents. In SIMS a knowledge representation model is required to identify the content and structure of the information sources and to determine complex relationships between information requested and data available from various sources before queries are made. Additionally the SIMS system integrates information from both knowledge bases and databases. The knowledge discovery methods of the Carnot [Woe93], [Sin97] framework are used to infer "*patterns and regularities*" (for consistency purposes) from the enterprise information sources using knowledge discovery tools. The framework also provides a knowledge base for storing *commonsense* knowledge of specialised domains, which is then used to establish relationships between different databases. Another knowledge based tool called KRBL (Knowledge Representation Base Language) is used to represent and access ontologies. HODFA [Kar95] framework uses a knowledge directory to maintain information about processing requirements, which is used to transform heterogeneous information systems onto a homogeneous environment.

4.3.6 Section Summary

"One of the main difficulties in supporting global applications over a number of localized databases is to cope with heterogeneities of these systems. These systems were not originally designed to facilitate any cooperation and there is no general model for interoperability among such isolated software systems. Should these systems exist under a homogeneous environment, global applications built upon a common set of tools and services can be developed much more efficiently." – Karlapalem et al. [Kar95].

4. Information Sharing in Distributed Environments

Ontology based approaches show that ontologies can be used to establish relationships between different concepts to aid global users to view information in an integrated way, by aiding the resolution of semantic mismatches or grouping semantically equivalent concepts together. An important reason to use ontology based approaches is to address heterogeneity problems which build up over time in an information system's life, as administrators or local users have autonomy to customise a system according to their needs. Although it is important that local users have autonomy so that they use the system according to their needs, this feature implemented at a local level poses difficult challenges at the global level, where information from several autonomous sources is required for global decision making. Knowledge based approaches are also required for integrated information retrieval in environments with little or no centralised control or in heterogeneous environments. Knowledge based systems provide reasoning mechanisms to support intelligent retrieval of information for a user query by providing reasoning and inference capabilities. They provide several benefits as identified in the systems reviewed. It can be inferred from Section 4.3.3 that even though information sharing systems operate in heterogeneous environments, it is extremely important to integrate these systems through a common platform, so that standard mechanisms can be used to access information from all sources despite heterogeneity. Therefore in most of the systems surveyed, a common data model and common query language approach, or other common techniques are used. In fact the models use a variety of techniques to tackle heterogeneity - wrappers, ontologies, mediators, knowledge and schema integration (see Section 4.4) to achieve the goal of integrated access via standard means.

WebFINDIT [Bou00][Bou94] describes an approach to information sharing by creating coalitions (a group of databases storing information pertaining to a common domain of information) and linking them via service links. The WebFINDIT approach to sharing information is appropriate, if a number of web-accessible databases are willing to be linked via the service links. However in MDSSF service links cannot be established between web-accessible supplier databases, because product suppliers are business rivals and do not wish to share their sensitive product information with other competing suppliers on the grounds of privacy and protection of trade secrets. In WebFINDIT, a query can navigate through a

4. Information Sharing in Distributed Environments

number of coalitions via service links to find relevant information. This is also not applicable in MDSSF, as there are no links between product supplier databases. The architecture of the WebFINDIT System, is not suited to MDSSF as it is not extendable as *“the number of participating databases in the coalition is usually very small”* [Bou00]. In contrast, the MDSS System incorporates in its federation a cluster of machines in a Grid network to access a large number of web-accessible supplier databases simultaneously. In their paper [Bou00] have 19 service links for 13 databases grouped into 4 coalitions. We believe that an increase in databases and coalitions in this model will bring about a higher increase in service links which are not only difficult to manage but are using a less efficient technique to access information from a large number of databases.

In MDSSF the need to establish relationships between different concepts does not arise, as the product supplier databases do not interoperate with each other. Additionally different product suppliers use the same data model, therefore there are no semantic conflicts. Also since the MDSSF is based on a CDM the issue of addressing heterogeneity does not arise. All approaches surveyed use a common data model at the middleware level to provide access to heterogeneous resources via standard means. MDSSF also uses a CDM, but it is not required at the middleware level because product suppliers already provide information to the federation users in the CDM of the federation. Ahmed et al. [Ahm91] recognise that *“One approach to reduce the number of mappings between diverse data systems is to define a common data model and language”*. This means data conversion techniques are not required in MDSSF. This fact has also been identified by Tomasic et al. [Tom98] who state that: *“There are several approaches to handling the source capability problem. One approach is based on standardization – all underlying data sources are required to have the same functionality or conform to the same communication standard”*. Because of the use of a standardised and homogeneous data model, a standardised communication mechanism and a loosely coupled approach no changes are made to the components of the MDSSF architecture (at the middleware level), that access data from several data sources. The assumption of a CDM for the federation allows product suppliers to manage their product data, provides an opportunity to design a novel database federation model which uses a Grid based search mechanism at its core in place of wrappers and mediators to

4. Information Sharing in Distributed Environments

aggregate information from different product supplier databases without worrying about heterogeneity issues.

4.4 Database Interoperability and Schema Integration

Database interoperability is an important research issue and is applicable in environments where data held in more than one database needs to be interoperated to provide information. Distributed and federated database systems support such database interoperability. This is achieved by integrating data available from participating databases and this process creates a virtual database which is logically defined but not physically centralised [Par98]. The virtual database is created by integrating schemas of databases into a schema of the virtual database called the global schema. In the process of database integration a single unified description of schemas called an integrated schema (or global schema) is created from the set of input schemas. A global schema provides a mechanism enabling access to data from several data sources in an integrated way, where the integrated access is achieved via the mapping information, which associates an integrated schema with the input schemas [Par98]. Parent and Spaccapietra [Par98] identify three steps for developing an integrated schema (Figure 4.1):

- **The Pre-integration step** transforms input schemas and makes them more homogeneous syntactically and semantically. The outcome of this step is representations of input schemas in the same data model called the common data model (CDM) of the virtual database.
- **Correspondence identification** At this stage identification and description of inter-schema relationships is undertaken to establish commonalities between databases. Two databases have something in common when the real world facts they represent have common elements or are interrelated in some way.
- **Integration** of input schemas is the final step which resolves inter-schema conflicts and unifies corresponding items to create an integrated schema.

4. Information Sharing in Distributed Environments

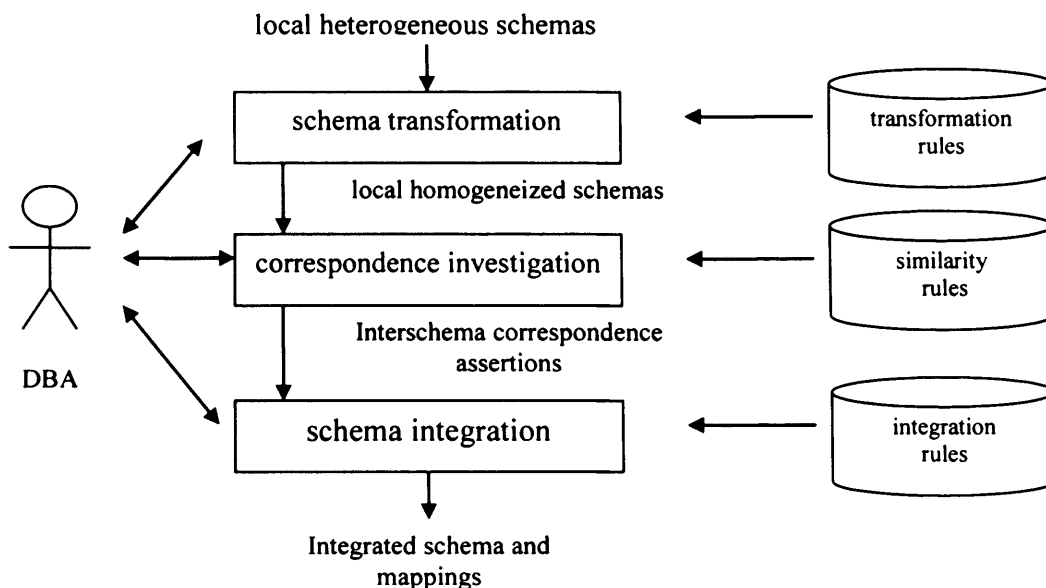


Figure 4.1 The global schema integration process identified by Parent and Spaccapietra [Par98]
Source: Parent and Spaccapietra [Par98]

Several different approaches to schema integration were studied with the aim of understanding how they integrate data so that their potential could be assessed for use in this project. A summary of these approaches identifying the key features and schema integration strategies adopted is presented in Table 5 (see Appendix 3). Some of the important reasons for schema integration identified by Batini et al. [Bat86] include different perspectives of user groups in modeling the same object, equivalence among constructs of the model, incompatible design specifications, common concepts which have different representations, and semantic and structural incompatibilities. Batini et al. [Bat86] provide a comparison of 12 different methodologies for database schema integration with different solutions to schema integration problems. For the sake of brevity they are not listed here. However fundamental problems identified by them, and addressed by these schema integration methodologies apply to the resolution of semantic and structural differences in schemas.

The schema integration strategies reviewed are used in different contexts and scenarios to enable database interoperability in heterogeneous environments. The aim of the review was to identify reasons for schema integration. The need for schema integration arises when

4. Information Sharing in Distributed Environments

legacy data having different representations are to be presented to the user in an integrated way conforming to a set of rules for its uniform analysis and interpretation. The schema integration approaches identified in Table 5 (Appendix 3) address these issues in various ways. Although the aim of this research is not to present a new schema integration approach but to understand existing approaches, a study of these is necessary as this helps understanding of how to build information sharing systems. The second reason for this work was to distinguish between information sharing models, which use schema integration approaches and the MDSSF approach to information sharing, which does not use a schema integration approach, in order to highlight the MDSSF's novel architecture.

Of the 40 different information sharing systems reviewed 14 systems or architectures (see Appendix 3 Table 2 Column 10; and Table 1 Column 3) supported a schema integration (or data or view integration) approach for providing integrated access to heterogeneous resources. Although the mediator based systems support data integration, they generally do not use schema integration techniques because mediator based systems support structured, semi-structured and unstructured resources. The two XML and mediator based information sharing models (XML-based Mediation Framework (XMF) [Lee02] and the XML Data Integration System [Alm04]) support XML based schema integration to create a global schema. This aids query formulation and retrieval of data from appropriate sources and provides a homogeneous view over heterogeneous XML data. Another XML based system MIX [Bar99] uses XML DTDs (Document Type Definition) of mediator views (which are effectively mediator schemas in XML), in an integrated way for construction of queries in an intuitive way [Bar99]. MVDS [Duw96] system provides multiple integration views to support integration of the same information distributed across multiple sites in various ways tailored for different users and applications.

Schema integration is required to meet many different user requirements and the approach chosen will vary with the aim. For example, the key concept behind the DATAPLEX [Chu90] approach is to enable sharing of information in environments where a standardised model for data management using a single DBMS cannot be adopted because of diverse and specialised user requirements, which a single DBMS approach cannot satisfy, such as

4. Information Sharing in Distributed Environments

system constraints (for example some DBMS can only run on specialised computers) and rapid technological evolutions. In such environments where different DBMS have to coexist to meet specialised user requirements and retain different types of autonomies, integration of heterogeneous databases is one of the most appropriate solutions to achieve an integrated access to several databases. The *mediator-like* DIKE [Pal03] system provides a mechanism for identification of sub-schema similarities, resolution of conflicts such as naming conflicts (homonyms and synonyms), type conflicts and structural conflict to establish inter-schema relations. The mediator and wrapper based Garlic [Car95] system supports different types of information from heterogeneous information sources, therefore it requires the approach of schema integration and wrappers to share data in a uniform manner. A notable exception is the MRDSM [Lit85] which is a relational multi-database management system which do not use schema integration as it provides techniques for joint manipulation of different databases via a single query.

“...to define a global schema over a large number of databases should typically be at least a very complex task. Also, users may not wish a global administrator, even if the distributed database may be created technically. These reasons seem in particular most responsible for the lack of distributed database systems in service, despite the large research effort during the last decade.” – Litwin [Lit85]

As well as the DATAPLEX, DIKE and Garlic approaches, the other information sharing models or reference architectures, identified in Table 1 and 2 (see Appendix 3) also have similar problems caused by the heterogeneous data models, query languages and specialised user or system requirements which makes it necessary for these approaches to use schema integration techniques. However in MDSSF, structural and semantic heterogeneity does not occur in the standard data model and a common query mechanism is adopted throughout the federation. For example, if a contractor is retrieving information for a particular product from several sources the data providers (suppliers) will provide information for that product only using a representation which is used by all the suppliers. Because of the small level of heterogeneity, various types of conflicts, such as classification, naming, semantic, syntactic do not arise in MDSSF. Lack of such conflicts in

4. Information Sharing in Distributed Environments

the MDSSF provided the author with an opportunity to look into novel ways of building federated database-based information models without using schema integration techniques. We believe this is necessary to support new business models for collaborative working, of which the PSCD application is one example. The PSCD application provided us with an opportunity to test the applicability and use of the MDSSF architecture in such business models.

4.4.1 Mediator vs. Schema Integration Based Systems

An important advantage of mediator based architectures over schema integration based architectures is that mediator based architectures provide scalability to incorporate a growing number of information sources in dynamic information sharing environments. In global schema integration based approaches, every time a new data source has to be added, it may require a DBA to change the global schema and add new definitions. This can be a challenging and hard problem to address in dynamic environments with a growing number of data sources. This is also recognised by Lie and Pu [Lie95] in:

“As the number of databases participating in the interoperable database system increases, the design of an integrated schema involving n different systems requires to reconcile an order of n^2 possibly conflicting representations (i.e., heterogeneity in semantics or in data formats). Such activities are time consuming and can be aggravated when incorporating the system evolution issue with the integration strategies.” - Liu and Pu [Liu95]

4.5 Federated Database Systems and Federated Information Systems

“Distributed databases may be either homogeneous or heterogeneous. Homogeneous distributed databases require that every underlying database conforms to the same data model and query language. Heterogeneous distributed databases relax this restriction and permit each underlying database to have different data models, query languages, and thus, different functionality.” – Tomasic et al. [Tom98].

4. Information Sharing in Distributed Environments

An FDBS is a collection of cooperating but autonomous component database systems (DBSs) [She90]. The component DBSs take part in a federation to serve the data needs of federation users. Creating an FDBS involves resolving various issues such as data distribution, heterogeneity and autonomy. It provides an integrated and transparent mechanism of accessing information from a number of component DBSs for different classes of federation users. The author investigated the existing FDBS architectures and federated information systems to identify their applicability in the MDSSF model of information sharing.

The two important characteristics of FDBS identified by Sheth and Larson [She90] and applicable to the MDSSF are distribution and autonomy. In the MDSSF, product data is distributed across several product supplier databases. Although several types of autonomies such as design, communication, execution and association have been identified, not all of them are presently supported because of the nature of the challenge the MDSSF aims to address. For example the MDSSF does not support design autonomy because the MDSSF is built on the assumption that product suppliers will subscribe to the standard PCD system and provide product descriptions based on its constructs. Heterogeneity in an FDBS is primarily caused by design autonomy among component databases [She90]. Since the MDSSF provides a fully designed and implemented DBS system to the product suppliers, it does not allow changes to its data model by product suppliers. The communication autonomy is also restricted in the MDSSF architecture. The product suppliers do not communicate with other product suppliers as they do not share their product related sensitive data with their competitors. However the MDSSF supports execution and association autonomy of product suppliers to allow them to perform local operations on their databases without any external interference, and exercise their right to associate or disassociate from the federation at their will. The third characteristic of FDBS called heterogeneity is presently not applicable to the MDSSF as it does not support heterogeneous product data representations in its environment. The homogeneous MDSSF supports only product data representations which conform to its standard product class data model.

4. Information Sharing in Distributed Environments

The author has reviewed a number of information sharing systems, which are built using federated database concepts (i.e. they support three different characteristics - distribution, autonomy and heterogeneity). Almost all the systems identified in Table 2 (see Appendix 3) under the category of schema integration belong to this group and all these systems also support data manipulation operations in addition to database access operations. However an exception to this rule are the last two systems in this category the XML Data Integration with OWL [Leh04] and the MVDS [Duw96] system which provides data access operations only. The XML DATA Integration system (which supports access to data resources accessible via the web) uses the Web Ontology Language OWL of W3C [Bec04], as its global schema definition language. This provides semantic mapping features so that heterogeneous data resources can be accessed using a common vocabulary. The MVDS system supports the creation of multiple integration views to support user or application requirements, which may need access to the same data in different ways. Another exception to this rule is the multi-tier federated approach of the SPICE system [Jon00], [Xu02] which does not integrate schemas, but wraps the heterogeneous taxonomic databases to a fixed schema of the wrappers. This provides a standard interface based on a common data model and hides database heterogeneity.

We also reviewed FDBS architectures which do not support global schema integration (identified in Table 4 (see Appendix 3)). The federated database architecture proposed by Heimbigner and McLeod [Hei85] is based on a loosely coupled approach which enables information sharing between office information systems. This information sharing approach, which unites a collection of independent database systems, is based on three schemas: export schema, import schema and private schema. The approach allows users to specify information they wish to share with other users via their export schemas and define import schemas to gain access to information available from other databases (via their export schemas). The information described using private schemas is not shared. The MRDSM [Lit85] system is also based on a loosely coupled approach of managing relational databases without the need for schema integration. An important objective of the approach is joint manipulation of different databases via a single query without the need for a global schema. The paper identifies that a global schema integration is not conducive

4. Information Sharing in Distributed Environments

because users are unlikely to understand all the schemas in a “*reasonable*” learning time and typical users do not usually deal with all the data of an enterprise. It is also identified that schema integration imposes additional constraints and denies users the flexibility to model their universe according to their needs. Heimbigner and McLeod’s [Hei85] federation approach does not support schema integration in order to minimise the role of central authority. The federated database models proposed by Hsiao [Hsi92] [Hsi92a] outlines five different approaches to data sharing which are largely based on database conversion and mapping techniques to address different types of heterogeneities.

The extended schema architecture of the INFINITY [Här97] FDBS prototype is based on not only having a CDM over local heterogeneous schemas but also a common schema structure provided by the ISO’s standard for the exchange of product (STEP) [ISO94], [Gle89] data. In the two-step process adopted by the system, the system first achieves the “*data-model homogenized*” schema (equivalent to the CDM of the FDBS [She90]) and the second step resolves semantic conflicts to create a “*schema-structure homogenized*” schema. The extended schema architecture approach of INFINITY [Här97] strengthens this research by demonstrating that there is a need for a CDM and common schema structure approach in order to achieve flexible access to heterogeneous databases. Although both systems use different product data standards, they are based on the similar concept of providing access to underlying databases using a CDM approach. The MDSSF already provides a CDM in the product class data model, whereas INFINITY uses a two-step approach of schema integration to resolve data model and schema heterogeneities and correspond to the STEP standard in order to create the CDM of the system. Other distinctions can also be identified in the two approaches on the basis of MDSSF’s Grid based search and subscription based approach.

Although the MDSSF possesses FDBS characteristics such as distribution and autonomy it cannot be called a fully fledged FDBS, because the tasks required for building a FDBS such as schema translation and schema integration are not required for the MDSSF approach because of the use of a CDM. The five level schema architecture of an FDBS proposed by Sheth and Larson [She90] integrates multiple export schemas to create a federated schema.

4. Information Sharing in Distributed Environments

Since MDSSF does not support schema integration, the approaches identified by various federated systems reviewed here are not suitable for building MDSSF. Another important task relevant to FDBS, transaction management, is also not relevant to the MDSSF, as it provides database access operations only. The FDBS systems (and also some of the mediator based systems, such as DIKE [Pal03]) are created to share data between independent systems [She90], [Hei85], which is also not the case in MDSSF as data sharing between the independent product supplier databases does not occur. From our evaluation of traditional FDBS systems it can be concluded, that the traditional FDBS architectures lack functionality and scope to meet the needs of new information sharing models.

4.5.1 Federated Information Systems

In recent years the concept of federated information systems (FIS) has been introduced in research aimed at building interoperation solutions for heterogeneous and autonomous information systems [Bus99], [Wys03], [Has00], [Con99]. An important characteristic of FIS systems is that they are constructed by creating an integrating layer over the existing legacy applications and databases [Bus99]. These federated information systems can be classified into three different types: loosely coupled information systems, federated databases and mediator-based information systems [Bus99]. Busse et al. [Bus99] further identify that “*evolvability*” of information systems is an important aspect that characterises federated information systems. According to the definition of FIS by Busse et al. MDSSF can be classified as a FIS to a certain extent with the Grid based search mechanism at the core of MDSSF considered as the federation layer providing uniform access to autonomous supplier databases. MDSSF also supports the evolution feature of an FIS as it provides a versioning capability to describe new product data with enhanced features and functionality. A key research area of FIS is “*the systematic development of interoperable solutions for autonomous, heterogeneous systems covering both database sources and also non-database information sources, providing (structured or semi-structured) files, multimedia data, or proprietary systems’ data.*” [Has00]. Based on the focus of FIS research, identified by Hasselbring et al. [Has00] and Busse et al. [Bus99], distinctions between FIS and MDSSF can also be identified on the grounds of: interoperability (and therefore exchange of data) between the participating supplier databases which MDSSF

4. Information Sharing in Distributed Environments

does not support; absence of wrappers because of the adoption of a common data model approach; and use of only database systems product data cannot be stored in files because of their complex hierarchical structure and interdependency.

Another class of information system identified by Busse et al. [Bus99] is loosely coupled information systems which do not offer a federated schema. Busse et al. also identify that loosely coupled information systems do not offer location or schema transparency. Although MDSSF does not provide a federated schema it offers location and schema transparency. In MDSSF contractors do not need to know the physical location of supplier databases. They are also not required to be aware of the product class schema (which provides mechanisms to describe different product descriptions) as product data is presented to them in a readily usable form. Distinctions between the three types of FIS have been identified by Busse et al. based on thirteen different classification criteria. These are presented in the Table 6. The author has extended the table by adding an extra column in the end, which identify the features of MDSSF, to distinguish between MDSSF and other FIS types.

4. Information Sharing in Distributed Environments

Classification criteria	Loosely coupled Information Systems	Federated Databases	Mediator-based Information Systems	MDSSF
Types of heterogeneity addressed	Only technical and language heterogeneity	All, except query restriction heterogeneity; schema integration difficult for schematic heterogeneity	All.	None. Based on common data model
Loss of autonomy?	Execution autonomy	Execution autonomy; notification of schema changes	Execution autonomy	Design and communication
Transparency	Language	Location, schema and partly language	Location, schema and partly language	Location, schema and language
Kind of component	Structured	Structured	Any	Structured
Access methods	Query language	Query language	Any	Web browser
Access restrictions	No	No	Yes	Yes
Write access?	Yes	Yes	No	No
Tight vs. loose integration	Loose	Tight	Tight	Loose
Kinds of semantic integration	Collection	Collection and fusions	Collection, fusion ¹ , sometimes abstraction	Collection
Necessary metadata	Technical, infrastructural	Logic, technical, semantic	Logic, technical, semantic	Technical, infrastructure
Bottom-up vs. top-down	Not applicable.	Bottom-up	Top-down	Not applicable.
Virtual vs. materialised	Virtual	Virtual	Virtual	Virtual
Evolvability	High	Low	High	High

Table 6: Types of FIS as identified by Busse et al. [Bus99]

Source: Busse et al. [Bus99]

(Note: The last column of MDSSF in the table is added by the author for comparison)

The table identifies the distinct features of MDSSF, in comparison with other FIS. These distinct features are required to address the needs of new information sharing models whose conception has been made possible by industry requirements, information explosion, ubiquity of the internet and development of Grid technology (see Section 4.6) which

¹ Fusion is the process which identifies semantically equivalent entities retrieved from different information sources [Pap96a].

4. Information Sharing in Distributed Environments

provide ways to access different types of resources (computational and software) for analysing and processing large scale information outside the boundary of information users or providers such as contractors and product suppliers.

4.6 Grid-based Systems

Information access and its management is an important area of research when creating new models of information retrieval and sharing to meet the needs of business paradigms whose conception has been made possible by the emergence of network technologies such as the Grid and Web Services. An important aspect of the Grid concept is “*coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations*” [Fos01]. Middleware infrastructure available via the Grid environment enables the creation of distributed and scalable virtual organisations (VO) for sharing and utilising resources in a particular domain [Fos02]. Web Services are a paradigm enabling computing in distributed and heterogeneous environments [Fos02]. Research in the area of Engineering Federated Information Systems (EFIS) has recognised “*GRID*” computing as an emerging area for building new models of data exchange [Wys03].

This research creates a Grid enabled solution to federate a large number of autonomous databases, where the Grid layer provides a scalable solution to search a large number of databases in real time in response to contractor requests for required product information, available from a large number of supplier databases. In this context, understanding Grid technology and the benefits it can provide is important to this research. A novel feature of this research is large scale information retrieval, information processing and analysis in the Grid environment and sharing via collaborative means and in an integrated way, in federated database environments with participants such as suppliers and contractors belonging to different organisations. In such complex information sharing environments, Grid computing is perceived to be an important way, to provide the necessary infrastructure to support such information sharing models, which bring together actors from several autonomous organisations. Grid environments provide additional functions to the existing functionalities of the Internet. As well as high speed networking, it offers features such as enhanced security infrastructure including single sign-on capability, security between

4. Information Sharing in Distributed Environments

consortia, simple setting up of networks to support VOs, distribution of computationally intensive jobs across multiple distributed processors and resource information sharing. As part of this research the author reviewed various Grid based systems, middleware and toolkits, which provide different functionalities for enabling information processing in distributed environments. Table 3 (see Appendix 3) identifies four such Grid based systems.

European Data Grid's Data Management Work Package [Eur03] [Gag02], provides a mechanism to access relational databases in a Grid environment with the aid of the Spitfire System [Spi03] [Bel02]. The Spitfire project is optimised to support common database operations from the Grid environment. It allows simple access patterns to a database where access is required to a number of rows in a lookup operation. This approach is useful when data is retrieved from one or a small number of database systems. Spitfire does not address the issue of data retrieval from a large number of autonomous databases.

The OGSA-DAI [Atk05], [Ogs07] is a middleware which allows access to relational and XML databases in a Grid environment. Using the tools provided by the middleware a standardised mechanism can be created to access different RDBMS. The MDSS component of MDSSF accesses product suppliers' databases in a Grid environment. Although the facilities provided by the OGSA-DAI middleware could have been used for MDSS, the approach and means of accessing databases via OGSA-DAI was not found suitable for MDSSF. Based on our evaluation of OGSA-DAI's framework against MDSSF requirements, the approach was found to be complex, adding an extra layer of complexity without giving any significant benefits. In the OGSA-DAI framework requests for data from a given data resource are specified in *perform documents* which identify database operations, the framework should perform on behalf of the client. The framework provides a set of pre-defined activities, which are specified in *perform document*. These activities identify the SQL queries, the user wishes to execute on databases. In MDSSF, the user does not specify queries via SQL statements. The user identifies the products for which information is required and then SQL statements are executed as part of stored procedures at remote supplier databases. Additionally in the OGSA-DAI framework, each *Data*

4. Information Sharing in Distributed Environments

Service Resource component (which is part of the OGSA-DAI core), supports a single data source only. Therefore providing access to a large number of databases via the *Data Service Resource* component would require setting up a large number of instances of these components and creation of a large number of *perform documents* to query a large number of supplier databases in real time. This would be a complex approach to meeting the MDSSF requirements. Although the framework provides mechanisms for *Data Service Resource* components to access multiple data resources at a time, this access is provided via a JDBC interface, which is also not suitable for our approach. In fact, access to RDBMS in the OGSA-DAI framework is only provided via JDBC interfaces. This is appropriate when the databases are available in a Grid environment, whereas in MDSSF the product databases are not available in a Grid environment. In MDSSF product databases are subscribed to and managed by independent product suppliers, therefore a JDBC based approach is not appropriate because of security, scalability and interoperability reasons. In MDSSF access to supplier databases is provided via web services, which allow invocation of underlying database operations in a platform independent way. The MDSS Grid (which is the core of the MDSSF) only has database access operations for accessing a large number of databases because the supplier databases are not part of the Grid environment, whereas the OGSA-DAI approach is more suitable to scenarios where databases are available as part of a Grid environment.

The workflow based GeneGrid [Kel05] [Jit05] system provides access and integration of disparate and heterogeneous applications and datasets from across the globe through the creation of a 'Virtual Bioinformatics Laboratory'. It provides access to resources and tools to biologists interested in the development of antibodies and drugs. The Biodiversity World (BDW) system [Pah06], [Pah06a] is also a workflow based system which provides access to biodiversity resources in its Grid environment for analyzing biodiversity patterns. In addition to the Grid based middleware and systems identified in Table 4 (see Appendix 3) other Grid based systems such as myGrid [Gob03], BASIS [Gil05], SEEK [Sek06] and Chimera [Fos02a], were reviewed, in order to learn about the data sharing approaches these systems use and the functionalities they provide. Clear distinctions can be drawn between these approaches, which provide integrated access to diverse resources and the approach

4. Information Sharing in Distributed Environments

used in MDSSF. However for the sake of brevity these approaches are not elaborated here. In recent years the portal based approaches have also become popular for providing access to Grid environments via web browsers. Pegasus [Sin05], GENIUS [Gen07] and P-Grade [Gra07] are examples of popular portal based approaches.

4.6.1 USECA Properties

According to Ling and Pu [Liu95] large scale, dynamic and interoperable database systems should have five properties, referred to as USECA (Uniform access, Scalability, Evolution, Composability and Autonomy) which are critical for a system to be useful. The USECA properties proposed by Liu and Pu [Liu95] and Lee et al. [Lee97] in the DIOM system are applicable in the MDSSF model of information sharing because MDSSF has the potential to be deployed in large-scale and dynamic environments. MDSSF provides “*uniform access*” to a large number of supplier DBS using a single system. The architecture provides “*scalability*” by supporting a growing number of suppliers who wish to join the federation using a subscription based approach. It also provides Grid technology support, so that a large number of supplier databases can be searched in real time in response to a contractor’s query. The architecture supports an “*evolution*” property by providing evolvable data structures to meet the changing requirements of suppliers to manage new and evolving products, whilst maintaining backward compatibility to old and existing products. The DIOM architecture supports a “*Composability*” property to manage the incremental design and construction of interoperable interfaces so that data could be integrated in many ways to produce different kinds of integrated systems. Since MDSSF does not support interoperation between supplier DBSs the property of “*composability*” has been considered from a different perspective in our research. In the MDSSF environment, the Product Class data model allows composition of different types of product classes to manage different types of product data whilst retaining similar interfaces and without the need of making changes to the underlying data structures. Because the underlying data model does not change the interface constructs for viewing and sharing data also do not change. Finally in MDSSF the “*autonomy*” property is supported as the autonomous product suppliers retain full control of their product data.

4. Information Sharing in Distributed Environments

4.7 Other Information Sharing Systems

In addition to the mediator based and schema integration based systems other information sharing systems were reviewed, which provide access to information from different resources in their environment. Two important systems which fall in the context of this research and which also provide some similar features are the E-Union [Kon04] framework and the Information Manifold [Lev96] system. These are briefly described here.

E-Union [Kon04] provides a “*platform*” for information sharing between E-commerce systems for construction material procurement by linking different E-commerce systems together. Its key framework feature is that it allows linking of different material trading sites so that a buyer can also view information provided by the other websites, when accessing a website which is part of the E-Union framework. Although both MDSSF and the E-union frameworks bring together construction supply chain actors for procurement, the systems use different approaches. The E-union framework is suitable for environments where partnering relationships exist between different suppliers because it allows inter-organisational information sharing. However MDSSF is built on the concept that product suppliers do not share their product information with their competitors on the grounds of privacy. Therefore it does not allow inter-organisational sharing of information. MDSSF links different construction supply chain actors from a different perspective. It can be perceived as a service provider whose services can be used by any autonomous contractor or supplier (by subscribing to the PCD data model in the case of a supplier). Whereas the E-union framework only caters to a group of organisations who wish to combine their systems together to aid buyers to view information using a single website. It can also promote products available at other websites.

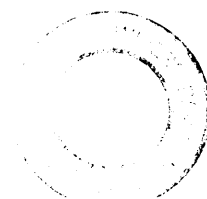
Information Manifold (IM) [Lev96] provides uniform access to more than 100 structured but heterogeneous information sources, many of them available via the web. There are interesting similarities between MDSSF and IM, however the approach adopted is different. The IM system demonstrates the applicability of information sharing models in real world scenarios. The important similarities and differences between the two approaches are now discussed.

4. Information Sharing in Distributed Environments

In both systems relational data models which also have object oriented features are used in order to describe the information content in greater detail. For example, in both information entities can be defined in terms of a class with a super class and a set of attributes. In IM a subclass “*UsedCar*” belongs to the superclass “*Car*” and has simple attributes such as *Model, Year and Category*. However in MDSSF, in order to describe complex construction industry product data, the Product Class Database (see chapter 5) provides different types of specifications so that a large amount of information about a given entity (and its different versions) can be specified hierarchically. Hence, unlike IM, MDSSF handles complex attributes. In IM, the data model of relations and classes is used as a schema (known as “*world view*” – a collection of virtual relations and classes) against which a user poses queries without needing to create an integrated schema. A similar approach of posing queries by using the Product Class schema without an integrated schema is adopted in the MDSSF. However unlike the “*world view*” schema (which does not store any data) the Product Class schema of MDSSF is used to store actual product data by suppliers and not to describe the contents of the information sources only. Since “*world view*” does not store any data, IM requires an additional step to relate the contents of information sources to the classes, attributes and relations in the *world view*, whereas in MDSSF this step is not required. Another fundamental and conceptual difference lies in the challenges, MDSSF is trying to address. MDSSF is built on the assumption that the product information will be retrieved by contractors or potential buyers. Therefore it is incumbent on product suppliers to actively participate in the federation (by subscribing to the Product Class data model) in order to gain the opportunity to sell their products. However the objective of IM is to provide integrated access to heterogeneous data sources using standard methods.

4.8 Chapter Conclusions

Several different information sharing models or reference architectures were reviewed, which had several different approaches to sharing information in an integrated way when the information is available from different autonomous sources. Several schema integration approaches were also reviewed to understand their scope and applicability to new information sharing models such as MDSSF. Various different methods have been suggested in the literature covering different approaches, such as ontologies, knowledge



4. Information Sharing in Distributed Environments

bases, mediators, wrappers, CDM, schema integration, CIS, FDBS, FIS and Grid-based. These address different types of conflicts and enable interoperability of information systems and/or provide information in an integrated way for different purposes. Although all these surveyed approaches provide useful techniques, none of the information sharing models or reference architectures (which use these approaches) fully met the requirements set for MDSSF's model of information sharing. MDSSF shares certain features with other models, but there are also clear distinctions, which highlight the novel features of the MDSSF. These novel features (see Section 7.4.2), such as its subscription based approach, its Grid based distributed database search, its virtual environment which allows suppliers to compete with each other, its cooperation model, its model of aggregating information from several databases without the need for schema integration techniques, allow MDSSF to meet the requirements identified for the new model of information sharing in construction industry domain.

5. The Product Classes

5.1 Introduction

Chapter 2 introduced the concept of product classes which are created using the Product Class Database (PCD) system. The prototype PCD system of the PSCD application supports the creation of product classes. The aim is that these will be utilised by the product suppliers to provide the description of their products in the Supplier Database (SD) system. This chapter explains product classes at the conceptual level. The architecture of MDSSF system components which enable creation and subscription of product classes; and creation of production information based on product classes is presented in chapter 6. Section 5.2 describes product classes and the benefits they provide. Product classes are composed of one or more different specifications which conform to individual specification types. Section 5.3 identifies different specification types that were developed as part of this research to support the description of product features. Section 5.4 provides a description of the versioning support that is required for the evolution of product classes and their specifications. In section 5.5 a brief summary is given of some of the important competing product modelling/management systems or reference architectures used in the AEC industry. These provide similar features to the product class concept. Finally Section 5.6 presents chapter conclusions.

5.2 Product Classes

Product classes enable the creation of standard product definitions which can be used by product suppliers to provide descriptions of products, they can supply to consortia members or contractors. These product definitions are called product classes. The creation of product classes is a basic task which must occur before products can be described in the common data structure of the MDSSF by product suppliers in their SD systems. It is anticipated that product classes will be created by industry knowledgeable specification designers, who have a full understanding of the features that are used by product suppliers to describe their products. Alternatively product classes based on the standards that are used industry wide or agreed by the user community can be created. In this respect product classes provide a benefit to product suppliers by describing their products using the industry defined or agreed standards and storing these standard descriptions in structured format in their SD systems.

A product class can be defined as an entity made up of a number of specifications. The specifications can be of several types and are created as part of creating a product class. The specifications correspond to pre-defined specification types and provide a mechanism for defining the properties of a product. In other words a product class is a template having a set of pre-defined attributes, which can be used by product suppliers to describe actual products and their features. Each product has a corresponding product class. For example a supplier dealing in furniture can subscribe to product classes such as the *Chair* product class, *Filing Unit* product class and *Desk* product class. A product class can be used to create descriptions of a number of products conforming to it in the SD System. For example, using a *Desk* product class, description of different kinds of desks (products) such as roll top and office desk can be created in an SD System. Product classes thus defined can be used to create product descriptions in SD systems by product suppliers who give values for the specifications defined for the product class. In this way product descriptions can be rapidly created once product classes are available.

To enable all product suppliers to adopt a standard mechanism for supplying product information in their databases, the notion of the product class was adopted. An important advantage of product classes is that they provide product suppliers with a readily available means to describe their product features in a standard and structured way. The concept of product class thus addresses the heterogeneity problem in MDSSF by providing a standard way for exchanging product data. It is envisaged that product suppliers will subscribe to the product classes that correspond to the products they supply. It is also expected that industry standards will be developed and standard criteria established by the industry actors for product class creation and their evolution.

The product database of the original PSCD application provides support for describing features of products by assigning specifications to product descriptions. However this support is limited to assigning atomic specification only to product descriptions. The construction industry products usually have complex specifications which cannot be described using simple attribute-value constructs in a database system. Hence, in redesigning the PSCD application and to provide product suppliers with a facility to describe their complex product features, which goes beyond simple atomic descriptions,

new specifications were introduced, as part of this research, to further the development of the product class concept. These specifications are described next.

5.3 Composition of Product Class

Defining a product class requires the definition of its specifications, as a product class is made up of a number of specifications. The specifications can conform to one of several specification types available in the PCD system, and are created as part of the product class design process by specification designers. These specifications are created from pre-defined specification types, and this provides a mechanism for defining the properties that a product class can have. For example, a *window* product class can have specifications (properties), such as *width*, *height*, *weight*, *wood type*, *panel shape* and *glazing configuration*. A particular product can conform to only one product class at a given time. A product conforms to a product class, when it uses the specifications defined for the class to store product information.

Designing a product class requires the creation of its specifications from pre-defined specification types. These specifications are independently created and are then assigned to the product class being designed. A product class is thus made up of a number of individually and independently created specifications. From a technical perspective this design model provides an important advantage. Since specifications are individually created, they can be assigned to more than one product class. This enables specification reuse and rapid product class development, as they are not required to be created during each product class design process. A specification designer creates a new specification only if it is not already available in the PCD system, otherwise an existing one is used. For example, if a unit specification such as *width* already exists in the PCD system, it can be assigned to *table*, *chair* or other product classes, which require such a specification, when they are created.

Figure 5.1 shows the five different specification types identified as part of this research. This also includes the unit specification whose concept was adapted from the product database of the original PSCD application. As identified in Figure 5.1, some specification types can be further decomposed into a number of sub-specification types. Decomposing specification types into sub-types, enables the description of complex

product attributes/features. The type of specification used for a product depends upon the product feature description requirements and the degree of complexity involved. Simple features can be described using the unit specification type, whereas complex features can be described using the other specification types, described here.

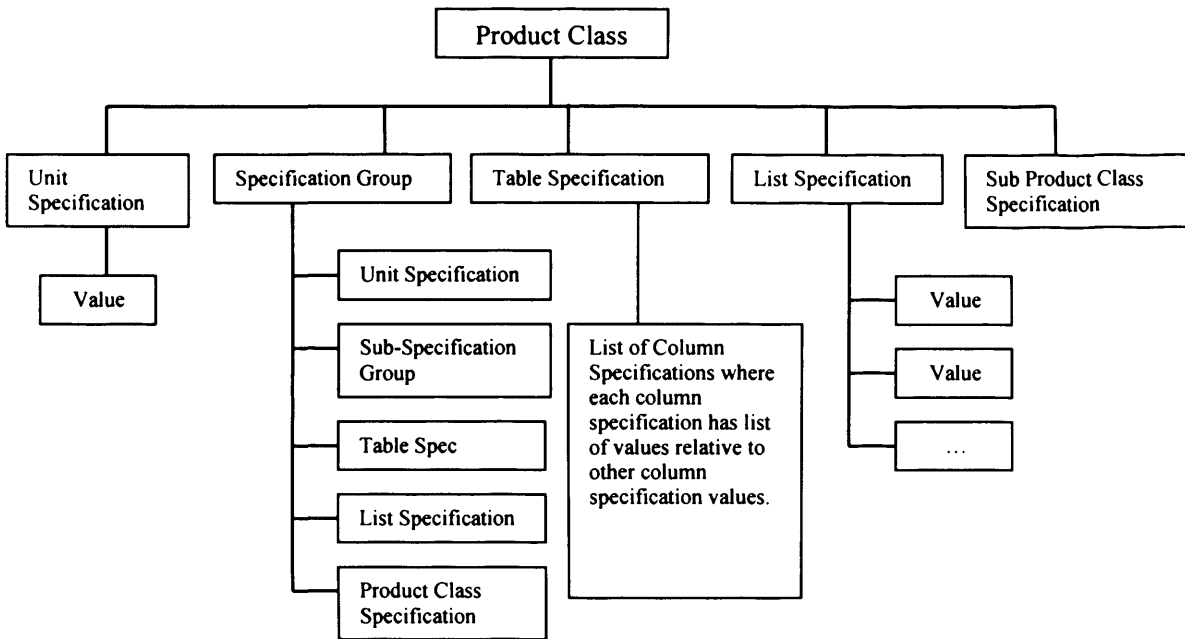


Figure 5.1 -The Product class and its various specification types

5.3.1 Unit Specification

The *UnitSpecification* specification type holds a single value. For example a specification *Manufacturer's Name* (which corresponds to a *UnitSpecification* specification type) holds the name of the manufacturer, as its value. The value of a specification can also optionally correspond to a measurement unit (such as centimetres, inches and degree) which identify its size, quantity and degree.

5.3.2 Specification Group

This *SpecificationGroup* specification type is used, when a number of different specifications need to be grouped. The need for grouping specifications may arise when a number of specifications, as part of industry standards need to be addressed as a single unit, or when they are commonly used across more than one product class. Grouping specifications also makes the development of new product classes easier, as it supports the benefit of reuse and rapid assignment of grouped specifications to new product classes (using a single operation) which saves effort and time. This is particularly

important when creating complex specification groups, which group several different specifications together. A specification group can be made up of individual specifications, sub-specification groups, product class specifications, list specifications and table specifications. The example (see Figure 5.2) shows a simple specification group. The name of the specification group is *Performance Criteria* (which corresponds to *SpecificationGroup* specification type) of an air conditioning fan coil unit and is made up of a list specification (see Section 5.3.3) and a number of unit specifications.

Performance Criteria
Electrical Supply: 240v, 1ph, 50 hz.
Chilled Water Flow Temp: 6 Degrees C.
Chilled Water Return Temp: 12 Degrees C.
External Static Pressure: 30 Pa.

Figure 5.2 An example showing a Specification Group

5.3.3 List Specification

In Figure 5.2, the *Electrical Supply* list specification corresponds to the *ListSpecification* specification type, which contains three values with their corresponding measurement units. List specifications are used to describe product features which have multiple values. List specifications can also be used to provide product suppliers with a list of pre-defined options from which one option can be used. For example, in a furniture equipment product class, the *wood type* list specification, can contain values, such as *oak*, *pine* and *cherry*.

5.3.4 Table Specification

A *TableSpecification* specification type comprises of a number of column specifications, where each column specification has a list of values. Specification values in each row are part of a collection, where each value describes some aspect of a product. A table specification can be used in a product class to represent technical details of the product in the form of rows and columns. Figure 5.3 is a table specification representing specification values of a series of fan coil units.

Chester HBWS Series 200	GUIDE NR	AIR FLOW RATE (L/S)	MAX DUTY AVAILABLE		AIR OFF 12°C		AVAILABLE HEATING DUTY (KW)	R.L.C. (A)
			TOTAL COOLING DUTY (KW)	SENSIBLE COOLING DUTY (KW)	TOTAL COOLING DUTY (KW)	SENSIBLE COOLING DUTY (KW)		
201	30	110	1.71	1.45	1.56	1.35	1.80	0.72
	32	115	1.77	1.50	1.64	1.42	1.85	0.74
	35	130	1.95	1.67	1.85	1.60	1.99	0.78
	37	140	2.07	1.77	1.99	1.72	2.08	0.80
	40	165	2.39	2.06	2.35	2.03	2.33	0.86

Figure 5.3 An example showing product attributes arranged in rows and columns. These can be represented in PCD by using the *Table Specification* specification type.

5.3.5 Sub-Product Class Specification

The *SubProductClass* specification type is in fact a complete product class. It is named, in order to distinguish between hierarchically defined product classes. For example a *Chair* product class can be made up of product classes such as *Caster*, and *Frame*. Here the *Caster* product class is a complete class in itself but is called a sub-product class in the context of the *Chair* product class. In the construction supply chain certain individual items, (even though they are part of a fuller or complete product) can be manufactured/supplied individually by different suppliers. These items can be supplied directly to contractors or to other intermediaries, who then assemble (by procuring different items from different suppliers) or value-add the items. The mechanism of assigning a product class to another product class helps the description of complex products whose individual parts or components may be supplied by different suppliers. This allows the description of products in an SD system, where information about certain product components is available from other suppliers or other external sources. *SubProductClass* specification types also enable product class reuse. For example a *Caster* product class can be used across a number of different *Chair* product classes. A sub product class can be made up of one or more specifications of each type. There is no limit on how many levels deep, a product class can be defined. For example a *Door Set* product class can be made up of *Door Closer*, *Door Lock*, and *Door Handle* sub-product classes, where each of these sub-product classes, (in the context of the *Door Set* product class), itself can be complex entities made up of a number of different specifications.

5.4 Versioning of Product Classes

New products or a new range of existing products are introduced by suppliers on a regular basis, as they enhance features and functionality. These changes to products cannot be defined within the scope of the existing product classes - as they cannot support these extra features. Hence, with the evolution of the products, the product classes need to evolve as well. It is therefore necessary to allow new versions of existing product classes to be created. A new version of product classes allows the product suppliers to describe new products with enhanced features.

Versioning of a product class, requires versioning of the different specifications, which a product class is made up of. As a product evolves through its lifecycle only some of its features, parts or components may change whilst other may remain unchanged in the new version. Hence only those specifications that support the extra features or changes need to be versioned. This allows reuse of existing specification versions together with the newly created specification versions, when designing new versions of existing product classes. Versioning of product classes not only facilitates rapid creation of new product classes, but also provides backward compatibility. For example a list specification *Wood Type* with version ID 1.0 can have a number of values such as *Alder*, *Cherry*, *Fir*, *Hemlock* and *Mahogany*. Once *Wood Type* version 1.0 list specification is created, it should remain persistent throughout its lifetime. There could be more than one *Door*, *Window* or other furniture equipment product classes using the *Wood Type* version 1.0, list specification as part of its composition. When a need is identified to expand the *Wood Type* list specification, a new version can be created with enhanced values.

5.5 AEC Industry Information Modelling/Management

Systems/Standards

In the AEC industry there are various modelling standards, languages, information reference models and data exchange formats available for information exchange between different industry actors. A modelling language provides syntax and semantics to create data representation of real world objects [Lui93]. The EXPRESS modelling language developed by the International Standards Organisation (ISO 10303 Part 11) [ISO94a], [Wil98] as part of the STEP (STandard for the Exchange of Product data)

5. The Product Classes

[ISO94], [Gle89] is based on an entity-attribute modelling approach and provides a rich set of facilities for defining complex data types [Nin97] [Lui93]. However this modelling language only supports creation of abstract data models as opposed to the implementation of specific data models which are created using a database design process [Nin97]. The IRMA (Information Reference Model for Architecture, Engineering, and Construction) [Lui93] is a framework for conceptual modelling and provides mechanisms for identifying relationships between entities such as products, activities, resources and participants in a building project via its modelling constructs. IRMA can be used to represent generic concepts and high level relationships between the entities. GARM (General AEC Reference Model) [Gie89] provides generic modelling concepts for defining product models in the AEC industry. The International Alliance for Interoperability (IAI) provides the AEC/FM (Facilities Management) community with a model for exchanging information among project participants throughout the lifecycle of a facility, via the Industry Foundation Classes (IFCs) [Wix98] and the aecXML [Wen01] schemas. IFCs provide data elements to represent parts of buildings, processes and relationships between them which can be used to create building models, that can be shared among project participants. aecXML supports B2B transactions over the internet via a common data format. A number of implementations of IFC specifications are also available from leading software companies to meet end user requirements.

The product class concept enables modelling of product attributes, so that product suppliers can provide product descriptions, using a standard and structured approach enables interoperability and enhances collaboration between different industry actors. The concept of product class is developed and enhanced in this research to support the data management needs of the PSCD application. Similarities and distinctions exist between the product class approach and some of the existing approaches identified in this section. However these similarities and distinctions are not explored in this research, because they do not fall within its scope. This research explores the creation of a new information sharing architecture (MDSSF) to address the procurement challenges, by providing support for various new distinctive features identified in Section 7.4.2. And in this research the MDSSF information sharing architecture has adopted product modelling techniques, based on the product class concept. Hence the

product class approach of modelling product attributes can be considered, as one of the competing approaches for product data management in comparison with other modelling standards, languages, information reference models and data exchange formats used in the AEC industry.

5.6 Chapter Conclusions

This chapter described product classes and the different types of specifications which have been developed to aid product suppliers create their product descriptions in their SD systems. Versioning support is developed to enable description of new products with enhanced features. A short summary of some of the existing product modelling/management approaches was provided to familiarise the reader with some of the competing information management approaches used in the AEC industry. This chapter provided a conceptual view of the product class concept. The next chapter describes how these concepts are implemented in the PCD system. In the next chapter we also show how the MDSSF architecture addresses the challenges identified in Section 1.2.

6. The MDSSF System Architecture

6.1 Introduction

The contributions to the PSCD application were mainly in its data management area and in the Grid-enablement of the application. The data management area were created to aid industry specification designers create standard mechanisms for storing product data (in the form of product classes) and to aid product suppliers who by using these standard mechanisms could manage product data within their databases. The need to adopt standard mechanisms, identified in Chapter 1, was to increase the collaboration between different industry actors, such as suppliers and contractors at the user level and to achieve interoperability at the system level. The research in this area aims to address the third challenge identified in Section 1.2.

This research also investigated how the advanced distributed computing facilities provided by Grid technology can be used in construction industry domain and to support procurement activities. This led to the creation of distributed database search services called MDSS (Multiple Database Search Service) which uses Grid middleware provided by the Globus toolkit, from the Globus project [Glo09] – the de facto standard for open source Grid computing infrastructure [Glo09a]. MDSS lies at the core of the PSCD application (see Figure 2.1) and provides scalability support, when accessing product data from several supplier databases. The research in this area aims to address the first and second challenges, identified in Section 1.2, i.e. to provide an integrated means of accessing a large number of supplier databases for up-to-date information about products available from external product suppliers. This allows information, such as product specifications, availability, delivery time and cost, to be taken into account in procurement planning.

Research efforts in the design and development of the data management and the Grid based database search components of PSCD identified this need, and led to the development of a novel federated information sharing architecture based on the federated database concepts called MDSSF - the main contribution this research. A distributed information sharing system/application cannot be realised without a well defined architecture that identifies its various system components. These components

6. The MDSSF System Architecture

provide necessary features and functionalities, and interact with each other to address user requirements. For example Sheth and Larson [She90] have proposed various FDBS architectures and identified various multi-DBMS/FDBS systems that correspond to these architectures. In this respect the components of the PSCD application (particularly the data management and database search components) are a manifestation of the MDSSF architecture. This chapter describes the novel architecture of MDSSF through its components. The proposed architecture is novel because it provides certain architectural features and functionalities to achieve information sharing between product suppliers and contractors, which address some of the construction industry's procurement challenges (see Chapter 1).

MDSSF's data sharing architecture brings together autonomous contractors and suppliers. Through the architecture product suppliers share their product data with the contractors. The architecture does not allow competing suppliers to see each other's data. The architecture is created by utilising features of FDBS architecture, such as distribution of data and autonomy of local database systems (DBS) [She90] and coupling them with Grid technology to provide scalability support. The federation model adopts a service oriented approach (using Web Services [Gra02] technology) to provide flexibility in retrieving data from the databases of several suppliers and share it with contractors. Section 6.2 describes the architecture of the data management components of MDSSF for product data management in the PSCD application. Section 6.3 describes the architecture of the Grid-enabled MDSS which provides a mechanism to perform a distributed database search in this Grid environment by invoking and searching databases of several product suppliers to retrieve the required product information. Section 6.4 provides a brief description of the software development process used to develop the components of MDSSF's architecture as part of the PSCD application. Section 6.5 presents the chapter conclusions.

6.2 Product Data Definition and Management in the PSCD Application

Product data definition and data management area of the PSCD is concerned with how the mechanisms of product data definition, its description and management can be achieved within the application and at the supplier's end. It incorporates an infrastructure comprising the Product Class Database (PCD) system, the Supplier's side

6. The MDSSF System Architecture

PCD (SPCD) system and the Supplier Database (SD) system. This section describes these system components at the architectural level. As part of system testing in Chapter 7 (see Section 7.2) we describe how a product class can be created and subscribed; and how product suppliers can describe product information in their SD systems by means of an example.

6.2.1 The Product Class Database (PCD) System

The product classes are created in the PCD system, which implements the product classes described in Chapter 5. PCD is a database centric tool which allows its users, the independent and knowledgeable industry specification designers to create different types of specifications, as part of creating a product class by using the interfaces and application modules of the PSCD front-end web application. The PSCD web application calls the stored procedures of the back-end PCD system¹ to perform the actual task of creating product classes and its specifications. These specifications provide a mechanism to define the different attributes a product may have. For example a product class corresponding to furniture equipment, such as a chair, can have a number of specifications, such as *width*, *height*, *weight*, *chair description* and *wood type* (in the case of a wooden chair). A product supplier, who supplies furniture equipment, can use these specifications to describe the product features by inputting the required product data or use the feature's default specification values as part of a product's data description in their database. Thus the PCD system stores information about product classes, product categories and product specifications to facilitate the description of actual products by product suppliers. It enables specification designers to create new product classes or new versions of existing product classes, when required. It supports the creation of different types of product classes to describe different types of products, with the aim that these product classes will be used by product suppliers to describe products in their databases. This will allow searching of these products by contractors. In MDSSF, it is a requirement that all product suppliers adhere to the schema of the PCD system to describe products in their databases. Hence, the schema of the PCD system acts as the common data model (CDM) of MDSSF.

¹ The PSCD web application was designed and implemented by other members of the COVITE team and is not a part of author's research.

6. The MDSSF System Architecture

The PCD system is implemented using the SQL Server 2000 RDBMS [Mic07a]. Figure 6.1 is the implementation level database diagram (logical database design) of the PCD system. In the figure entities are linked using relationship edges (lines), where the entity with a “key” endpoint denotes a parent entity and the entity with “figure-eight” endpoint is a child entity to establish parent-child relationships (i.e. one to many relationship). The PCD system provides several different stored procedures, which enable the creation of a product class and its various specification types. These stored procedures aid the web application modules of the PSCD application to perform database operations. All the data insert operations are handled by these stored procedures, because of the degree of complexity involved. The stored procedures help to maintain integrity, consistency and dependency of data in the PCD system. For example, creating a product class requires insertion of data into more than one table and therefore a specified order for inserting data is followed, to ensure that the PCD system’s integrity constraints are not violated. Section 7.2.2.1 demonstrates how a product class can be created in the PCD system. Some of the important features of the PCD system are:

- a) Create new product classes and specifications,
- b) Create new versions of existing product classes and specifications,
- c) Reuse existing product classes and specifications when creating new product classes,
and,
- d) Build complex product classes consisting of different types of specifications.

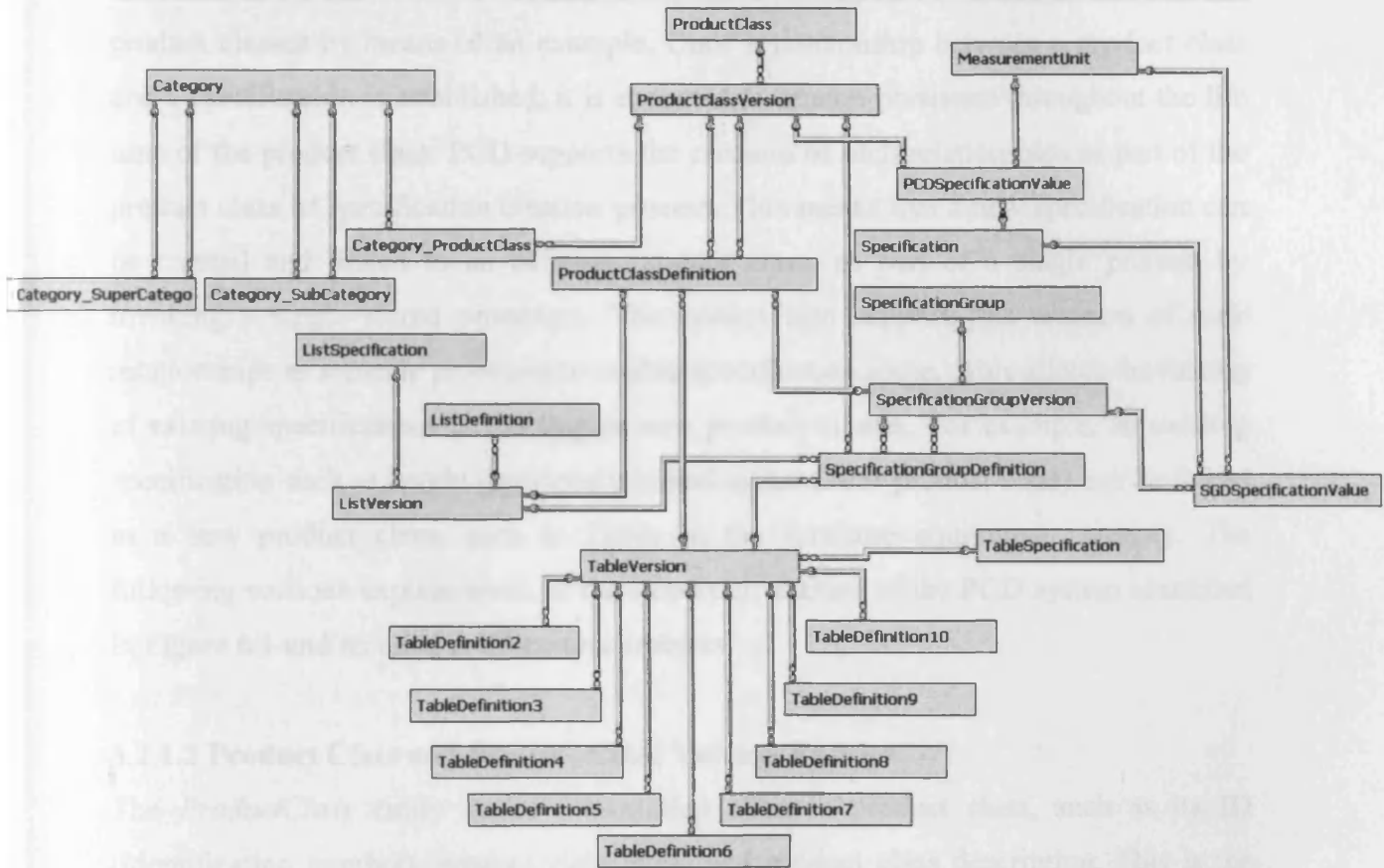


Figure 6.1 Database diagram of the PCD system

6.2.1.1 Modular Approach of PCD

In order to manage the complexity of describing a large number of product features via their specifications, the PCD system adopts a modular approach, which allows the creation of a product class independent of its specifications. This means a product class or its specifications can be created separately and in any order. For example a designer may wish to create a product class first, then create its specifications, and then link them together, or vice versa. A modular approach also provides the benefit of product class and specification reuse. A particular specification can be linked to more than one product class. Linking specifications in this way enables creation of complex product classes, which provides a way of describing different product features via specifications. The support for linking specifications is provided in PCD by using a number of stored procedures which perform specification assignment operations. This enables creation of a parent-child relationship between a product class and a specification. For example, a given version of a product class, such as a *Chair* can be assigned a unit specification called *height*. Section 7.2.2.2 describes assignment of product class specifications to

product classes by means of an example. Once a relationship between a product class and a specification is established, it is expected to remain persistent throughout the life time of the product class. PCD supports the creation of such relationships as part of the product class or specification creation process. This means that a new specification can be created and linked to an existing product class, as part of a single process by invoking a single stored procedure. The system also supports the creation of such relationships as separate processes to enable specification reuse. This allows the linking of existing specifications to existing or new product classes. For example, an existing specification such as *height* (previously linked to the *Chair* product class) can be linked to a new product class, such as *Table* in the furniture equipment category. The following sections explain some of the important entities of the PCD system identified in Figure 6.1 and its other architectural features.

6.2.1.2 Product Class and Product Class Version Entities

The *ProductClass* entity stores information about a product class, such as its ID (identification number), product class name and product class description. This is the highest level entity in PCD, which stores the standard name of a product. The name of a product class corresponds to the actual product name. Since it is the highest level entity in PCD, it only provides a general description of the product class. The specifications that a product class may have are therefore not directly linked to this standard product class entity. The specifications of a product class are linked to its specific versions and the information regarding product class versions is stored in the *ProductClassVersion* entity. This is because a particular class can have several versions (one-to-many relationship) and each of these versions may support different types of specifications. As described in Chapter 4, versioning of a product class enables its evolution, which allows product suppliers to create product descriptions with enhanced features. PCD provides stored procedures for creating new product classes and new versions of existing product classes. Creating a product class requires generation of a new product class ID and then insertion of values (specified by the specification designer) into the *ProductClass* and the *ProductClassVersion* tables. Every time a new product class is created, its default version (with version ID 1.0 or the ID specified by the designer) is automatically generated so that different types of specifications can be assigned to it.

6.2.1.3 Versioning Support in PCD

PCD provides a three level hierarchy to support the definition of a product class and the definition of its specifications. These definitions supported by the entities *ProductClassDefinition*, *SpecificationGroupDefinition*, *ListDefinition*, and *TableDefinition* (see Figure 6.2) identify in complete detail the individual specifications a particular product class version is composed of (in the case of a product class), and the individual sub-specifications a particular specification is composed of (in the case of a specification). In other words, the definition of a particular version of a product class identifies the versions of its specifications which are linked to it. The definition of a particular specification version identifies the sub-specification versions linked to it in greater detail. PCD supports the versioning of both the product class and its specifications. Hence in order to enable versioning of a product class and its specifications the three level hierarchy is introduced and implemented in PCD (Figure 6.2). This is a reduced version of the PCD database diagram identifying this hierarchy. As shown in the figure, a product class (*ProductClass* entity, level 1) can have several versions (*ProductClassVersion* entity, level 2) and each of these versions, via the *ProductClassDefinition* entity are linked to different specification versions (such as *ListVersion*, *SpecificationGroupVersion* and *TableVersion* entities, level 3). Hence a particular product class can have different versions, and each of these versions can support different types of specifications. The three level hierarchy also applies to all the specifications defined in PCD, except the lowest level unit specification (*UnitSpecification*) which does not require versioning support, as it only supports a single value when defined under a given product class. As illustrated in Figure 6.2, the *ListSpecification* entity is linked to the *ListVersion* entity, which is then linked to the *ListDefinition* entity. This also occurs for the *TableSpecification* and the *SpecificationGroup* specification entities. A sub-product class specification is not shown in Figure 6.2, because it is a complete product class assigned to another product class. Hence a sub-product class specification is a product class (i.e. a top level entity), which can have further specifications. A product class is called a sub-product class specification in the context of the product class to which it is assigned. For this reason there are two links (i.e. two parent-child relationships, see Figure 6.1) between the *ProductClassVersion* and the *ProductClassDefinition* entities. The first link identifies the definition of a particular version of a product class and the second identifies a sub-

product class specification (like any other specification) linked to the particular version of the product class. For similar reasons there are two links between the *SpecificationGroupVersion* and *SpecificationGroupDefinition* entities. These extra links (and other entities) are not shown in the database diagram (Figure 6.2) as it only presents a simplistic view of PCD, to illustrate the three level hierarchy mechanism. The three layer hierarchy implemented in PCD enables the creation of different product class versions and in this process provides a mechanism to support product class versioning. The system also supports assignment of different specification versions to product class versions to allow creation of complex product classes, which can manage different types of product features. Section 7.2.2.4 describes versioning support in MDSSF databases by means of an example.

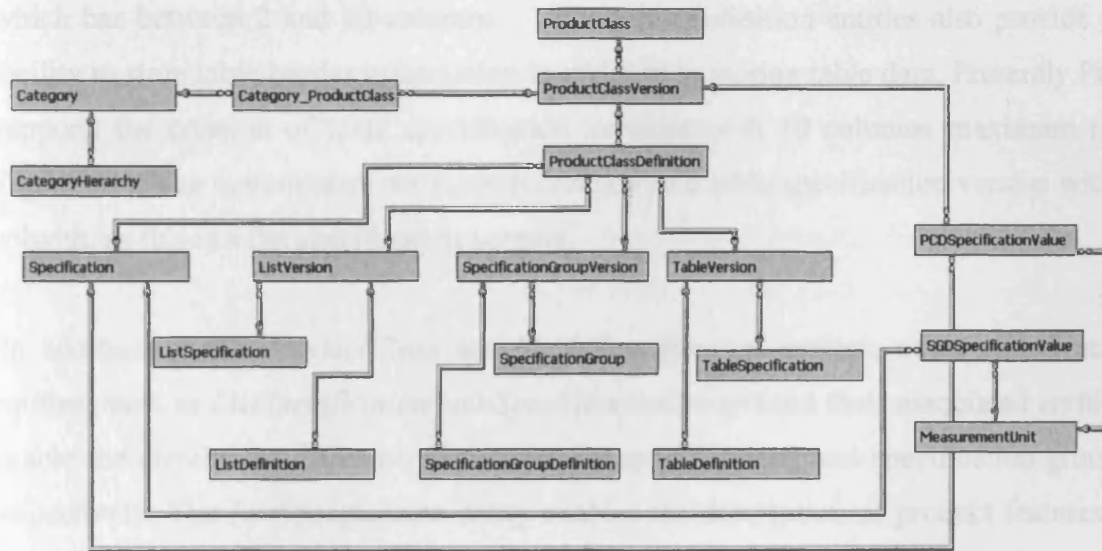


Figure 6.2 The reduced version of the PCD System identifying the three level hierarchy implemented in the system

6.2.1.4 Table Specification and Associated Entities

In PCD, the support for describing product features in a tabular format when product features can be organised in the form of rows and columns, is provided by the *TableSpecification*, *TableVersion* and a number of *TableDefinition* entities. PCD provides a mechanism for storing product data in a tabular format in pre-existing tables such as *TableDefinition2* to *TableDefinition10* (see Figure 6.1). For database administration reasons, individual database tables cannot be created every time a product data specification needs to be stored in tabular format. Giving users the

6. The MDSSF System Architecture

flexibility to create new database tables would also introduce different types of heterogeneities in PCD. Hence the concept of table specification is introduced in PCD with the aim of allowing product suppliers to describe product features (such as technical details) in the form of rows and columns, without creating actual database tables. For example if a specification designer requires a table specification with 5 columns, then the *TableDefinition5* entity (in addition to *TableSpecification* and *TableVersion* entities) is used to create the new version of the table specification having 5 columns. Thus all table specification versions which use 5 columns are defined using the *TableDefinition5* entity. This design is also important from an efficiency point of view. For example a large amount of tabular data for different table specification versions can be stored in these few pre-defined tables. In the table definition entity, each individual row is uniquely identified and links the row to its table specification version, which has between 2 and 10 columns. The table definition entities also provide the facility to store table header information in addition to storing table data. Presently PCD supports the creation of table specification versions with 10 columns maximum (see Figure 6.1). The system does not support creation of a table specification version with 1 column, as this is a list specification version.

In addition to the *ProductClass* and *TableSpecification* entities, other specification entities, such as *ListSpecification* and *SpecificationGroup* (and their associated entities) enable the creation of different versions of list specifications and specification groups, respectively. The *ListSpecification* entity enables the description of product features in the form of a list and the *SpecificationGroup* entity groups different product class specifications. The three level hierarchy is also implemented for these specifications to provide versioning support.

6.2.1.5 Default Values and Measurement Units

PCD also provides a mechanism which specifies default values for a specification. Providing default values allows the suppliers to readily select values available in PCD, in order to describe a particular product feature. For example list specification *Wood Type* can have a number of default values such as *Alder*, *Cherry*, *Fir*, *Hemlock* and *Mahogany*. One of these values can be chosen by a product supplier to describe the product feature of products such as furniture equipment. The *SGDSpecificationValue*

(Figure 6.1) entity stores the default values a unit specification may have, which are grouped as part of a given specification group version. Whereas the *PCDSpecificationValue* entity stores the default values a unit specification may have. This is defined for a given product class version. The default values of other specification types are stored in their respective specification definition entities. The PCD system also provides mechanism to create measurement units. This information is stored in the *MeasurementUnit* entity. Measurement units can be assigned to different specification types (these links are not shown in Figure 6.1 for clarity).

6.2.1.6 PCD Category Management

PCD also provides a category management system so that product classes can be categorised under different groups. For example, all furniture equipment related product classes can be categorised under the *Furniture* category. The system also supports hierarchical definition of categories by allowing the definition of sub-categories under existing categories. For example, sub-categories such as *Home Furniture* and *Office Furniture* can be created under *Furniture*. A product class can be assigned to a category or sub-category. Further, PCD also supports assignment of product classes to more than one category. This design allows, provision of links to a particular product class version from more than one category, which lets product data be accessed from more than one category. For example, certain types of furniture equipment can be used both in offices and homes. Although a given furniture equipment product class is defined only once in PCD, by providing more than one link it enables categorisation based on factors such as product features, usage domain and target audience.

In the same way as a product class can appear under different categories, a category can also be assigned to more than one category. For example, it is possible to assign the category *Office Furniture* to the category *Office*, which is already assigned to the category *Furniture*. This flexibility is offered by PCD with the aim of allowing specification designers and product suppliers to categorise product classes and categories in the most appropriate way using the factors identified above or other factors. Figure 6.1 identifies the category entities in PCD. The two entities *Category_SubCategory* and *Category_SuperCategory* with their associated stored procedures enable the creation or assignment of a category to more than one category.

6. The MDSSF System Architecture

The entity *Category_SubCategory* lists all the categories that are defined under a given or specified category and the entity *Category_SuperCategory* identifies the super categories that are available for a given category. A general description and category name of all the categories is stored in the *Category* entity. Each of the *Category_SubCategory* and *Category_SuperCategory* entities are linked twice with the *Category* entity. The first link identifies a given category and the second link the sub or super categories defined for the given category. The entity *Category_ProductClass* enables identification of categories under which a given product class appears.

6.2.1.7 The Other Features of the PCD System

PCD provides error notification support. If input values are not specified correctly or if database operations terminate abnormally, then the stored procedures raise error statements to notify users of the location, where the error occurred and the error code. The stored procedures also provide notification, if they execute successfully. Creating product classes or specifications requires inserting values in more than one table. The system provides transaction management support to ensure that all the database insert operations are carried out as part of a transaction so that if an error occurs the entire transaction is roll backed in order to avoid leaving the database in an inconsistent state. If all database insert operations are successful then the transaction is committed.

PCD also provides an ID generation system, which prefixes a three digit code to the IDs of product classes and specifications. This allows identification of whether an entity is a product class or a specification by analysing its ID. For example, the three digit code of a product class is 103. This functionality is used to assign specifications and ensure they are correctly assigned. For example, a product class can only be assigned to a category, specification group or another product class. If an attempt is made to assign a product class to any other specification, then an error is raised. The three digit code helps to identify which specification is being assigned to what specification and ensures that only correct assignments are made.

6.2.2 Supplier Product Data Management

The specification designers create product classes in PCD with the aim that these will be used by the product suppliers to manage product data at their side and so make the

6. The MDSSF System Architecture

data available to contractors via the MDSSF's VDD (Virtual Distributed Database), which provides a distributed database search facility (see Section 6.3). In order to manage product data at the supplier side an infrastructure, is required which allows product suppliers to access the product class data available locally and use this data, which provides specifications for describing actual product features. Thus, an important aim in developing product data management support at the supplier side, is to provide product suppliers with a readily available way to describe product data in a structured and standard way. In MDSSF, access to the product class data is provided to product suppliers using a subscription based approach, in which suppliers subscribe to relevant product classes. For example, product suppliers dealing with furniture equipment would subscribe to the furniture related product classes available in PCD.

6.2.2.1 SPCD and Subscription of Product Classes

In order to enable subscription to product classes, a data management infrastructure was developed for the product suppliers, which consists of two database systems. The first is called the Supplier's side Product Class Database (SPCD) system. This has a repository of product classes subscribed to by a product supplier. The features and functionality of SPCD system are similar to those of PCD. To a large extent the SPCD system can be considered as a replica of PCD, as PCD and SPCD provide the same entities, which are linked in the same way. However, distinctions can also be identified between PCD and SPCD. For example, the stored procedures in PCD are oriented towards the design of product classes, and it provides an extra set of stored procedures to assign different specifications to new or existing product classes to produce new product classes, or new versions of existing product classes. This feature is not available in SPCD, as it was mainly developed to support the need to manage product classes at the supplier's side, so that they can be used to create product descriptions. Hence, SPCD provides mechanisms for inserting product class data available from PCD as part of creating product classes locally at the supplier's side.

Using standard mechanisms to manage product data is one of the challenges addressed in this research. It is thus a requirement of MDSSF architecture that all product suppliers have databases, which are identical to the PCD system, i.e. they conform to the schema of PCD as this allows product classes to be downloaded. This can only

6. The MDSSF System Architecture

happen if the database systems have identical schemas to PCD. Since the schema of SPCD is identical to the schema of PCD, the product classes can be subscribed in SPCD. Subscribing a product class can be defined as the process of downloading the product class and specification values from the central PCD system and putting these in the local SPCD system, which is managed and controlled by an autonomous supplier. The subscription of product classes can be achieved using two steps:

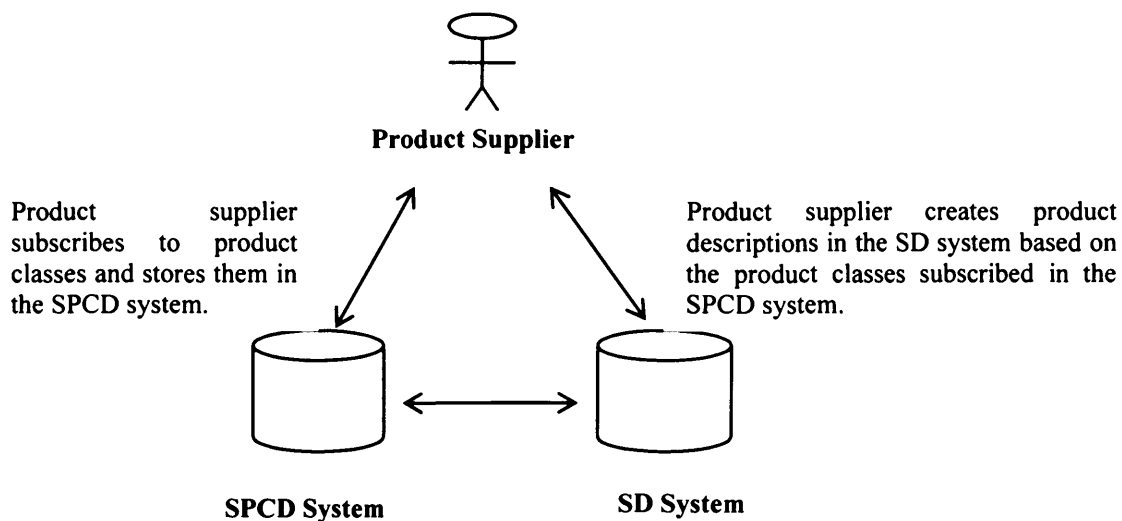
Step 1: Download the SPCD system into the local computing environment at the product supplier's side and install the system by following the SQL Server 2000 RDBMS database installation process [Vie00]. By using the functionality provided by the RDBMS, it is possible to convert the database into a file (by performing the database backup task) which can be downloaded by product suppliers. Once the file is downloaded the database can be restored to its original form by performing the database restoration task in the local computing environment. Since data in the PSCD application is managed using the SQL Server 2000 RDBMS, all the product suppliers must use this RDBMS to manage product data. This step needs to be performed only once, when a product suppliers joins MDSSF.

Step 2: Once SPCD is installed in the local environment, this step downloads the relevant product classes available from the PCD. At the data management level, downloading the product classes can be described as a process, where all a product class' relevant data is inserted into the SPCD system by using the stored procedures provided as part of SPCD. Hence, as in the PCD system, so in the SPCD system all the database insert operations are performed using the stored procedures. This maintains database consistency and integrity. In SPCD a number of stored procedures are provided which enable the creation of the product classes and their specifications which are subscribed by the product suppliers. This process is similar to the creation of product classes and specifications in PCD (see Section 6.2.1).

6.2.2.2 The SD System

The second database system developed as part of the product data management infrastructure at the product suppliers' end is called the Supplier Database (SD) system. This database stores the actual product data. Once a product class is downloaded into the SPCD system, product suppliers can enter descriptions of the features or specifications of their products into the SD system by using the product classes downloaded into the SPCD system. The relationship between the SPCD system and the SD system is illustrated in Figure 6.3. The need for a separate database system to store the product data occurs, because the SPCD system is designed to store only product classes and not the actual product data, which corresponds to these product classes. Thus SPCD identifies the specifications that a product can use, whereas the SD system uses these specifications to store actual product information. Product suppliers usually supply several products corresponding to a given product class. Although these products usually have the same features, they can still be differentiated on the basis of criteria such as colour, material used, dimension and weight. The SD system allows the description of several products corresponding to a given product class, where each product description can have different values from other descriptions. For example, there can be two chair products, corresponding to a *Chair* product class version, with each of the chair products possibly having different specification values for *dimension*, *weight* and *wood type*. Keeping the product information separate from its product classes also supports two important features namely the subscription of product classes and the description of products based on product classes, which are performed individually and independently.

6. The MDSSF System Architecture



Product data specifications in the SD system correspond to the corresponding product classes in the SPCD system.

Figure 6.3 Relationship between the SPCD and the SD systems at the product supplier end

The need for a separate database system to store the actual product information in the SD system was identified because unlike the PCD system, the SD system does not require a three level hierarchy to enable versioning support. Figure 6.4 is the implementation level database diagram of the SD System. Versioning support is implemented in SD using a two level hierarchy only as the products that are described in the SD system, correspond to a particular version of a given product class. The products cannot conform to a general top level product class because it has no specifications assigned to it (see Figures 6.1 and 6.2). Specifications are assigned to a particular version of a product class (see Section 6.2.1.2). Thus products conform to a particular version of a product class, which identifies different types of specification to describe its features. Hence in the two level hierarchy mechanism implemented in SD, the top level *Product* entity (See Figure 6.4), provides product information, such as product name and product description and is linked directly to the other specifications as shown in the figure. The specifications also support the two level hierarchy mechanism. As shown in the figure a product can have different unit specifications (stored in entity *ProductSpecification*) and other specifications (stored in entities such as *ListSpecification*, *SpecificationGroup*, *TableObject* and *Product* for sub-product specifications). These different specifications that a product has are linked to a product description (stored in *Product* entity) through the *ProductDefinition* entity to manage

6. The MDSSF System Architecture

different product feature values. Each of these specifications (as part of a two level hierarchy mechanism) is then linked to definition entities (such as *ListDefinition*, *SpecificationGroupDefinition* and a number of table definition entities) to provide versioning support and manage different product feature values for a given product. The unit specifications, which are stored in entity *ProductSpecification* are not versioned because these unit specifications store single values only. The unit specifications that are defined as part of a specification group are stored in the *SGSpecification* entity. The category management system implemented in SD is similar to the category management system of PCD (see Section 6.2.1.6).

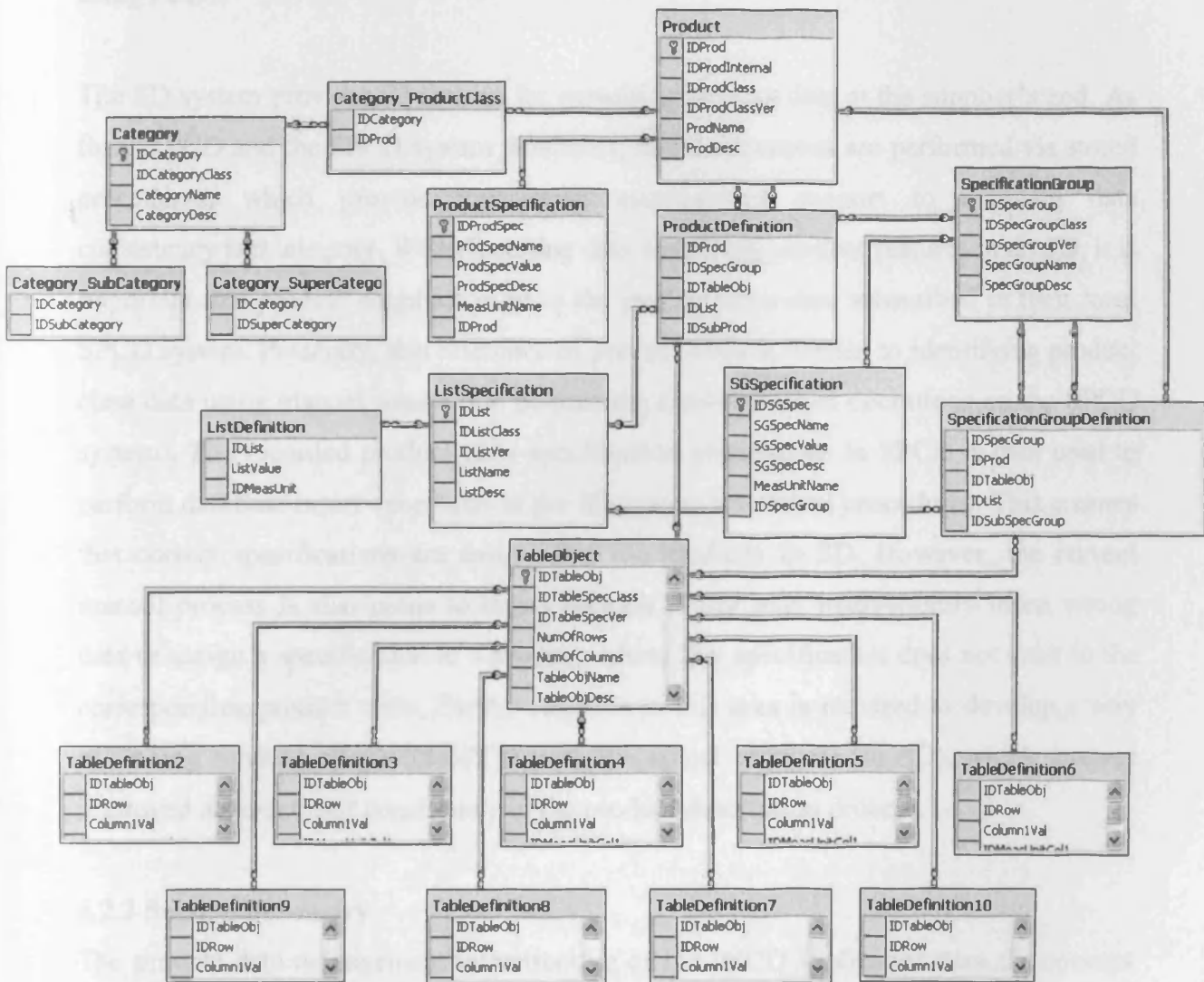


Figure 6.4 Database diagram of the SD System

Because of the two level hierarchy mechanism implemented in SD, the schema does not strictly correspond to the PCD schema. A number of version entities, such as

6. The MDSSF System Architecture

ListVersion, *TableVersion*, *SpecificationGroupVersion* and *ProductClassVersion* identified in Figure 6.1 and 6.2 appearing in the PCD and the SPCD systems, are missing in the SD system schema. In the SD system, the version information is provided in top level entities, such as *Product*, *ListSpecification*, *TableObject* and *SpecificationGroup*. For example, the attribute *IDProdClassVer* in the *Product* entity (see Figure 6.4) identifies the given version of a product class, which the product conforms to. Despite differences in the schemas, which is primarily a design decision, the SD system fully conforms to the product class concepts described in Chapter 4 and provides mechanisms to describe product features based on product classes created using PCD.

The SD system provides 22 entities for managing product data at the supplier's end. As for the PCD and the SPCD system databases, insert operations are performed via stored procedures, which provide transaction management support to maintain data consistency and integrity. When inserting data describing product features in an SD, it is important that product suppliers refer to the product class data subscribed in their local SPCD system. Presently, this reference of product class is limited to identifying product class data using manual means (i.e. performing database select operations on the SPCD system). The recorded product class specification information in SPCD is then used to perform database insert operations in the SD system via stored procedures. This ensures that correct specifications are assigned to the products in SD. However, the current manual process is also prone to errors because a user may inadvertently insert wrong data or assign a specification to a product, where this specification does not exist in the corresponding product class. Further research in this area is required to develop a way of linking product classes in SPCD with the actual products (in SD), which ensures improved accuracy and consistency in the product description process.

6.2.3 Section Summary

The product data management infrastructure of the PSCD application uses the concept of product classes as the underpinning way of managing product data using standard definitions which can evolve via the versioning support. As part of this research, three database systems were developed to support the product data management infrastructure. Approximately 10000 lines of code have been written to create this

6. The MDSSF System Architecture

infrastructure. As identified in Chapter 3, the UK construction industry is highly fragmented with large number of construction firms, most of which are small and medium scale enterprises. Many such enterprises still do not use IT systems for product data management, because of the complexity and cost, and make their data available using unstructured file formats such as PDF. It is anticipated that this research will provide these enterprises with a cost effective solution for managing their product data which also makes this data available to contractors using a distributed database search (see Section 6.3). Thus, standardisation plays a key role in the MDSSF architecture, and supports data sharing in a standard and integrated way. This addresses the third procurement challenge (see Section 1.2).

6.3 MDSS – a Virtual Distributed Database (VDD) of the MDSSF

In Chapter 1, an important aim of the COVITE project was identified, as investigating how advances in distributed computing and particularly in Grid computing can be used in the construction industry, to address the procurement challenges and improve the procurement processes [Mil02]. The first two procurement challenges (see Section 1.2) were addressed via the design and implementation of the MDSS (Multiple Database Search Service). The MDSS retrieves product data from a large number of SD systems in response to a contractor's query. These SD systems are autonomously managed by individual product suppliers. The MDSS provides a Grid based solution by utilising the Grid middleware Globus Toolkit [Fos98] 3.0.2 (Core) which is based on the Open Grid Services Architecture (OGSA) [Fos02].

Figure 6.5 provides a conceptual view of MDSSF. The Grid-enabled MDSS at the core (middle layer) enables the formation of a VDD to provide product data to contractors in the standard schema format of the PCD system. The VDD provides a single, integrated way of accessing product data which is made available by product suppliers through their SD systems. The VDD can be queried by contractors to find required product information. VDD is so named, because it does not store any product data, but provides such data when requested by a contractor by querying product supplier databases (SD systems) in real time. Hence VDD provides up-to-date information about products from external suppliers, so that the latest product specifications, availability and delivery time can be taken into account in procurement planning. The architecture of the Grid enabled

6. The MDSSF System Architecture

MDSS system is described here. However before describing the architecture in greater detail we identify the requirements of MDSS, the rationale for using Grid technology to support the MDSSF system components, and the benefits, MDSS provides to the PSCD application.

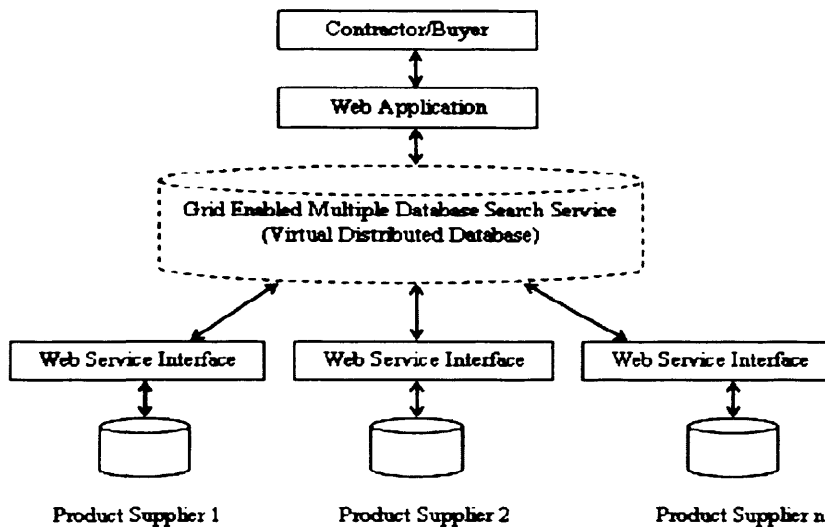


Figure 6.5 The conceptual view of the MDSSF

An important feature of the Grid is that it provides middleware to enable distributed computing in a particular domain to achieve high-end computational capabilities and high-throughput computing [Fos99]. In the PSCD application, the Grid support is provided to its MDSS system component to achieve system scalability². This is achieved by using computational resources to perform distributed database searches. Providing Grid support to the MDSS component is natural, because an important requirement of MDSS is to retrieve appropriate product data from a large number of supplier databases and support information sharing between contractors and suppliers in real time. There can be a large number of suppliers participating in an MDSS federation and a particular product can be supplied by several suppliers. Therefore, in response to a contractor's request, all the SD systems are searched that meet the current search criteria specified by the contractor. Thus by using Grid middleware in the MDSS system component the database access operations are Grid enabled so that a large number of

² The Grid support is also provided in the PSCD application to develop a secure access mechanism. The Globus toolkit [Fos98] for developing Grid enabled applications also provides tools to enable secure access to the resources in a Grid environment. The security features of the Globus toolkit were used by other members of the COVITE team to develop secure access to the PSCD web application. The security architecture of the application is explained in greater detail in [Joi04a], [Ran05].

6. The MDSSF System Architecture

these operations can be performed in real time to invoke a large number of SD systems to retrieve product data.

The Grid-enabled MDSS provides the benefit of integrated access to the large amount of product data available from several SD systems, through the use of a single system. It provides the scalability and computational capability that is required to access such information. The Grid-enabled MDSS also provides another important benefit as it enables the building of a large and comprehensive product data repository (the VDD). There is a high degree of probability that the information available in the VDD is accurate and up-to-date. This is due to the product data being decentralised and autonomously managed by the several product suppliers who are responsible for managing their own product data. It is anticipated that product information provided by product suppliers has the potential to influence procurement planning and procurement decision making by contractors. Thus, due to market forces and the need to compete with other product suppliers, who may also be participating in MDSSF and supplying similar or identical products, the product supplier will ensure accurate and up-to-date descriptions of their products in their SD.

In MDSSF, product suppliers share their product data with external users through the MDSS. The MDSS component of the MDSSF operates outside the boundary of supplier databases. By providing data access operations to the external MDSS system operating in the Grid environment, the data owners contribute to the analysis or processing of their product data by the Grid application. This benefits both the product suppliers and contractors. Product suppliers constantly are looking for new channels, which enable them to quickly disseminate product information to potential buyers as a reduction in the time to market is a competitive advantage [Mah04]. Hence by participating in MDSSF, product suppliers gain the opportunity to market their products through MDSS. This benefits contractors by providing a single, integrated means of accessing a large amount of product information from several supplier databases.

From a technical perspective, providing Grid technology support to the PSCD application at the middleware level (see Figure 6.5) also ensures that users of the application do not have to deal with the complexity of managing such an infrastructure.

6. The MDSSF System Architecture

The Grid support is provided outside the boundary of the autonomous SD systems, and therefore product suppliers do not have to deal with the Grid complexities. On the other hand, the PSCD application provides a browser based access to its users (contractors and specification designers) and so hides the complexities of the Grid infrastructure.

6.3.1 MDSS System Architecture

The MDSSF data sharing architecture brings together autonomous contractors and suppliers through MDSS. The name of the architecture (MDSSF) is derived from its MDSS system component, because MDSS plays a significant role in the overall architecture. In MDSSF, several SD systems are federated through MDSS. This provides a means for contractors to search a large number of supplier databases, and enables the creation of a federated information sharing system through its VDD. This section describes the MDSS system architecture, which was designed and implemented to operate in the Grid environment, and search for desired products based on search criteria submitted by contractors. Searching for products requires searching all supplier databases that have subscribed to the product classes corresponding to the products requested. MDSS queries a dynamic selection of relevant supplier databases to extract, in real time, information about the products, which the contractor wishes to acquire.

As identified above, the MDSS provides a Grid service solution for processing large amount of data by utilising the Grid middleware Globus Toolkit [Fos98] 3.0.2 (Core) which is based on the Open Grid Services Architecture (OGSA) [Fos02]. OGSA provides a set of standards, services and tools for building Grid services and supports an infrastructure to enable “*sharing and coordinated use of diverse resources in dynamic, distributed*” virtual organisations [Fos02]. OGSA is based upon OGSi (Open Grid Services Infrastructure) [Tue03] specifications “*which defines mechanism for creating, managing, and exchanging information among entities called Grid services*”. Grid services are created to perform a required functionality in the Grid environment. OGSA integrates Grid technologies with Web Service mechanisms, in order to create a distributed computing framework which is based on OGSi [Tue03] specifications. The Globus Toolkit version 3.0.2 is an implementation of the OGSi Version 1.0 [Tue03] specifications. The GT3 core component of the Globus Toolkit provides a set of system level services and a hosting environment for running the Grid services (i.e. a container

6. The MDSSF System Architecture

for Grid services) [Fer03]. The GT3 core framework can be deployed in the Apache Tomcat servlet container [Apa07], to expose the Grid services to a client and allow access via the XML based SOAP [Gra02] messaging protocol. In order to access Grid services via the SOAP protocol, the functionality of Apache AXIS [Apa07a], is also required as it is responsible for creating and exchanging SOAP messages over the HTTP protocol [Fer03]. The GT3 core framework, in addition to other external tools also provides Apache Axis system libraries. Once the Grid services are deployed in the hosting environment, access to these services is achieved using a factory mechanism which instantiates Grid services [Fer03]. Hence before Grid services can be accessed they must be instantiated to provide the required functionality to the client. Once these services are used, these instances have to be destroyed to free the system's resources (such as memory consumption and CPU) where they are deployed. An important advantage of the factory mechanism is that several of these Grid service instances can be created and each individual instance can be assigned to individual clients to meet their needs. The factory mechanism is used in MDSS to create individual Grid service instances which search supplier databases.

Figure 6.6 shows the architectural components of the MDSSF architecture. The two important components of MDSS are the MGS (Master Grid Service) and a number of DSSs (Database Search Services). These components are implemented using Java (Java 2 SDK 1.4.1_02) [Jav07]. The DSS components of MDSS are exposed as Grid services by deploying them in the GT3 hosting environment in the Apache Tomcat servlet engine. The MGS component is deployed as a Web Service for interoperability reasons. MGS is accessed by the PSCD web application (Figure 6.6), which is designed and implemented using the Microsoft .NET [Mic07] technology platform. The Globus toolkit does not provide client side Grid service access support for the .NET platform. Therefore MGS is deployed as a Web Service to allow its access from the PSCD web application modules via the SOAP messaging protocol. The components of MDSS are exposed as Grid services using the *createBottomUpGridService* tool [Glo09b], which is provided as part of the Globus Toolkit 3.0.2 (core) distribution, to generate Grid service support code [Glo09b]. The tool automatically generates the stubs, bindings, service locators, deployment descriptor, and an operation provider that delegates calls to Java

6. The MDSSF System Architecture

code which provides the actual Grid service functionality, i.e. the code of the DSS in this case.

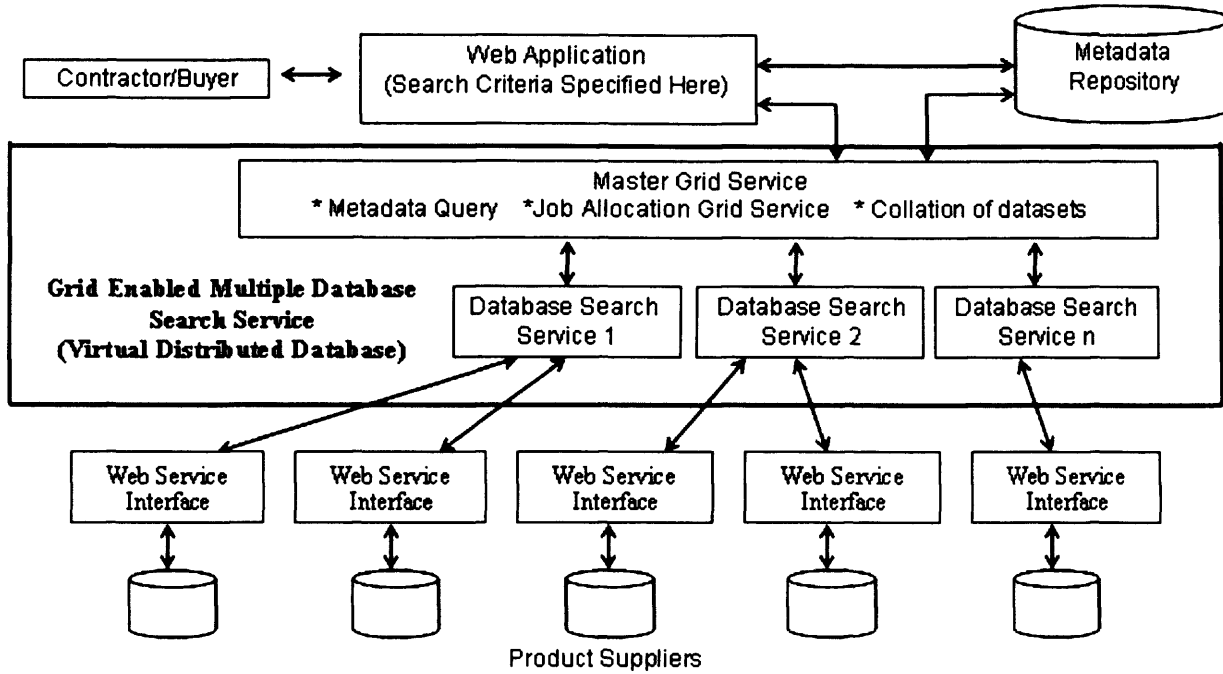


Figure 6.6 The MDSSF system architecture

6.3.2 Distributed Database Search Using MDSS

MDSS consists of two important components identified earlier as: the MGS and a number of DSSs. The MDSS Grid environment consists of a number of machines. One machine is used for the MGS Web Service and the other machines host DSS Grid services (see Figure 6.6). MGS is responsible for the overall coordination of DSS Grid services, which perform the searching of supplier databases. MGS divides the search task into roughly equal portions and allocates these to available DSS Grid service nodes which then work collaboratively to retrieve product data from several SD systems. The MGS also collates the result sets returned by these individual DSSs and sends the search result in a single XML document to the requesting contractor. In order to run a distributed database search in the Grid environment, MGS service is invoked by the PSCD web application with three input parameters. These are described in the following sections.

6.3.2.1 Search Criteria

MDSS searches for products based on the criteria submitted by a contractor or VO. The web application provides the search criteria as an XML document, which identifies the ID of the product class corresponding to which product information has to be retrieved from SD systems. Figure 6.7 shows an example search criteria which retrieves all products which conform to product class with ID 10325.

```
<?xml version="1.0"?>
<searchCriteria>
  <IDProdClass>10325</IDProdClass>
</searchCriteria>
```

Figure 6.7 An example search criteria identifying the ID of the product class.

6.3.2.2 Search Space

The search criteria specified in the input is analysed to identify the suppliers that meet the requirements of the contractor i.e. supply products corresponding to the product class identified in the search criteria. The search criteria is used for the dynamic creation of an XML document which contains the list of all the SD Systems that need to be searched in response to a user's request. For example a VO may be interested in searching for electric beds only. Therefore only those SD Systems that have subscribed to the *Electric Bed* product class need to be searched. This makes the search efficient as only those SD systems which are expected to provide the needed product information are invoked. The information as to which product suppliers have subscribed to product classes is stored in the metadata database. This information is used to create the second XML document (see Figure 6.8), which identifies the supplier databases to invoke via their Web Services. For example the *SupplierWS* XML element in the XML document in Figure 6.8 is the Web Service interface URL of an SD system.

```
<?xml version="1.0" ?>
<SupplierDataSet>
  <Supplier>
    <IDSupplier>55</IDSupplier>
    <SupplierWS>http://131.251.42.40/SupplierApp/ProductService.asmx</SupplierWS>
    <DataSetName>SupplierDataSet</DataSetName>
  </Supplier>
  <Supplier>
    <IDSupplier>132</IDSupplier>
    <SupplierWS>http://131.251.42.33/SupplierApp/ProductService.asmx</SupplierWS>
    <DataSetName>SupplierDataSet</DataSetName>
  </Supplier>
</SupplierDataSet>
```

Figure 6.8 A snapshot of the XML document identifying the product supplier databases to search

6.3.2.3 Available Grid Resources for a Search

The third important input parameter required is the identification of Grid resources available in the MDSS Grid environment. The search is performed using these. There can be more than one grid machine running the DSS Grid service in a distributed computing environment. Searching a large number of SD Systems typically takes place using a cluster of Grid machines, which work collaboratively and invoke supplier databases to retrieve product information in the form of XML documents. The Grid resources that are available for undertaking the search are dynamically identified and an XML document is created on the fly containing a list of Grid Service Handles (GSH). A GSH is a permanent network pointer to a particular Grid service instance [Tue03] which in the present case is the DSS Grid service. Figure 6.9 is an XML document identifying the Grid services that can be used to perform a search. This information is also obtained from the metadata repository. The *GSH* XML element in the XML document identifies the factory service's (*DatabaseSearchImplProviderFactoryService*) URL, which is used to create DSS Grid service instances to perform the actual search i.e. invoke SD systems by sending the search criteria identified, as part of the first parameter in Figure 6.7 (Section 6.3.2.1).

```
<?xml version="1.0"?>
<GridServiceHandle>
  <IDGsh>2</IDGsh>
  <GSH>
    http://131.251.47.110:18080/ogsa/services/services/uk1/co/activeplan/mdss/impl/
    DatabaseSearchImplProviderFactoryService
  </GSH>
  <MachineName>bouscat.cs.cf.ac.uk</MachineName>
</GridServiceHandle>
<GridServiceHandle>
  <IDGsh>3</IDGsh>
  <GSH>
    http://131.251.128.7:18080/ogsa/services/services/uk1/co/activeplan/mdss/impl/
    DatabaseSearchImplProviderFactoryService
  </GSH>
  <MachineName>agents-comsc.grid.cf.ac.uk</MachineName>
</GridServiceHandle>
```

Figure 6.9 XML document identifying a subset of available Grid services which can be used to perform distributed database search

In the next stage, MGS which is invoked by supplying three identified input parameters distributes the database search jobs to the available DSS Grid service nodes in the Grid cluster in roughly equal proportion. For example if there are 90 SD systems to search and there are 3 DSS Grid service nodes available then each DSS is allocated 30 SD systems to search. In order to distribute the search jobs equally in the available cluster, MGS parses the *GridServiceHandle* XML document (see Figure 6.9) and invokes each of the Grid service factory instances to create new DSS Grid service instances. To invoke operations on SD systems, only a single DSS Grid service instance per node has to be created for any number of database search operations at a given time. Hence, if a DSS Grid service is allocated the job of invoking 30 SD systems, then this job can be done by creating a single DSS Grid service instance only. This efficiency is achieved by specifying all the SD systems to be invoked in a single XML document (see Figure 6.8). The MGS divides the main document into several sub documents – one for each DSS Grid service instance.

A DSS instance is invoked by MGS by specifying two input parameters: a list of SD systems to search (which is a subset of the SD systems list identified in Figure 6.8) and the search criteria XML document (see Figure 6.7). The DSS instance then sends the search criteria to each SD system identified in the list by invoking it via its Web Service

6. The MDSSF System Architecture

interface. At the supplier side, the Web Service interface parses the XML search criteria, retrieves product data based on the criteria from the underlying SD and sends the results (product data) back to the DSS instance operating in the Grid environment as an XML document.

The division of a database search job into roughly equal portions and allocation of these to machines in the Grid cluster increases the efficiency of the search and reduces the overall search time as each of these machines perform its distributed database search job in parallel with the other machines. It also enhances the system scalability. In order to gain further efficiency the size of the Grid cluster can be increased by incorporating new nodes and installing the DSS component of the MDSS software to accommodate an increasing number of contractor's request or search an increasing number of SD systems. Figure 6.10 provides a snapshot of the product data returned from SD systems.

```
<NewDataSet>
  <Supplier IDSupplier="3"
    SupplierWsURL="http://131.251.42.40/SupplierApp/ProductService.asmx">
    <Product>
      <IDProdInternal>1143</IDProdInternal>
      <IDProdClass>10325</IDProdClass>
      <IDProdClassVer>1.0000</IDProdClassVer>
      <ProdName>Ampio</ProdName>
      <ProdDesc>General purpose electric bed model</ProdDesc>
    </Product>
  </Supplier>
  <Supplier IDSupplier="4"
    SupplierWsURL="http://131.251.42.33/SupplierApp/ProductService.asmx">
    <Product>
      <IDProdInternal>1144</IDProdInternal>
      <IDProdClass>10325</IDProdClass>
      <IDProdClassVer>1.0000</IDProdClassVer>
      <ProdName>Plano</ProdName>
      <ProdDesc>General purpose electric bed model, acute care</ProdDesc>
    </Product>
  </Supplier>
</NewDataSet>
```

Figure 6.10 A snapshot of the product data returned from an SD System

6.3.2.4 Data Aggregation

The data returned from the SD systems has to be aggregated, before it is finally sent to the web application. This aggregation takes place at two levels:

6. The MDSSF System Architecture

- a. At the DSS Grid service level which aggregates product data retrieved from several SD systems. This is then sent to the MGS (see Figure 6.6).
- b. The second aggregation takes place at the MGS Grid service level. The MGS which earlier divided the total database search jobs into equal proportions now aggregates the product data returned by each of the DSS Grid service instance.

MGS finally sends the collated product data retrieved from the relevant SD systems in response to a contractor's query to the PSCD web application where it is displayed.

6.3.3 SD Web Service Interface

A Web Service interface for SD was also implemented, so that product data can be retrieved in a platform neutral way. The Web Service interfaces are implemented using the .NET technology [Mic07], whereas the MDSS infrastructure is implemented in Java [Jav07]. The supplier side infrastructure uses .NET technology, as this allows easy integration of Web Service interfaces with the backend databases due to the common platform. However MDSS uses Java, because the open source GT3 core of the Globus toolkit and other associated tools are available as Java libraries. Further, the PSCD web application is also implemented using .NET technology. Thus, using SOAP based XML Web Services for communication between different system components in MDSSF is a sound approach due to the heterogeneous nature of the system components.

6.3.4 MGS and DSS Architectures

This section describes in more detail, the architectures of the MGS and DSS components.

6.3.4.1 MGS

Figure 6.11 is the UML class diagram of the MGS system component. MGS mainly allocates database search jobs to available DSS Grid service nodes and then collates the results. The diagram provides a high level view of some of the important functionalities of each class. Further details are available in the MGS system source code (see Appendix 5). The diagram shows the ten main classes. The *MasterGridImpl* class at the

6. The MDSSF System Architecture

middle of the diagram is one of the main classes and it implements the methods defined in the *MasterGrid* interface class. The *executeJob* method of this class is invoked by the PSCD web application via MGS's Web Service interface to perform a distributed database search operation. This method is provided with the three input parameters identified in Section 6.3.2. The second method (i.e. *callTestGridService()*) in *MasterGrid* interface is used to test whether a particular DSS Grid service instance is available in the MDSS Grid environment.

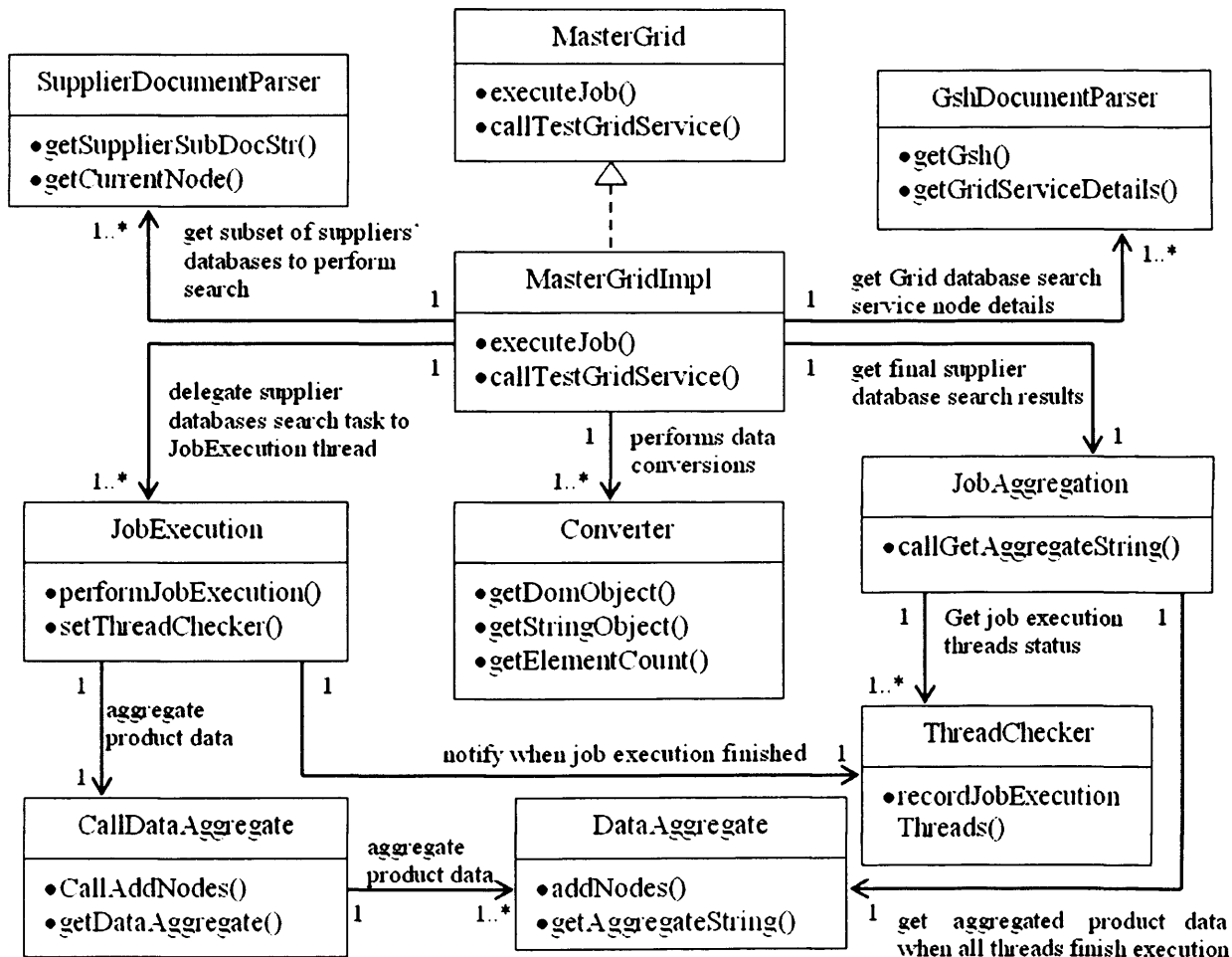


Figure 6.11 UML class diagram of the MGS system component of the MDSS

The *executeJob* method delegates supplier database search tasks to *JobExecution* class. It uses the services of the *SupplierDocumentParser* class to divide the total number of SD systems to search into roughly equal portions and allocates each portion to an available DSS instance. The *GshDocumentParser* parses the XML document to identify the DSS Grid service details, such as the GSH and other Grid service details, needed to

6. The MDSSF System Architecture

perform the search. This information (a sub-set of the SD system list and the DSS instance) along with the search criteria XML document is then passed to the *JobExecution* class by creating a new job execution thread. *JobExecution* class then invokes individual DSS instances operating in the Grid environment. The *executeJob* method of the *MasterGridImpl* class repeats the above process until all the DSS instances are allocated their portion of the database search jobs. This operation is performed as a multi-threaded process so that all the DSS instances can run in parallel in the Grid environment.

JobExecution class use the services of the *CallDataAggregate* class, so that search results returned by a single job execution thread can be collated by *DataAggregate* class. This operation is performed via the *CallDataAgregate* class so that access to the *DataAggregate* class can be synchronised. In this multi-thread environment more than one job execution thread may try to access the *DataAggregate* class at the same time. Therefore the *CallDataAggregate* class ensures that all access operations are synchronised and access is granted to one thread at a time. The responsibility of the *ThreadChecker* class is to track the progress of job execution threads and record the threads that have finished execution. This is important because the final result can be sent to the contractor only when all the threads have finished execution. For this purpose the job execution threads notify the *ThreadChecker* when they finish execution. The responsibility of the *JobAggregation* thread is to get the final result from the *DataAggregate* object. This is done only after all threads have finished execution. This information is obtained by the *JobAggregation* object from the *ThreadChecker* object. Finally the *MasterGridImpl* object (which originally initiated the search) gets the final result from the *JobAggregation* object which it sends to the PSCD web application. Objects of the *Converter* class are used to transform data representation i.e. convert the XML document to a string (for transportation), or DOM object [Dom05] (to parse XML document contents), or to identify the number of elements in a document. Facilities of the *Converter* class are also used by several other classes, but these links are not shown in the diagram.

6.3.4.2 DSS

Figure 6.12 is the UML class diagram of the DSS system component of MDSS. A number of DSS components have been deployed as Grid services, in the MDSS Grid environment. The DSS Grid service instances perform the actual work of invoking several SD systems to retrieve the product data. This diagram provides a high level view of some of the important functionalities of each class. Further details are available in the DSS system source code (see Appendix 5). The diagram shows the five main classes that comprise DSS.

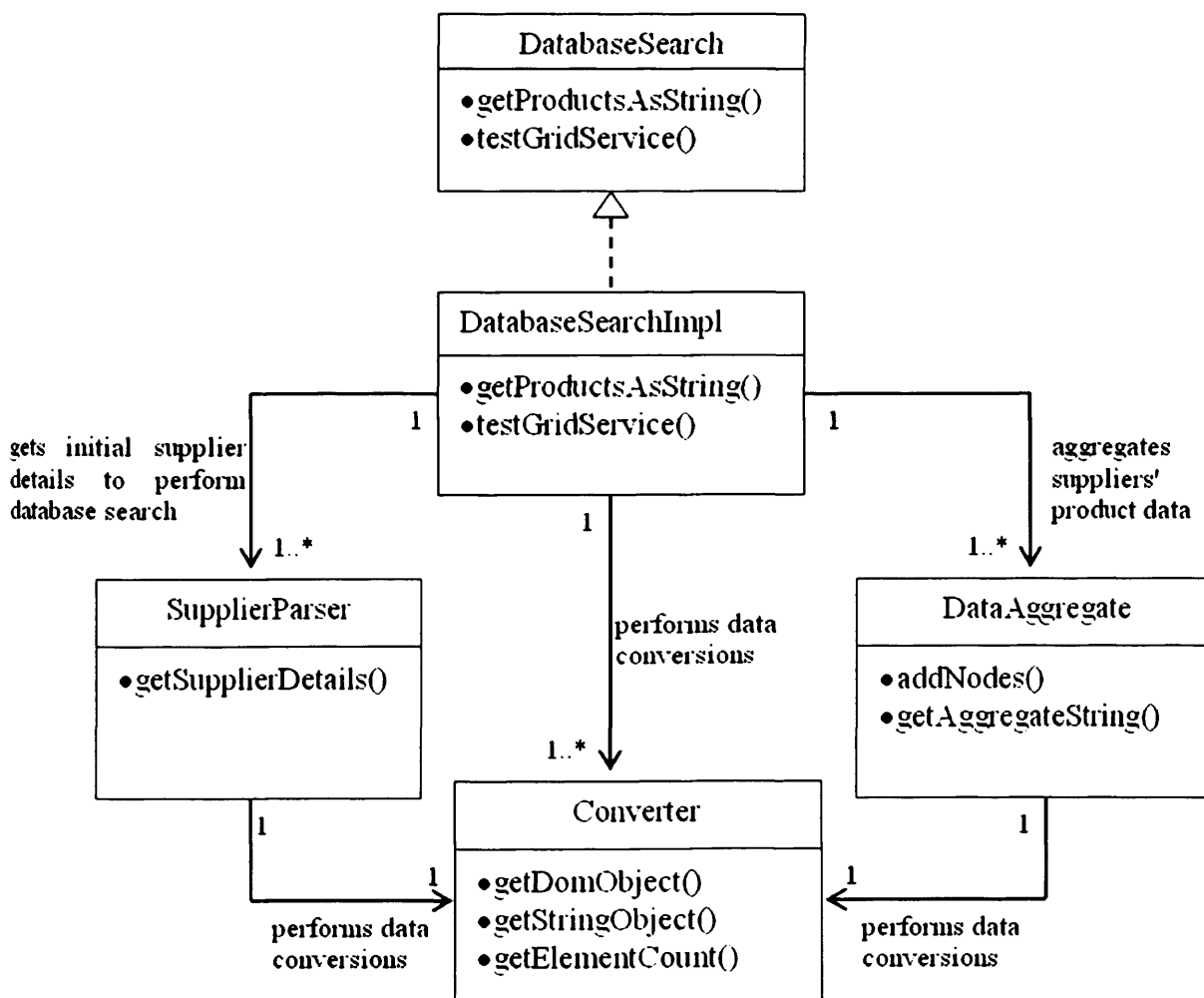


Fig 6.12 UML class diagram of the DSS system component of the MDSS

The *DatabaseSearchImpl* class at the middle of the diagram is one of the main classes of DSS and it implements the methods defined in the *DatabaseSearch* interface class. The *getProductsAsString* method of this class is invoked by the job execution thread of

6. The MDSSF System Architecture

the *JobExecution* class of MGS. For this purpose, the thread invokes the factory service (whose URL is identified by the GSH XML tag, see Figure 6.9) and instantiates the DSS Grid service instance. Once this instance is created the thread invokes the *getProductsAsString* method of the *DatabaseSearchImpl* object. This method is provided with two input parameters to identify the SD systems to search (Figure 6.8) and the search criteria (Figure 6.7). The second method of the *DatabaseSearchImpl* class is invoked by MGS to test the availability of a DSS Grid service instance in the MDSS Grid environment before the search job is submitted.

The *DatabaseSearchImpl* class use the services of the *SupplierParser* class to get SD system invocation details, such as the Web Service interface URL which is identified as part of the *SupplierWS* XML element of the supplier XML document (see Figure 6.8). The *DatabaseSearchImpl* object then invokes each of the SD systems identified in the supplier document and aggregates the results using the services of the *DataAggregate* class. A large number of XML product data nodes can be retrieved from several SD systems. However this depends upon the number of SD systems identified in the supplier document and the number of products in SD systems which correspond to the product class identified in the search criteria. Once all the product data is retrieved and aggregated it is sent to the calling *JobExecution* thread of MGS. The DSS also provides error management support by providing details of SD systems from where data could not be retrieved. This can occur if the URL supplied is incorrect, or malformed, or the SD system is not online.

6.3.5 Section Summary

This section described in detail the architecture of MDSS and how it performs distributed database search in a Grid environment. Data communication between the components of the MDSS and other PSCD application component such as the PSCD web application and the SD systems takes place using the XML-based SOAP messaging protocol, which allows flexible information sharing between the heterogeneous system components. MDSS provides mechanisms to achieve an integrated and up-to-date product data access from several product supplier databases with the aim of addressing the first and second procurement challenges identified in Section 1.2.

6.4 Software Development Process

A software development process must adopt a well defined methodology for software development which suits the nature of the software system being developed. A software development process incorporates a set of activities which are result oriented and lead to the development of a software product [Som01] meeting the identified requirements. A number of software process models have been identified by Sommerville [Som01] such as the waterfall model, evolutionary development, formal systems development and the re-use based development. The development and building of MDSSF system components was carried out as part of a research project, where different system functionalities were incrementally added during its development phases. Different system components were also developed simultaneously because of software dependency (i.e. the components of MDSS were dependent on the SD system to ensure that they worked properly) and in order to demonstrate the results achieved. Whilst developing a distributed system whose components operate in a heterogeneous environment, it is important to test the functionality of software components at each incremental stage to ensure they meet the technical and user requirements. Incremental addition of functionalities provides mechanisms to test software features and the interaction between heterogeneous system components. It identifies flaws or bugs on a functionality basis and addresses them early on and gives a sound basis for further development. An incremental software development process, which combines the features of the waterfall model of software development and the evolutionary approach [Som01] was adopted, as it suited the development of the MDSS components of the MDSSF architecture.

The database development methodology for the physical database design required a high degree of understanding of the target DBMS and the functionality it provides [Con05]. The data management infrastructure of the MDSSF architecture is designed and implemented using the SQL Server 2000 RDBMS. A sound understanding of the RDBMS and its database language called the T-SQL (Transact-SQL) which is an extension of the ANSI standard SQL language was gained, to ensure proper methods are followed for the database development. The databases are normalised to ensure little or no data redundancy or duplicate data. The stored procedures provide error management support to give information if errors are generated during data insert

6. The MDSSF System Architecture

operations and also to notify the user when stored procedures execute successfully. Transaction management support is also provided in stored procedures to ensure that data does not become inconsistent during error states.

6.5 Chapter Conclusions

This chapter described in detail the architecture of MDSSF through its components and its architectural features. The MDSSF not only provided a data sharing mechanism but a fully fledged infrastructure incorporating three different databases based on the well defined concept of product classes and Grid enabled distributed database search. The MDSSF architecture provides this infrastructure with the aim of serving data definition, data management and data search needs of the construction industry actors. This infrastructure was developed not only to support the PSCD application but also to address the wider procurement challenges (see Section 1.2) which the construction industry presently faces. The next chapter provides information on the test results of the MDSSF system components; highlights the distinctive features of the architecture which are part of its novelty and critically evaluates the architecture and its components.

7. System Testing, Verification and Validation of the MDSSF Architecture

7.1 Introduction

The new PSCD application is based on the MDSSF architecture and comprises several individual components. Each of these components was designed and implemented to provide needed functionality, whilst interacting with other system components. The three important functionalities provided by the components with respect to the procurement process are: data definition, data management and data search of construction industry product data. This chapter describes the testing and evaluation of the MDSSF system components.

The main contribution of this research is a novel Grid-based federated information sharing architecture. This architecture provides a novel mechanism of information sharing, which is generic but was designed for construction industry actors. The MDSSF architecture is verified and validated against the research objectives and its ability to address the procurement challenges. Whilst highlighting the distinctive and novel features, which MDSSF provides, we also highlight and assess the implications of its potential drawbacks to determine their effect. The chapter consists of: Section 7.2 - information on system testing of the prototype, Section 7.3 – a description of demonstration of the PSCD application based on the MDSSF and the interest it generated in external organisations, Section 7.4 - MDSSF system architecture is verified and validated against the research objectives, and Section 7.5 draws conclusions on the chapter.

7.2 System Testing of Prototype

In this research, the original PSCD application supplied by industrial partner APSL [Aps09] (see Section 1.3) is transformed into a new application utilising the advanced features of distributed computing provided by Grid technology through the MDSSF architecture. Information on its design and development is presented in Chapter 6 as part of the MDSSF system architecture. The principal aim of system testing is to test the ability of the components of the MDSSF architecture in meeting the procurement

7. System Testing, Verification and Validation of the MDSSF Architecture

challenges (see Section 1.2) and addressing the limitations of the original PSCD application (see Section 1.3). This is achieved by designing a testing strategy, consisting of 4 test objectives which test architectural components from different perspectives.

7.2.1 Test Objective 1

Test the ability of MDSSF architecture to function in a distributed computing environment and utilise the features of Grid technology.

The centralised database and the client/server based architecture of the original PSCD application cannot address the procurement challenges, because the domain is inherently distributed, consisting of a large number of independent, autonomous and far flung industry actors (see Section 1.3). The MDSSF is based on a distributed system architecture (see Figure 7.1) and its components function in a distributed computing environment to address this limitation. The main objective of this test is to set-up the MDSSF system components in a distributed computing environment, which utilises the features of Grid technology provided by Grid middleware. The technical details of setting up the MDSSF in a distributed environment are presented in full in Appendix 4. This is the first step to be performed as it gives a physical framework for the MDSSF system components and establishes collaboration between them. The other tests, such as performing a distributed database search in a Grid environment, are dependent upon the availability of this framework. To achieve the objectives of this test, several tasks were identified and undertaken as follows.

7. System Testing, Verification and Validation of the MDSSF Architecture

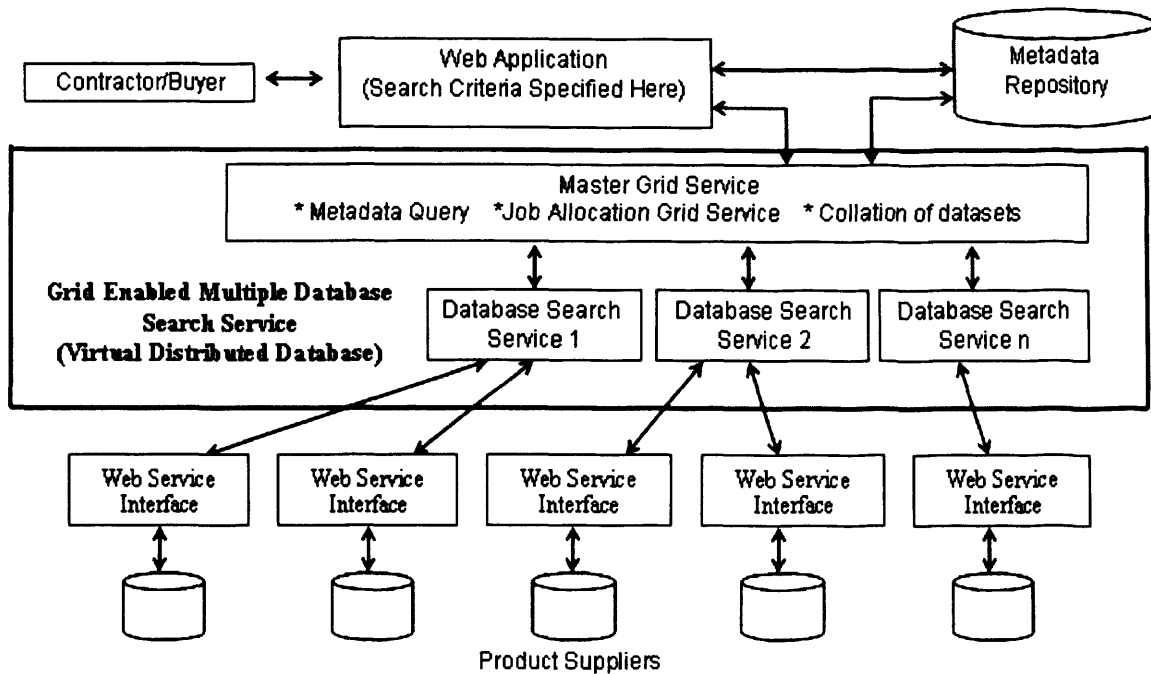


Figure 7.1 The MDSSF system architecture

7.2.1.1 Identification of machines

Setting up MDSSF in a distributed computing environment requires participation of several machines each running different system components. It is important in this test to install different system components on individual nodes to test domain specific requirements. For example, product suppliers operate autonomously, therefore supplier databases (SD systems) were installed in separate nodes. On the other hand, to test collaboration between different machines in a Grid environment, when searching product data from across several SD systems, it is important that the Grid middleware is installed in separate nodes. This enables machines in the Grid environment to perform the task of searching SD systems only in response to a contractor's query. Seven machines were made available to the project team for testing purpose. A list of these machines is identified in Table 7.1. Out of these 7 machines, machines 1 to 4 were in a local area network and the other 3 machines belonged to the Welsh e-Science Centre (WeSC) [Wel07] and access to them was provided by the centre for testing and development purposes.

7. System Testing, Verification and Validation of the MDSSF Architecture

1	tennis.cs.cf.ac.uk
2	cognac.cs.cf.ac.uk
3	violin.cs.cf.ac.uk
4	legend.cs.cf.ac.uk
5	arsenic.cs.cf.ac.uk (WeSC machine)
6	bouscat.cs.cf.ac.uk (WeSC machine)
7	agents-comsc.grid.cf.ac.uk (WeSC machine)

Table 7.1 List of machines which were used to set-up system components of the MDSSF architecture in a Distributed Environment

7.2.1.2 Installation of MDSSF System Components

Once the machines were identified the next step was to install MDSSF system components on them to set up MDSSF in a distributed computing environment. The MDSSF architecture consists of several system components. Figure 7.2 gives a list of these components and the machines on which they were installed. Each of these components is designed and implemented independently to provide the required functionality, whilst collaborating with other system components in a distributed environment. The MDSSF system components are dependent on prerequisite software tools which are also required to be installed. These software tools include the Grid middleware Globus Toolkit [Fos98] 3.0.2 (Core); hosting tools such as Apache Tomcat [Apa07] and Microsoft Internet Information Server (IIS) 6.0 [Iis09]; Apache Axis [Apa07a] for enabling collaboration between system components via the Web Services [Gra02] mechanism; and Microsoft SQL Server 2000 [Mic07a] RDBMS for creating product classes and managing product data. A full list of these software tools and their versions used in the MDSSF is provided in Appendix A4. After the installation of prerequisite software tools, the MDSSF system components were installed. This task was broadly split into two parts: set-up the Grid-enabled MDSS system component of the MDSSF to create a Grid environment; and set-up MDSSF database systems.

7. System Testing, Verification and Validation of the MDSSF Architecture

MDSSF Component	Installed in machine(s):
1. Database Search Service (DSS) System	bouscat.cs.cf.ac.uk, agents-comsc.grid.cf.ac.uk, violin.cs.cf.ac.uk, arsenic.cs.cf.ac.uk
2. Master Grid Service (MGS)	tennis.cs.cf.ac.uk
3. Product Class Database (PCD) System	tennis.cs.cf.ac.uk
4. Supplier Database (SD) System	tennis.cs.cf.ac.uk, cognac.cs.cf.ac.uk, legend.cs.cf.ac.uk
5. Supplier's Product Class Database (SPCD) System	tennis.cs.cf.ac.uk, cognac.cs.cf.ac.uk, legend.cs.cf.ac.uk
6. Supplier Web Service	tennis.cs.cf.ac.uk, cognac.cs.cf.ac.uk, legend.cs.cf.ac.uk
7. Metadata Repository	tennis.cs.cf.ac.uk

Figure 7.2 The MDSSF system components and the machines in which they were installed

7.2.1.2.1 Set-up Grid enabled MDSS

The MDSS performs distributed database search to retrieve product data from several SD systems in response to a contractor's query. It consists of two system components: the Master Grid Service (MGS) and the Database Search Service (DSS) systems. The Grid-enabled DSS System performs the task of searching and retrieving product data from several supplier database (SD) systems via their Web Service interface. It is installed in 4 machines as shown in Figure 7.2. The Grid middleware Globus Toolkit 3.0.2 (core) is also installed in these 4 machines. When installed on a machine, the Java system code of the DSS system is exposed as a Grid service, which allows the code to utilise the features of the Grid middleware (see Section 6.3). To create a Grid service, the DSS system code is run through the *createBottomUpGridService* tool which comes as part of the Globus Toolkit 3.0.2 (core) distribution. This tool generates the required utilities such as the stubs, the service locators, the deployment descriptor fragment, and an operation provider that delegates its calls to the DSS system code when the DSS Grid service is invoked for searching SD systems. Hence, the deploying of the DSS system on 4 machines and exposing them as Grid services by using the Grid middleware, enabled the creation of a Grid environment for performing the distributed database search of the SD systems. A DSS system can be referred to as a DSS Grid service, after

7. System Testing, Verification and Validation of the MDSSF Architecture

it is deployed in a node in the Grid environment. Once the DSS system is deployed in a node, it can be accessed via its Grid Service Handle (GSH), which is effectively its Web Service access URL. The GSH is actually the URL of the DSS factory service, using which a DSS Grid service instance can be created which then performs the operation of searching SD systems (see Section 6.3.2.3). Figure 7.3 shows the GSH of the DSS system deployed in the machine *bouscat.cs.cf.ac.uk* in the Grid environment. The SD systems are accessible to DSS Grid services via their Web Service interface, which is described using the WSDL (Web Service Description Language) [Gra02] (see Figure 7.1). The WSDL is an XML based format for describing network based services, which can be accessed via the SOAP protocol by exchanging SOAP messages between the services [Gra02].

Machine Name: <i>bouscat.cs.cf.ac.uk</i> Access URL (GSH): http://131.251.47.110:18080/ogsa/services/services/uk/co/activeplan/mdss_4/impl/DatabaseSearchImplProviderFactoryService
--

Figure 7.3. The Grid Service Handle (GSH) of a DSS Grid service deployed in machine *bouscat.cs.cf.ac.uk*

The DSS Grid services are dependent on a set of Java classes to create and send SOAP messages to SD systems and request product data. In a node where the DSS Grid service is deployed, these classes can be generated by *WSDL2Java* utility, which is available as part of the Apache Axis tool [Apa07a]. The classes are compiled and made available to the DSS Grid service for interacting with SD systems. The complete procedure of compiling and installing the DSS Systems in a Grid environment involves performing 14 steps. The technical details of the entire procedure and the steps involved are described in Appendix A4 (see Section A4.5.3). Every time a DSS system is deployed in a Grid environment, this procedure is followed. Therefore this procedure was followed 4 times to install the DSS system in the 4 machines of Figure 7.2.

The second component of the MDSS, i.e. the MGS, divides the total work to be done into roughly equal portions, and allocates each portion to individual Grid machines, running DSS instances (see Section 6.3.1). The MGS is deployed in machine *tennis.cs.cf.ac.uk* as a Web Service and is accessed by the PSCD web application to

7. System Testing, Verification and Validation of the MDSSF Architecture

submit product data search jobs (see Figure 7.1). The MGS then accesses DSS Grid service instances using their GSH to perform a distributed database search of SD systems in the Grid environment. The compiling and deploying of the MGS system code as a web service is a 7 step process which was performed when installing the MGS in machine *tennis.cs.cf.ac.uk*. The technical details of this entire procedure are described in Appendix 4 (see Section A4.5.4).

7.2.1.2.2 Set up MDSSF Databases and Supplier Web Service Interface

The second part of setting up MDSSF system components in a distributed environment involved installing its database systems. The MDSSF provides three important database systems: the PCD, SPCD and SD systems. The SPCD and SD systems and Supplier Web Service are deployed in three machines (see Figure 7.2). The database development for the three databases was performed on *tennis.cs.cf.ac.uk* using the SQL Server 2000 [Mic07a] RDBMS. The SQL Server Enterprise Manager utility of the SQL Server 2000 RDBMS allows backing of databases for easy installation of database in other machines. Thus, by using this utility the two supplier side database systems (i.e. the SPCD and SD Systems) were installed in the three machines and this procedure was followed once for each machine and for each database system. The PCD database is installed in machine *tennis.cs.cf.ac.uk*. The MDSSF also provides another database system called the Metadata Repository (see Figure 7.1). This database stores metadata about product suppliers and product classes subscribed by them. This information is helpful in identifying which supplier databases to search in response to a contractor's query (see Section 6.3.2). The Metadata Repository is also installed in machine *tennis.cs.cf.ac.uk*. The Web Service interface at the product supplier's end is developed using the Microsoft Visual Studio .NET 2003 [Vis08] development tool. This interface allows a DSS Grid service to perform database search operations on the underlying SD system to retrieve product data. The Web Service is hosted in the IIS server and describes the database search operation which can be performed on the underlying SD system through the WSDL (see Section 6.3.3).

The above described installing MDSSF system components and prerequisite software tools in different machines in a distributed environment. The complete technical details of the installation procedure are available in Appendix A4. These tasks were completed

7. System Testing, Verification and Validation of the MDSSF Architecture

successfully and were performed as part of the first test, to test the ability of system components of MDSSF architecture to function in a distributed computing environment and utilise the features of Grid technology.

7.2.2 Test Objective 2

Test the ability of MDSSF components to manage complex product information and support its evolution by using standard product definition mechanisms provided by product classes.

The products used in the construction industry are usually complex having a large number of different types of specifications. An important requirement of the MDSSF architecture is to allow product suppliers to manage complex product information through its databases. The architecture should provide standard means of product data management at a suppliers' end, and also provide mechanisms for product data evolution by providing versioning support. This is achieved in MDSSF through the concept of product classes. The MDSSF provides data management functionality through its three databases. These are the PCD, SPCD and SD systems. Section 6.2 explains in greater detail how these database systems were implemented and the features they provide to support the product data management infrastructure of the PSCD application. An important objective of creating this infrastructure is to support product data definition for managing complex product information. In MDSSF, product data definition provides a mechanism to create product classes in PCD, their subscription in SPCD and to create product information in SD by referring to the product classes subscribed in SPCD. Appendix 2 provides complete worked examples showing how product data definition, subscription and creation of product description is achieved in MDSSF. The following sections describe how the MDSSF system components were tested to provide required functionalities.

7.2.2.1 Creating a Product Class

The data definition and data management capabilities of MDSSF's databases were tested by using product data from real world products. This section describes one such product, which is a model of an electric bed product. Section A2.2 (see Appendix 2) provides a detailed description of this product and its different attributes. Information on

7. System Testing, Verification and Validation of the MDSSF Architecture

this product was provided to the COVITE team by its industrial partner APSL. When creating a product class, the first task is to represent product attributes in terms of product class specifications. This involves checking product attributes and identifying suitable specification types to represent them in a product class. For example a product attribute such as its weight can be represented by using a unit specification type called *Weight* in a product class. This allows the description of weight information for a product in a product class via its *Weight* specification. This research identified five specification types which enable description of complex product information in MDSSF databases (see Section 5.3). Section A2.3 shows how the electric bed product represented in the form of a product class uses these specification types to enable representation of product specification data. In Section A2.3, the attributes of the electric bed product have been categorised into different specifications of an *Electric Bed* product class, so that these can be represented in the PCD, SPCD and SD systems. This product shows the complexity of managing product data, as the *Electric Bed* product class has 42 unit specifications, 7 specification groups, 5 list specifications (out of which 2 list specifications have more than one version) and 2 sub-product class specifications. Thus it is a challenging product to represent and represents the more complex products.

Product classes are created in PCD. This support is provided in PCD through its stored procedures and functions which enable the creation of a product class and its different specifications. The PCD provides 55 different stored procedures and functions to support creation of product classes. Figure 7.4 shows an example of a stored procedure's execution code for the *proc_CreateNewProductClass* stored procedure. It specifies a list of input and output parameters. This stored procedure is used to create a new product class. The actual source code of this stored procedure is available in Appendix 5. This stored procedure enables the creation of the top level *Electric Bed* product class in PCD. After execution of the *proc_CreateNewProductClass*, no specifications exist for the *Electric Bed* product class in PCD. The modular approach adopted in MDSSF requires execution of further stored procedures to create different specifications which are assigned to the product class (see Section 6.2.1.1). Therefore, the creation of the *Electric Bed* product class and its different specifications (such as *Model*, *Where Marketed*, *Type*, *Patient Controls* and *Overall Dimensions* (see Section

7. System Testing, Verification and Validation of the MDSSF Architecture

A2.2)) led to the execution of 47 stored procedures. A selected list of the stored procedure execution code, their input and output parameters and execution results is presented in Section A2.4 (see Appendix 2)¹. In Figure 7.4 the input and output parameters of the stored procedure identify the name of the product class, its version, a general description of the product class and a description of a particular version of a product class. The *IDAssignToSpecTypeDef* input parameter identifies the ID of an entity to which a product class has to be assigned. In this case the value 1024 of this input parameter is the ID of a category called *Bed*. Whilst creating the *Electric Bed* product class, the stored procedure also assigns it to the *Bed* category. This functionality of assignment of entities is also provided to all the specifications of a product class.

```
declare @IDProdClassDef bigint
declare @IDProcState tinyint
declare @message nvarchar(500)
exec proc_CreateNewProductClass
@ProdClassName           = 'Electric Bed',
@IDProdClassVer          = 1.0,
@ProdClassDesc           = 'Electric Beds for hospitals',
@ProdClassVerDesc        = 'Electric Beds for hospitals version 1.0',
@IDAssignToSpecTypeDef   = 1024,
@IDProdClassDef           = @IDProdClassDef   OUTPUT,
@IDProcState             = @IDProcState      OUTPUT,
@Message                 = @message          OUTPUT

print 'IDProdClassDef: ' + cast(@IDProdClassDef as nvarchar)
print 'IDProcState: ' + cast(@IDProcState as nvarchar)
print @Message

Result of executing the stored procedure:
IDProdClassDef: 10526
IDProcState: 0
Product Class Created Successfully. Product Class ID is: 10325.
Product Class Version is: 1.00.
```

Figure 7.4 An example stored procedure execution code showing how to execute *proc_CreateNewProductClass* stored procedure in PCD and the result obtained after successful execution of the stored procedure.

¹ A full list of stored procedure execution code is not provided in Appendix 2 for brevity. The stored procedure execution code which is not shown in the appendix is similar to the code which is shown but executed with different input parameters to create different specifications for the product class. In the case of the SPCD and SD systems also, a selected list is provided for the same reason. However the source code of all the stored procedures in PCD, SPCD and SD systems is available in Appendix 5.

7. System Testing, Verification and Validation of the MDSSF Architecture

7.2.2.2 Assigning a Product Class

A product class is a complex entity having many different types of specification. Assignment of an entity (such as a unit specification called *Weight*) to another entity (such as a product class called *Electric Bed*) in PCD, SPCD and SD allows linking of different specifications to create a composite product class (see Sections 5.3 and 6.2.1.1). A product class can be assigned to entities such as a category (For example, *Bed*, see Figure 7.4), a sub product class or a specification group. The assignment feature can also be used to assign a specification to another specification. For example, a specification group called *Overall Dimensions* (see Section A2.3, Tables 23-26) can be created by assigning unit specifications such as *Length*, *Width* and *Height* to it. This specification group can then be assigned to a product class such as *Electric Bed*. This allows defining attributes of an electric bed, such as length, width and height in terms of a product class. The assignment feature of MDSSF database systems were tested as part of creating product class definitions and product descriptions. Sections A2.4, A2.5 and A2.6 (See Appendix 2) describe tests performed and output result which show how this feature is implemented at the database level.

7.2.2.3 Creating a New Specification

Figure 7.4 shows the result obtained by executing the *proc_CreateNewProductClass* stored procedure. It returns a value 10526 for output parameter *IDProductClassDef*. This ID identifies a particular version of a product class. In the present case the value 10526 is the ID of *Electric Bed* version 1.0 product class. When a new specification is created, it is linked to a product class version through its *IDProductClassDef* field. Figure 7.5 shows how a new specification called *Model* is created and assigned to the *Electric Bed* product class via the *IDAssignToSpecTypeDef* input parameter. The input value provided to this parameter is 10526. This value is the same as the value of the output parameter *IDProdClassDef* of *proc_CreateNewProductClass* in Figure 7.4. In this example the *Model* specification is being assigned to the *Electric Bed* version 1.0 product class, to enable identification of model information for a given electric bed product.

7. System Testing, Verification and Validation of the MDSSF Architecture

```
declare @IDProcState tinyint
declare @message nvarchar(500)
exec proc_CreateNewSpecification
@SpecName           = 'Model',
@IDAssignToSpecTypeDef = 10526,
@IDProcState        = @IDProcState OUTPUT,
@Message            = @Message OUTPUT
print 'IDProcState: ' + cast(@IDProcState as nvarchar)
print @Message
```

Result of executing the stored procedure:

```
'IDProcState: 0
Specification Created Successfully. Specification ID is: 1004.
```

Figure 7.5 The stored procedure execution code showing how a new specification can be created and assigned to a product class

7.2.2.4 Versioning Support in MDSSF Databases

The PCD and SD systems provide versioning support to enable creation of new versions of existing specifications. This allows specification reuse and evolution of product classes (see Section 5.4). The following example shows how the versioning support is tested in PCD as part of the creation of *Electric Bed* version 1.0 product class. Section A2.4.5 and A2.4.6 (see Appendix 2) shows stored procedure execution code to create two versions of a list specification called *Type*. In Section A2.4.5 the *Type* version 1.0 list specification is created and assigned to the *Electric Bed* version 1.0 product class to enable identification of the type information of electric bed. This list specification has two values: *General purpose* and *Acute care*. Thus an electric bed product which conforms to *Electric Bed* version 1.0 product class can be of type general purpose and/or acute care. The model of the electric bed product used in this research uses the ‘Type’ attribute to describe patient controls also (See Section A2.2). Therefore a new version of the *Type* list specification (i.e. *Type* version 2.0) is created and assigned to *Patient Controls* version 1.0 specification group. This specification group is also assigned to *Electric Bed* version 1.0 product class (see Figure 7.6). The list specification *Type* version 2.0 has three values: *Pendant*, *Hand control box*, and *Handset (Nurse controls mounted at foot end)*. This shows that the electric bed product which conforms to *Electric Bed* version 1.0 product class has three different types of patient controls. A snapshot of the *Electric Bed* version 1.0 product class showing the two versions of list specification *Type*, their list values and the entities to which they are assigned is presented in Figure 7.6. In this example, the list specification *Type* is created only once and it is extended through the versioning support by creating new versions to describe

7. System Testing, Verification and Validation of the MDSSF Architecture

different product attributes. When a need is identified in future, another version of the list specification such as *Type* version 3.0 can be created and assigned to describe additional features of the product class as part of its evolution. This example shows how the versioning support enables specification reuse to enable description of different attributes of electric bed product. Versioning support is also provided to other specification types in PCD to support product class evolution (see Section 5.4).

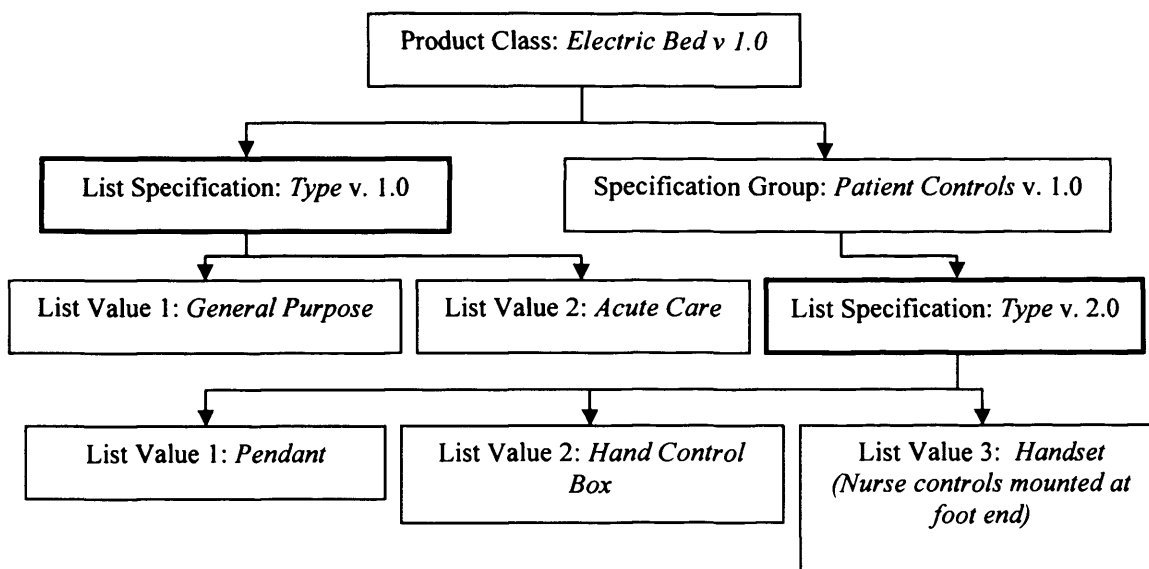


Figure 7.6 A snapshot of *Electric Bed version 1.0* product class showing two versions of *Type* List Specification (in bold boxes), their list values and the entities to which they are assigned.

In the above discussion we have shown the capability of PCD to create a new product class, a specification and assignment of the product class to a category and assignment of the specification to the product class. In the way a specification is assigned to a product class, other types of specifications such as *ListSpecification*, *SpecificationGroup* and *SubProductClassSpecification* can also be assigned to it to enable definition of the different types of attributes a product may have in terms of product class specifications. The examples in Section A2.4 (see Appendix 2) provides test results showing how these specification types have been used when creating the *Electric Bed* version 1.0 product class. This discussion has also shown how versioning support is implemented in PCD.

7. System Testing, Verification and Validation of the MDSSF Architecture

7.2.2.5 Product Class Subscription in SPCD

The SPCD and SD are used for product data management at a supplier end. These two databases are subscribed by product suppliers and managed locally. A product supplier then subscribes to relevant product classes in SPCD, so that products corresponding to these can be created in SD. Product classes, which are created in PCD are subscribed in SPCD by populating it with product class data (see Section 6.2.2.1). The SPCD provides its functionality through 28 stored procedures and functions. The SPCD system is installed in three machines in the MDSSF distributed computing environment (See Figure 7.2). Several tests were conducted to subscribe different product classes in the SPCD. Section A2.5 (see Appendix 2) provides a list of stored procedures that were executed to subscribe one such product class, *Electric Bed Version 1.0*, in an SPCD system installed in machine *tennis.cs.cf.ac.uk*. A total of 47 stored procedures were executed to subscribe this product class and its different specifications in the SPCD system. However, only a selected number of these stored procedures are provided in the appendix for brevity. Since the SPCD system is installed in 3 machines, the same steps were performed to subscribe the *Electric Bed Version 1.0* in the other two machines also.

7.2.2.6 Creation of Product Information in SD

The Supplier Database (SD) system stores actual product information. The SD system provides its functionality through 39 different stored procedures and functions. The functionality of this database was tested by installing it in three machines in the MDSSF distributed computing environment (See Figure 7.2). Product suppliers can manage product information in SD by using product classes downloaded in SPCD. This functionality of the SD system was tested by creating product information. The example in Section A2.6 (see Appendix 2) shows *ETESMI/Plano* model of an electric bed product created in an SD system which is installed in the machine *tennis.cs.cf.ac.uk*. In SD, the stored procedure *proc_CreateNewProduct* created a top level product description for the electric bed product, which referred to the *Electric Bed* version 1.0 product class in SPCD via *IDProdClass* and *IDProdClassVer* input parameters (see Section A2.6.1). Specifications to this electric bed product were assigned by executing stored procedures, which created its specifications. Section A2.6 shows how different types of specifications were created and assigned to the electric bed product. A total of

7. System Testing, Verification and Validation of the MDSSF Architecture

47 stored procedures were executed in an SD to create a version of an electric bed product and its different specifications in the testing process. However, only a selected number of these stored procedure execution code is provided in Section A2.6 for brevity. The appendix provides further implementation level description of the creation of this product. In the testing, eight different models of electric bed products (including the *ETESMI/Plano* model demonstrated here) were created for SD systems installed in the 3 machines². These different models conformed to the *Electric Bed* version 1.0 product class, but used different specification values for specifications defined in the product class. These different models of electric bed product in different SDs demonstrate that when a product search takes place, information on the same or similar products, can be provided by more than one product supplier.

For testing purposes, in addition to the *Electric Bed* product class, several other product classes were also created in the PCD. These product classes were created to describe furniture related products such as doors, windows and chairs. In addition the product classes were also created for industrial products such as fan coil units. The product classes were created for the *Cheetah* range of fan coil units [Che08]. Information on this product was also provided by industrial partner, APSL. The products for which products classes were created in PCD and their description in SD, ranged from simpler products having a small number of specifications to more complex products having a large number of different types of specification. The *Electric Bed* product class, which has a large number of specifications, shows the capability of MDSSF databases to handle complex product information.

7.2.3 Test Objective 3

Test the ability of MDSSF components to provide an integrated means of viewing product information available from several different suppliers.

The functionality of searching SD systems and aggregating product data retrieved from several SD systems is provided in MDSSF by its MDSS system component. This aggregated product data provides an integrated view of product information available from several different suppliers. The MDSS system is a middleware component in the

² Out of these eight electric bed products, four products were created by author's colleague Pete Burnap who was also a member of COVITE project team and rest four were created by the author.

7. System Testing, Verification and Validation of the MDSSF Architecture

MDSSF architecture and is made up of two sub-components: the Master Grid Service (MGS) and a number of Database Search Services (DSS) operating in a Grid environment (see Figure 7.1). These two components of the MDSS enable the creation of a Virtual Distributed Database (VDD) of product information where the information is provided by several product suppliers. Section 6.3 describes the architecture of MDSS and the rationale for providing Grid technology support to it in greater detail. As part of this testing objective, the MDSS sub-components were tested individually and then together as a single system to provide an integrated view of product information, available from several SD systems. Several tests were conducted to test the ability of MDSS to provide the needed functionality.

7.2.3.1 Testing MGS

The MGS is tested for its ability to:

a) Allocate database search jobs to DSS nodes in a Grid environment

The MGS divides the total number of supplier databases (SD) to search, into roughly equal portions and allocates a portion (i.e. a sub-set of the total SDs to search) to each of the DSS nodes available in a Grid environment. As part of this testing, the MGS invoked DSS nodes by providing them with a sub-set of total SDs to search and search criteria. In the MGS, *MasterGridImpl* class (See Figure 6.11) provides the functionality of allocating database search jobs to DSS nodes. This class was successfully run on the MGS node (i.e. machine *tennis.cs.cf.ac.uk*). The method *executeJob* of this class (via its Web Service access URL) is invoked by the PSCD front-end web application to perform a distributed database search of SD systems. The method is invoked by the front-end application providing three arguments in XML document format. The first argument *searchString* identifies the product search criteria. Figure 7.7 shows an example search criteria, which retrieves all products that conform to a product class with ID 10325. This is the ID of the product class *Electric Bed* version 1.0 (see Figure 7.4). The second argument *supplierString* is the list of supplier databases to search (see Figure 7.8) and the third argument *gshString* is a list of Grid Service Handle (GSH) URLs, which uniquely identify each DSS node available in a Grid environment (see Figure 7.9). With the above three arguments and by using the functionality provided by the other classes of the MGS, the *executeJob* method initiates database search jobs. The

7. System Testing, Verification and Validation of the MDSSF Architecture

MGS provides multi-threading support, so that DSS nodes can perform their operation of searching supplier databases in parallel. This functionality was also tested as part of the *executeJob* method which creates a separate *jobExecution* thread for each DSS node available in the Grid environment (see Figures 7.10 and 7.11). The *jobExecution* thread is an instance of a *JobExecution* class and its role is to submit a database search job to a DSS node and collect product data returned by it. The *executeJob* method provides *jobExecution* threads with the product search criteria, a subset of the supplier databases to search and one GSH URL. A single *jobExecution* thread invokes one DSS node deployed in a separate machine through its GSH URL. Since several *jobExecution* threads are run in parallel, several DSS instances running in their individual machines are invoked in a Grid environment, to collaboratively perform database search jobs by invoking several SD systems. As part of the testing process the *executeJob* method created four *jobExecution* threads and each of these threads in turn invoked four DSS nodes in parallel. This process is shown in Figure 7.10.

```
<?xml version="1.0"?>
<searchCriteria>
  <IDProdClass>10325</IDProdClass>
</searchCriteria>
```

Figure 7.7 An example search criteria identifying the ID of the product class.

```
<?xml version="1.0" ?>
<SupplierDataSet>
  <Supplier>
    <IDSupplier>55</IDSupplier>
    <SupplierWS>http://131.251.42.40/SupplierApp/ProductService.asmx</SupplierWS>
    <DataSetName>SupplierDataSet</DataSetName>
  </Supplier>
  <Supplier>
    <IDSupplier>132</IDSupplier>
    <SupplierWS>http://131.251.42.33/SupplierApp/ProductService.asmx</SupplierWS>
    <DataSetName>SupplierDataSet</DataSetName>
  </Supplier>
</SupplierDataSet>
```

Figure 7.8 A snapshot of *supplierString* XML document identifying the product supplier databases (SD systems) to search.

7. System Testing, Verification and Validation of the MDSSF Architecture

```
<?xml version="1.0"?>
<GridServiceHandle>
  <IDGsh>2</IDGsh>
  <GSH>
    http://131.251.47.110:18080/ogsa/services/services/uk1/co/activeplan/mdss/impl/
    DatabaseSearchImplProviderFactoryService
  </GSH>
  <MachineName>bouscat.cs.cf.ac.uk</MachineName>
</GridServiceHandle>
<GridServiceHandle>
  <IDGsh>3</IDGsh>
  <GSH>
    http://131.251.128.7:18080/ogsa/services/services/uk1/co/activeplan/mdss/impl/
    DatabaseSearchImplProviderFactoryService
  </GSH>
  <MachineName>agents-comsc.grid.cf.ac.uk</MachineName>
</GridServiceHandle>
```

Figure 7.9 A snapshot of XML document identifying a subset of available Grid services via their GSH which can be used to perform distributed database search.

7. System Testing, Verification and Validation of the MDSSF Architecture

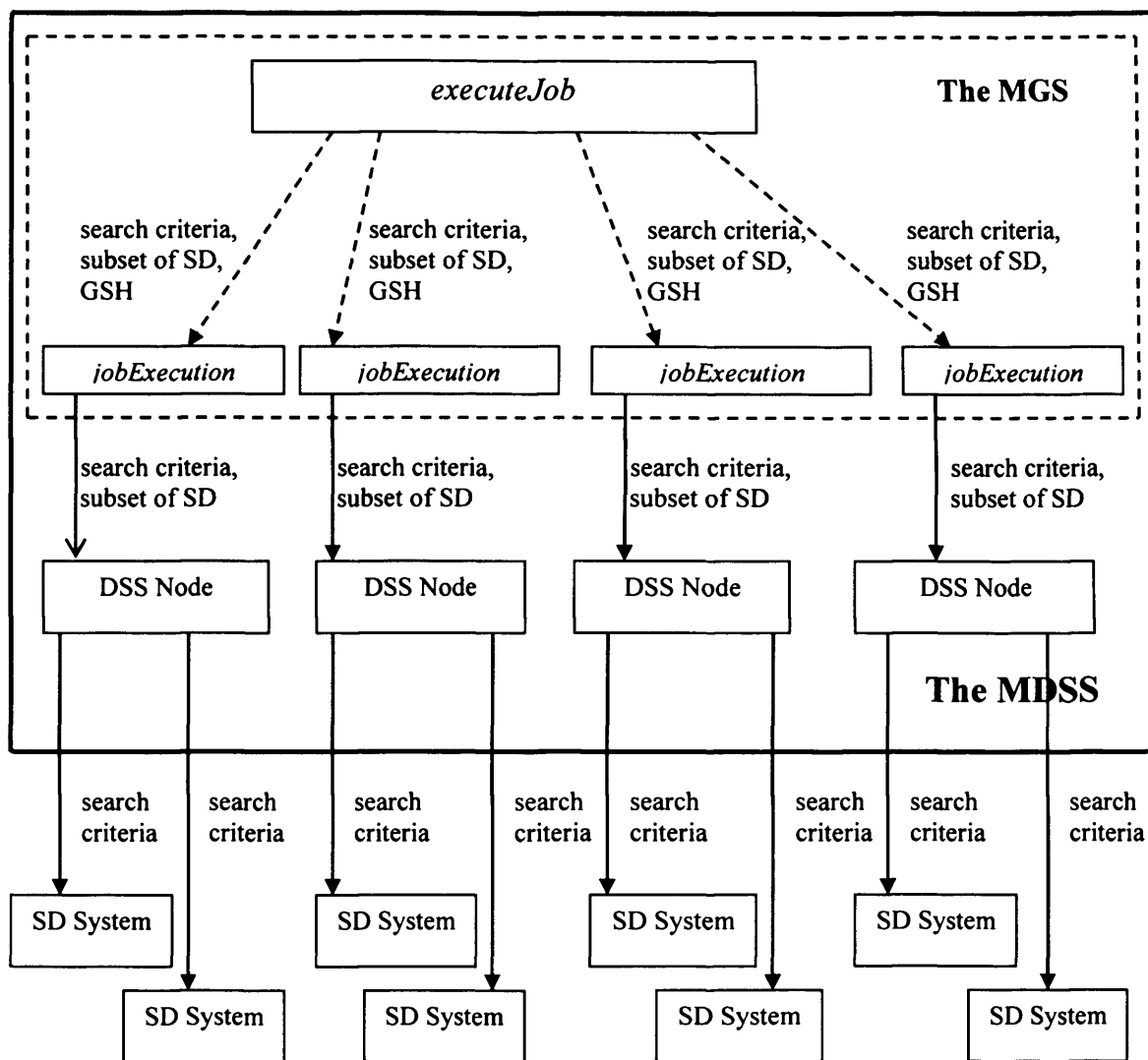


Figure 7.10 This figure shows how the *executeJob* method of the *MasterGridImpl* class in the MGS distributes database search jobs to DSS nodes. The *executeJob* method creates four *jobExecution* threads (of class *JobExecution*) and provides each thread with the search criteria and a subset of SD systems to search, and a GSH of DSS node. These *jobExecution* threads then invoke DSS nodes (via their GSH) and provide them with the same information (i.e. search criteria and subset of SD systems). Invoking DSS nodes simultaneously as part of a multi-threaded operation enables them to run in parallel in the Grid environment to collaboratively perform distributed database search. The DSS nodes then invoke SD systems to retrieve product data by providing them with the search criteria. A DSS node invokes several SD systems. The DSS nodes invoke SD systems via their Web Service interface (not shown in the figure for brevity. This is shown in figure 7.1).

7. System Testing, Verification and Validation of the MDSSF Architecture

```
for(int i=0; i<gshCount;i++)
{
    jobExecution = new JobExecution(searchString,
        supplierDocumentParser.getSupplierSubDocumentString(),
        gshDocumentParser.getGsh() );
    jobExecution.setThreadChecker(threadChecker);
    Thread thread = new Thread(threadGroup, jobExecution);
    thread.start();
} // end of for loop.
```

Figure 7.11 A snapshot of code from *executeJob* method of the *MasterGridImpl* class, which shows how the method allocates database search jobs by creating *jobExecution* threads in a *for* loop. A *jobExecution* object is initialised with three arguments: *searchString* i.e. *searchCriteria*; a subset of SD systems to search provided by *SupplierDocumentParser* class; and GSHs of DSS nodes available in the Grid environment. The GSHs are provided by the *GshDocumentParser* class. This *jobExecution* object is then run as a Java program execution thread. The *for* loop shown in the figure creates four such threads, one for each DSS node in the Grid environment and these four threads run in parallel. See the UML class diagram of MGS in Figure 6.11 for further details.

b) Aggregate product data retrieved by DSS nodes by searching SDs

The MGS was also tested for its ability to aggregate product data retrieved by DSS nodes into a single XML document and send the result to the PSCD front-end application. The functionality of product data aggregation in MGS is provided by the *JobAggregation*, *CallDataAggregate*, *DataAggregate* classes, which help to accumulate product data (retrieved from several SD systems) in one central place in the form an XML document. In MGS, product data collected by individual *jobExecution* threads, is aggregated using *CallDataAggregate*, *DataAggregate* classes. Once all *jobExecution* threads have finished execution, the *executeJob* method uses the functionality of the *JobAggregation* class to send aggregated product data to the PSCD web application to be displayed to the user. As part of this testing, the product data aggregation classes of MGS collected electric bed product data returned by DSS nodes and aggregated them into a single XML document. Figure 7.12 shows a snapshot of an XML document providing information on product data retrieved from SD systems. In the testing, the product data corresponding to *Electric Bed* version 1.0 product class was retrieved from several product supplier databases. The entire XML document showing product data retrieved from all SD systems, and data corresponding to different product specifications is not shown in the figure for brevity. Figure 6.11 shows the class diagram of the MGS system component, and Section 6.3.4.1 identifies in greater detail

7. System Testing, Verification and Validation of the MDSSF Architecture

the other support classes, which provide different functionalities. These were also tested.

```
<NewDataSet>
  <Supplier IDSupplier="3"
    SupplierWsURL="http://131.251.42.40/SupplierApp/ProductService.asmx">
    <Product>
      <IDProdInternal>1143</IDProdInternal>
      <IDProdClass>10325</IDProdClass>
      <IDProdClassVer>1.0000</IDProdClassVer>
      <ProdName>Ampio</ProdName>
      <ProdDesc>General purpose electric bed model</ProdDesc>
    </Product>
  </Supplier>
  <Supplier IDSupplier="4"
    SupplierWsURL="http://131.251.42.33/SupplierApp/ProductService.asmx">
    <Product>
      <IDProdInternal>1144</IDProdInternal>
      <IDProdClass>10325</IDProdClass>
      <IDProdClassVer>1.0000</IDProdClassVer>
      <ProdName>Plano</ProdName>
      <ProdDesc>General purpose electric bed model, acute care</ProdDesc>
    </Product>
  </Supplier>
</NewDataSet>
```

Figure 7.12 A snapshot of product data returned from SD systems.

7.2.3.2 Testing of DSS

The DSS component of MDSS was deployed in four machines in a Grid environment (see Figure 7.2). Each DSS component was hosted in a single machine (node). Section 6.3.4.2 describes the architecture of DSS in greater detail and its UML class diagram is presented in Figure 6.12. The testing of DSS took place by simultaneously invoking four DSS nodes from the MGS (see Figures 7.10 and 7.11). An important class of the DSS system is *DatabaseSearchImpl*, which is exposed as a Grid service and performs the task of searching supplier databases. In the testing process, the MGS invoked this class via its GSH and provided it with a subset of total supplier databases to search and the search criteria. The DSS node then invoked each of the supplier databases by sending a SOAP [Gra02] message containing search criteria to the product supplier database Web Service front-end to retrieve product data. At the product supplier end, the search criteria is evaluated by the Web Service front-end, which interacts with the backend SD system to retrieve relevant product data based on the search criteria (see Figure 7.12). This product data is then sent to the calling DSS node.

7. System Testing, Verification and Validation of the MDSSF Architecture

7.2.3.3 Testing the MDSS as a Single System

The MDSS component of the MDSSF architecture was tested for its data search capability in the Grid environment. During the testing of the Grid enabled distributed database search, a typical search used 7 machines. Of these, the SD system was installed in 3 machines (see Figure 7.2). The MDSS system components, such as MGS was installed in a single machine and DSSs were deployed in 4 machines. The test involved submitting a database search criteria, to available DSS Grid service nodes using MGS. The MGS split the database search operations into roughly equal portions and allocated these to individual DSS nodes (see Figure 7.1). These DSS nodes executed database search operations in parallel to retrieve product data from several SD systems, and presented the retrieved information in an integrated form by aggregating the retrieved data.

An important element of testing the MDSS was to test its scalability. This involved testing the ability of the Grid nodes of the MDSS to search a large number of SD systems in response to a user's query. However, due to limitation of resources, the SD systems were deployed in three machines only (see Figure 7.2). This meant that the MDSS could only invoke these three SD systems to perform a distributed database search in response to a single query. These three SD systems were clearly not enough to test the scalability of MDSS. Therefore, a testing scenario was simulated, whereby these three SD systems were invoked by MDSS several times as part of a single query, under the assumption that these several SD systems are installed in separate machines. This was achieved by creating a *supplierString* XML document which contained several duplicate entries for an SD system and sending this document to MDSS as one of the input parameters. Figure 7.13 shows an example XML document where a supplier with *IDSupplier* value 2 is entered twice in the document. With this document, the MDSS invokes this supplier's Web Service twice and retrieves two sets of the same product data from its SD system. Therefore, when a supplier is mentioned several times in a *supplierString* XML document, the MDSS invokes that supplier that many times as part of the distributed database search. Thus by using this technique, it was possible to test the scalability of the MDSS by giving it a large number of SD systems to search (i.e. three SD systems but mentioned several times in the *supplierString* XML document).

7. System Testing, Verification and Validation of the MDSSF Architecture

```
<?xml version="1.0"?>
<SupplierDataSet>
  <Supplier>
    <IDSupplier>2</IDSupplier>
    <SupplierWS>http://131.251.42.33/SupplierApp/ProductService.asmx</SupplierWS>
    <DataSetName>SupplierDataSet</DataSetName>
  </Supplier>
  <Supplier>
    <IDSupplier>2</IDSupplier>
    <SupplierWS>http://131.251.42.33/SupplierApp/ProductService.asmx</SupplierWS>
    <DataSetName>SupplierDataSet</DataSetName>
  </Supplier>
  <Supplier>
    <IDSupplier>3</IDSupplier>
    <SupplierWS>http://131.251.42.40/SupplierApp/ProductService.asmx</SupplierWS>
    <DataSetName>SupplierDataSet</DataSetName>
  </Supplier>
</SupplierDataSet>
```

Fig 7.13 A snapshot of *supplierString* XML document identifying the product supplier databases (SD systems) to search. In this document a product supplier with ID 2 is mentioned twice. This technique enables searching an SD system by MDSS more than once depending upon the number of times it is entered in the XML document.

Several tests were conducted by providing MDSS with varying numbers of SD systems ranging from 100 to 400 by using the above technique. When the MDSS was given a larger set of SD systems to search, it provided an acceptable response time. In a typical scenario, the MDSS has been tested to retrieve and aggregate product data from up to 400 SD systems in about 35 seconds.³ Table 7.2 shows the approximate time in seconds, which the MDSS system took to search varying numbers of SD systems. In one such testing, the MDSS invoked 400 SD systems (i.e. three SD systems invoked 400 times) by dividing database search jobs among the DSS Grid service nodes. The DSS Grid service nodes collaborated at system level to retrieve and aggregate product data from the SD systems. Several of these tests were conducted. The output of this test is product data which is retrieved from 400 SD systems and aggregated into one XML document. Figure 7.12 shows a snapshot of this XML document. The *Supplier* element in the XML

³ The aim of this testing is to demonstrate the scalability of the MDSS system when retrieving data from a large number of supplier databases in a Grid environment. The Grid support is provided in MDSSF to achieve system scalability (see Section 6.3). The figures presented in Table 7.2 are for the sole purpose of demonstrating the ability of MDSS system to search large number of SD systems and aggregate product data in a reasonable time. In this regard it can be mentioned that the Grid computing (via the Grid middleware and machines on which it is installed) is used in the present research with the aim to achieve system scalability and coordinated resource sharing, although Grid computing also provides various other features [Fos01] [Fos02] [Don06] [Fos99]. It is not considered from the perspective of application of Grid computing in other areas or areas such as High Performance Computing [Hua06] [Dar07] [Don06] which does not lie in the domain of present research.

7. System Testing, Verification and Validation of the MDSSF Architecture

document of Figure 7.12 differentiates the product data returned by each SD system. For example, the XML document shows product data returned by product suppliers with ID 3 and 4. The MDSS is also tested for its ability to perform several product database search jobs at a time. This demonstrates the ability of MDSS to search a large number of SD systems and provide an integrated view of product data retrieved from these SD systems in a reasonable time. The DSS Grid service nodes of MDSS are deployed in 4 machines only. Therefore, there is a limit to the maximum number of database search jobs it can handle at a time due to resource limitations (such as CPU and memory). The improvements which are needed to MDSS architecture to further enhance system scalability are discussed in Sections 7.4.3.6 and 7.4.3.7 as part of MDSSF evaluation.

Number of SD Systems to Search	Approximate Time (in seconds)
100	17
200	23
300	28
400	35

Table 7.2 The number of SD systems searched by MDSS and approximate time in seconds.

The above discussion describes the testing of MDSS for its ability to search a large number SD systems via a technique, where a single SD system is mentioned several times in the *supplierString* XML document (see Figure 7.13). The author believes that from a feasibility point of view, this is a worthwhile approach which tests the functionality of the prototype system in a reasonable timescale and with the use of finite resources. Due to limitation of resources, the SD systems were deployed in 3 machines only. Assuming that a large number of machines were actually made available to the project for testing purpose, it would still have been a complex and time consuming task to set-up SD systems in all these machines. The main aim of this testing is to demonstrate the ability of MDSS to handle and process a large amount of product and supplier data in a Grid environment. For this purpose, the presence of several of SD

7. System Testing, Verification and Validation of the MDSSF Architecture

systems is not needed. This is because *DatabaseSearchImpl* class⁴ of the DSS system (and the other classes of MDSS) performs a series of steps when it invokes an SD system to retrieve product data. It performs these same steps every time and for every SD system it invokes. Therefore, from a technical point of view and for testing purpose, it is irrelevant whether the same or a different SD system is invoked every time product data is requested. Since the MDSS performs same steps for every SD system mentioned in the *supplierString* XML document, a large number of SD systems can be specified in the XML document. This can also contain duplicate entries of an SD system. These large number of SD systems then help to test the scalability of MDSS because it invokes each SD system mentioned in the XML document to request product data. An SD system is invoked that many number of times as it is mentioned in the XML document. For example if an SD system is mentioned 100 times in a XML document it gets invoked 100 times by the MDSS. The drawback of this approach is that it retrieves duplicate data due to duplicate entries of an SD system in *supplierString* XML document. As a result the same set of data is returned by an SD system every time it is invoked for a given search. This is acceptable in a testing environment because the aim here is to test the ability of MDSS to retrieve and process large amount of product data. This approach has successfully tested the scalability of MDSS despite limited resources. Thus, by providing the same supplier information in the *supplierString* XML document more than once, a large number of SD systems were invoked in the Grid environment without needing a large number of SD systems to be installed in separate machines.

7.2.4 Test Objective 4

Test the ability of the MDSSF architecture to provide up-to-date product information to consortia members such as contractors.

The MDSSF architecture provides local autonomy to product suppliers and it retrieves product data from SD systems through the MDSS system. This allows suppliers to manage their own product information without external influence. This means that the information available through the MDSS is likely to be accurate and up-to-date. This is due to the product data being decentralised and autonomously managed by the product

⁴ The series of steps refer to the functionality provided by the *DatabaseSearchImpl* class and the other classes of the MDSS system for retrieving product data from SD systems (see Section 7.2.3.3). See Appendix 5 for system code of MDSS.

7. System Testing, Verification and Validation of the MDSSF Architecture

suppliers themselves who are responsible for managing their own product data and it is in their interest to keep it up-to-date. It is anticipated that product information provided by product suppliers has the potential to influence procurement planning and procurement decision making by contractors if it is readily available. Thus, due to market forces and the need to compete with other product suppliers, who may also be participating in MDSSF and supplying similar or identical products, the product supplier will ensure accurate and up-to-date descriptions of their products in their SD.

Product information can be updated in an SD by performing database update operations through SQL queries. The MDSSF architecture does not provide mechanisms to cache product data. When product data is requested by a user, a new distributed database search takes place. Therefore changes to product data made by product supplier in their SD can be reflected immediately in the PSCD application, when the next search takes place. This ensures that up-to-date product data is retrieved every time. Several tests were conducted to test this feature of MDSSF by updating product description in an SD and then performing a distributed database search.

Another important aspect of ensuring that up-to-date information is available, is to provide notification to users when product information cannot be retrieved from an SD. This can happen when an SD system is unavailable or is inaccessible due to network problems or when the Web Service URL is malformed. During such events the *DatabaseSearchImpl* class of the DSS system generates an error message as an XML element and adds it to the returning XML document (see Figure 7.14). This error message identifies the SD system from which product data could not be retrieved.

7. System Testing, Verification and Validation of the MDSSF Architecture

```
catch(Exception e)
{
    /* This exception is thrown when the stub is unable to invoke the Supplier
    * Web Service. When the stub is unable to retried data from Supplie Web
    * Service URL then error string is encoded into XML element and added to
    * the returned XML document by DataAggregate class. This XML element
    * identifies the supplier Web Service from which product data could not be
    * retrieved. */

    String errStr = "\n<Error>\n\t<ErrorString>\n\t\tCould not retrieve data from URL " + url.toString() +
        "\n Either URL is malformed or connection to the web service is \n refused (a web service " +
        "may not be available) or access to the database server is denied or is unavailable.\n\t " +
        "</ErrorString>\n\t</Error>\n";
    supplierDetails[2] = "Error";
    dataAggregate.addNodes(errStr, supplierDetails);
} //end of catch.
```

Figure 7.14 An exception generated by *DatabaseSearchImpl* class of DSS system when it fails to retrieve product data from an SD system.

7.2.5 Section Summary

Section 7.2 described testing of the MDSS system components as part of a testing strategy which determined if it met the procurement challenges (see Section 1.2) and addressed the limitations of the original PSCD application (see Section 1.3). In addition to the tests described in this section, several other tests were also conducted to test the features of MDSSF components not described in this section. These features include testing the MDSSF databases for transaction management, error notification, generation of IDs for entities such as product classes and its specifications and product class category management (see Sections 6.2.1.6 and 6.2.1.7). The three databases of MDSSF provide a total of 122 stored procedures – all of these were tested as part of their development. The Java classes of the MDSS system were tested individually and as part of a single system as they were developed. These classes provide other support functionality, such as parsing of supplier and product data, data aggregation and transformation, and invoking Web Services (see Section 6.3.4.1 and 6.3.4.2). Tests were carried out to test these support features of the MDSSF system components. Although the successful testing of the components of the MDSSF architecture, as part of the PSCD application demonstrates the viability of the architecture, there is a possibility of making further improvements to the system to enhance system scalability. This is discussed in Section 7.4 as part of the evaluation of the MDSSF architectural components.

7.3 MDSSF Demonstration

The MDSSF architecture was developed to Grid-enable the PSCD application. Several demonstrations of the application were given to different audiences during the COVITE project. The functionalities of the application (and with it the capabilities of MDSSF components) were demonstrated to the industrial collaborators of the COVITE project APSL on several occasions. The application was demonstrated to members of staff at the School of Computer Science, Cardiff University in a departmental seminar. A demonstration of the application was also given at the UK e-Science All Hands Conference in Nottingham in September 2004⁵. The demonstration showed the creation of a product class and its subscription by product suppliers. As part of the demonstration the product search took place using the MDSS deployed in the Grid environment. The Grid enabled MDSS System queried a dynamic selection of relevant supplier databases (SDs) to extract, in real time, information about the products which a Virtual Organisation (VO) or a contractor wished to acquire. The product search criteria submitted by a Contractor/VO retrieved - in real time - the available SDs meeting the search criteria. The SDs meeting the search criteria were then invoked via the XML based web service using MDSS. Searching a large number of SDs took place using DSS nodes in a Grid environment, where the nodes worked collaboratively and invoked SDs to retrieve product information in the form of XML documents. The Components of MDSSF were deployed in the local computing environment and these were invoked by the PSCD front-end application remotely from the conference location as part of the demonstration.

The PSCD application based upon the MDSSF architecture also received potential interest from external organisations to identify how it could be used to aid procurement activities. This interest was shown by the e-Procurement department of the Welsh Assembly Government in November 2005 and Welsh Health Estates in March 2006, which is also a branch of the Welsh Assembly Government. The e-Procurement department were interested in the potential use of PSCD to help local authorities throughout Wales with purchasing. The Welsh Health Estates were interested from the perspective of procurement and management of health facilities in Wales. It is likely

⁵ The PSCD application was demonstrated in the 3rd UK e-Science All Hands Conference by another member of the COVITE team. The other functional areas of the PSCD application (see Section 2.2) such as security, PSCD front-end application and user management were also demonstrated.

7. System Testing, Verification and Validation of the MDSSF Architecture

that these opportunities can be pursued with further research in this area and participation of the industrial partners who were interested in the system.

The MDSSF system architecture and the prototype system is the outcome of the research which was undertaken as part of the COVITE [Mil02] project. The project was funded for two years by the Department of Trade and Industry (DTI), it started on 01 October 2002 and finished on 30 September 2004. Due to limited external funding, the project (and author's research) could not move forward from the current prototype stage to the next stage. However, there is a possibility of taking this research forward if public funding becomes available.

7.4 Verification and Validation of the MDSSF Architecture

This section verifies and validates the MDSSSF information sharing architecture in terms of the research objectives achieved.

7.4.1 Benefits to the Construction Industry

The first objective (see Section 1.4.1) of this research was to identify how the new information sharing architecture will benefit construction industry actors with the aim of presenting a solution to the procurement challenges identified in Section 1.2. The requirements were identified by the industrial collaborator APSL, and the project supervisor Professor John Miles. These requirements were to address the procurement challenges and limitations of the PSCD application (see Sections 1.2 and 1.3). This research successfully met these requirements and proposed a solution to procurement challenges via a new information sharing architecture. In view of these requirements, the existing role of information, technology and practices that drive construction procurement were examined to identify how a new information sharing architecture will benefit construction industry actors in their desire to make better procurement decisions. Chapter 3 provides a detailed view on different ways in which the MDSSF information sharing architecture can benefit construction industry actors.

As part of the first objective of this research, the role of information in the construction industry was examined (see Section 3.4). It was necessary to understand this role because the MDSSF provides a new method for information management and sharing.

7. System Testing, Verification and Validation of the MDSSF Architecture

The MDSSF, via its software components also provided a technology based solution to procurement challenges. The volume of information generated by construction projects is so large, that using IT to handle it is highly beneficial. The role of IT to enhance and improve construction industry processes is well recognised (see Section 3.2.3). In section 3.5, this role of technology in construction industry processes is presented in greater detail. The MDSSF through its software components presented a technology-based solution and used similar kinds of technological options to those used by other procurement systems. Section 3.7 identifies in greater detail how the MDSSF, through its software components, can benefit existing procurement systems or approaches by providing product and supplier information.

Procurement is a very wide subject area. In order to carry out procurement related activities effectively, organisations also consider many other factors having wider influences, scope and implications on themselves and on the construction industry as a whole. Organisations spend considerable resources and time, and form different types of collaborations and partnerships, not only to ensure timely delivery of project outputs, but also to make further improvements, for example by introducing innovative techniques, by making changes based on lessons learned from past projects, by seeking greater specialisation to reduce costs and outsourcing non-core functions. This research also identified different ways in which the MDSSF can benefit practitioners in the construction supply chain for procurement in different organisational scenarios. Section 3.6 identifies 14 different factors and/or scenarios which influence procurement and how the MDSSF can be of benefit in those scenarios by providing integrated access to product data from several autonomous product suppliers. The different factors and/or scenarios underpin the need for such architectures and the potential benefits it can provide. The applicability and use of MDSSF in different procurement systems; its role in information sharing and providing a technology based solution; and the different ways in which it can benefit practitioners in the construction industry, which are identified in Chapter 3, confirms that it can provide benefits to industry actors in several different ways. This also confirms the achievement of the first research objective.

7. System Testing, Verification and Validation of the MDSSF Architecture

7.4.2 MDSSF Information Sharing Architecture

The second objective (see Section 1.4.2) of developing an information sharing architecture which meets the procurement challenges of Section 1.2 was realised through the MDSSF architecture, which is the main contribution of this research. In Section 7.2 we discussed the functionality of the different system components of MDSSF as part of the system testing. The functionality of different system components gives certain distinctive features to the architecture, which are part of its novelty and provide a new federated information sharing model enabling a novel type of collaboration between construction industry actors. This section describes the novel features of the MDSSF architecture.

7.4.2.1 Grid Support

MDSSF is designed to provide Grid technology support for retrieving product information and sharing it with contractors in real time. The MDSS system within MDSSF provides the functionality to distribute a product search across a number of machines in a Grid network, where the machines collaborate to execute database access operations. It also provides the functionality to aggregate the data retrieved from a large number of database systems. The architecture is unique in that it provides Grid technology support in a federated database environment. The architecture of MDSSF is also unique from another perspective. It does not Grid enable supplier databases, but Grid enables database access operations only. The need to Grid-enable database access operations only was identified, as there are a large number of supplier databases which need to be accessed and these databases are not normally available in the Grid environment. These databases are managed and controlled by independent and autonomous product suppliers. The MDSSF architecture is designed under the assumption that the autonomy of the SD systems must be maintained, as they are owned by organisations that do not provide database operations such as data definition, data insertion and data update to external users but will support access to the databases by external users to retrieve data. The distinction between this approach and some of the other Grid related database projects is identified in Section 4.6.

7. System Testing, Verification and Validation of the MDSSF Architecture

7.4.2.2 Federated Architecture Supporting Competing Product Suppliers

MDSSF provides an integrated way to access product information from a large number of product suppliers using a single system. It searches for the same or similar products, but supplied by different suppliers and provides an integrated view of this information to contractors. Hence, when the system brings together product information from different SD systems, it enables product suppliers to compete with each other in a virtual market place, based on the product information they provide to the federation users (contractors). It also provides contractors with an opportunity to judge competing product information against their own project requirements.

7.4.2.3 MDSSF's Cooperation Model

MDSSF can be described as a federated information sharing system, as it federates SD systems via the Grid-enabled MDSS and the CDM (Common Data Model), based on the product class concept. It is recognised in the literature that *cooperation among independent systems* is an important characteristic of federated databases (FDBS) for sharing data [She90]. MDSSF adopts a different model of cooperation from traditional FDBS architectures, which enables data sharing between the component DBSs participating in the federation. The MDSSF model of cooperation does not allow sharing of data between component DBSs supplying the data (i.e. between the SD systems of the product suppliers). In the MDSSF architecture product suppliers share their data with the contractors only. This is because product suppliers are business organisations, who do not wish to disclose their product related sensitive data to their competitors, who are participating in the same federation. They only cooperate with the centralised MDSS so that appropriate information about their products is retrieved and sent to the contractors in the standard data structure of the VDD.

7.4.2.4 Schema Integration

The MDSSF information sharing model is different from other federated database or mediator based architectures with respect to schema integration. In MDSSF, unlike other information sharing systems (see Section 4.4) the sharing of data takes place between contractors and the supplier databases without the need to create federated, integrated or external schemas. This schema integration is not required in the MDSSF approach, because of the homogeneous nature of the product data which is exchanged in

7. System Testing, Verification and Validation of the MDSSF Architecture

MDSSF. The adoption of standard data exchange methods, based on the product class concept, provided the opportunity to design a novel database federation model having a Grid enabled distributed database search mechanism at its core, in place of mechanisms for resolving schema conflicts. Schema integration is further not required in the MDSSF approach, because data is not integrated in its environment. Product information retrieved from several SD systems is only aggregated by MDSS, so that it can be presented to the user in an integrated and consistent way through the PSCD front-end web application. The retrieved product information is aggregated into a list but there is no integration of this information.

7.4.2.5 Subscription-based Approach

The MDSSF is also a novel information sharing architecture from the perspective of its data model and data subscription. The architecture (via its architectural components) allows subscription to the data model of the federation and product classes by the product suppliers (see Section 6.2.2.1). This subscription based approach provides product suppliers with readily available mechanisms to manage their product data in a structured way by using the standard specifications of product classes to describe product features and in this manner tackle the issue of heterogeneity. The approach is beneficial to the suppliers and particularly SMEs (small to medium enterprises), who may not use database systems for managing product data and often provide data in the form of PDF files or use other unstructured mechanisms which are difficult to manage and search. Thus the subscription based approach can provide SMEs with a database solution to manage their product data and also make their product information available to potential buyers and contractors via the MDSS.

7.4.2.6 A Standard Method of Information Storage and Exchange

The second objective of this research also identified the need to develop a standard method for information storage and exchange to address heterogeneity and allow suppliers to manage product information using standard means (see Section 1.4.2). The rationale behind this approach is to promote standardisation from the beginning at the data exchange level, so that the inconsistencies, which heterogeneity brings, can be avoided in the first place to enable collaboration. Standardisation plays a key role in the MDSSF architecture, to support data sharing in a standard and integrated way. This

7. System Testing, Verification and Validation of the MDSSF Architecture

objective has been achieved by using the concept of product classes in MDSSF. An important advantage of product classes is that they provide product suppliers with a readily available means to describe their product features in a standard and structured way. It also supports product data evolution. It gives standard product definitions which can be used by product suppliers to provide descriptions of products, they can supply to consortia members or contractors. When all product suppliers use standard product classes to describe the same or similar products, they can be easily compared by a contractor. The achievement of this research objective both at the conceptual and implementation level, addresses the third procurement challenge of this research which was to tackle the issue of heterogeneity by developing a standard means of product data management and evolution (see Section 1.2).

7.4.3 The Software Components of the MDSSF

The aim of the third objective (see Section 1.4.3) was to empirically show the viability of the new distributed information sharing architecture by designing and implementing its various software components, to address the procurement challenges at the data management and sharing level. This has been achieved by designing and implementing its various software components. Chapter 6 describes the architecture of the MDSSF. The data definition and management software components of MDSSF such as the PCD, SPCD and SD systems provide mechanisms to create product class definitions, product descriptions and allow their management. The functionality of data search in MDSSF is provided by its MDSS component.

The MDSS is designed to operate in a distributed environment comprising of several machines and uses Grid technology to perform a distributed database search of a large number of supplier databases to retrieve required product information (see Section 6.3). It provides the benefit of integrated access to the large amount of product data available from several SD systems, through the use of a single system. This addresses the first procurement challenge of being able to access product information available from several suppliers in an integrated manner. The Grid support provides the necessary scalability and computational capability that is required to access such information. The MDSS also provides another important benefit as it enables the building of a large and comprehensive product data repository (the VDD).As identified in Section 7.2.4 there is

7. System Testing, Verification and Validation of the MDSSF Architecture

a high degree of probability that the information in the VDD is accurate and up-to-date. This is due to the product data being decentralised and autonomously managed by several product suppliers, who are responsible for managing their own product data.

At the data management and exchange level, the software components of the MDSSF provide all the necessary features to enable product data management and sharing to meet the procurement challenges. The development of this prototype system to meet these challenges confirms the achievement of the third research objective.

7.4.3.1 Software Components at Supplier Side

The supplier side product data management infrastructure allows product suppliers to retain full control of their product data. However this flexibility also brings in a degree of complexity because setting up and managing the product data management infrastructure is a non-trivial task. In Section 7.2, as part of system testing, we have seen the complexity associated with managing product data through product classes. The infrastructure requires the expertise of a database specialist to ensure that databases are up and running all the time and efficiently serving the contractor's request via the Grid enabled-MDSS. If an SD is down or is unavailable then the MDSS will not be able to retrieve product information from it. A database specialist may also be required to perform other database related operations such as downloading product classes and describing product data using product class constructs. Thus, managing product data at the supplier side and ensuring that it is always available to serve a contractor's request can have implications on the cost and other resources of product suppliers. An alternate solution is to outsource the task of product data management to a third party, such as a product data management service provider. This approach can provide two potential benefits:

- A service provider can provide a low cost solution to managing all the product data of all product suppliers participating in MDSSF and making it available to contractors on behalf of product suppliers. Using this approach, product suppliers will not have to set-up and manage the complex infrastructure on their side. Product suppliers can be provided with a simple browser based access to subscribe to product classes and create product descriptions remotely.

7. System Testing, Verification and Validation of the MDSSF Architecture

- This approach is also efficient from the Grid-enabled database search perspective. In the present approach the Grid-enabled MDSS has to invoke several supplier web service interfaces to retrieve product data from several SD systems. If all the product supplier data is available from a central location (where the service provider manages the product data of several suppliers) then the search time can be reduced and searches made more efficient, as MDSS will not have to invoke several supplier databases. This task can then be performed by requesting all relevant data from the service provider using a single operation for these SDs.

We believe that this alternate solution will not affect the autonomy of product suppliers, as they will still be able to retain full control of their product by administering their data remotely. Contractual agreements between the service provider such as APSL and product suppliers, and security measures can be set up, which can restrict product data sharing between product suppliers and its viewing by an unauthorised party. This solution will also not affect the overall MDSSF architecture and the architecture will still be able to retain its distinctive features as described in Section 7.4.2.

7.4.3.2 MDSSF Supports a Single RDBMS and Schema

In the prototype system the data management infrastructure at the supplier side is rigid from the perspective of supporting a single RDBMS only. All three databases in MDSSF are implemented using SQL Server 2000 [Vie00] RDBMS. It does not offer a choice to product suppliers to use the RDBMS of their choice or to provide supplier side databases which conform to an already existing data management platform. Thus, currently MDSSF imposes the rigidity of having to use the single RDBMS provided to product suppliers, if they wish to participate. Supporting different RDBMSs will require implementation of supplier side databases using different RDBMS based on the PCD schema. Additionally, the prototype only supports the single database schema of the PCD system. The database schemas of databases at the supplier side (i.e. SPCD and SD) are also designed and implemented to correspond to the PCD database schema. This is a design decision for the prototype, where adopting a single schema is necessary to create standard mechanisms of product data definition, management and sharing in a reasonable timescale. However supporting a single product data management approach

7. System Testing, Verification and Validation of the MDSSF Architecture

via a single schema also restricts the interoperability of the system with other prominent product data management systems or modelling standards, languages, information reference models and data exchange formats. Section 5.5 presents some of the existing information modelling/management systems/standards in the AEC industry. Although comparing the MDSSF product data management technique with other product data management infrastructures is not the main focus of this research, significant research and effort will be required in the future in this direction, to make the MDSSF compliant with other product data management infrastructures. This will provide a number of benefits. Among them, an important benefit is that the MDSSF model of information sharing can also be used to support the other product management schemas or vice versa. Efforts in this direction are also required to support the migration of product data from other industry recognised product schemas to the PCD schema to allow product suppliers with existing databases to evaluate the MDSSF way of managing and sharing data before fully adopting it. We believe that providing such facilities and features will encourage wider participation in the product supplier community.

7.4.3.3 Linking of Databases at the Supplier Side

The product data created in a SD refers to its corresponding product class in the SPCD system (see Figure 6.3). Presently this reference to a product class is limited to identifying product class data using manual means (i.e. performing database select operations in SPCD) and then using the product class specification information to perform database insert operations in SD via the database stored procedures. This ensures that correct specifications are assigned to products in SD. However the current manual process is also prone to errors because SPCD and SD are not inter-linked. Linking these systems together via a software component is necessary to ensure that product data in a SD is consistent with its corresponding product class in SPCD. This will ensure a supplier does not inadvertently insert wrong data or assign a specification to a product where such a specification does not exist in the corresponding product class.

7.4.3.4 Consistency Checks and Data Constraints

The PCD system in its present form does not perform consistency checks to ensure specifications are assigned to product classes correctly. For example a sub-product class

7. System Testing, Verification and Validation of the MDSSF Architecture

such as *Panel* cannot be assigned to a product class such as *Chair*, as it will create an illogical description in the product class. In addition to consistency checks, there is also a need to apply constraints to product classes and specifications. It is required to aid product suppliers to provide correct descriptions of their products and ensure that a product description is valid and consistent. For example when providing a value for a specification such as *Height* assigned to a product class such as *Chair*, a constraint can be introduced to ensure that the value lies within a given range, which is typical of the product feature. The application of constraints is also required to allow product suppliers to describe compatibility criteria. For example, there is a possibility that certain types of door handles are compatible with only certain types of doors and therefore specifications cannot be assigned unless the real world products they represent are compatible. A possible solution to this problem is building a knowledge repository which guides its users to ensure correct usage of product classes. The knowledge repository can provide information on different types of specifications, and how and when these can be used and the constraints associated with them.

7.4.3.5 Need for the Support of Additional Database Operations

The three databases developed allow creation of product classes, support product class subscription and description of the actual product data. These database operations in their present state are limited to providing database insert operations only to enable creation of product descriptions. There is also a need for other database operations such as data update and data delete. It is anticipated that a particular product class version will be reviewed several times by specification designers and industry experts during its design process. In this process specifications may be assigned, unassigned or reassigned to meet the requirements of the actual product before the version is finalised and released for subscription by product suppliers. Database update and delete operations will be needed in such scenarios to allow specification designers to test various options and incorporate the feedback or opinion of industry experts. In SD, these database operations are required to let product suppliers manage their product data effectively by changing product data values or deleting product data (for old, obsolete or discontinued products).

7. System Testing, Verification and Validation of the MDSSF Architecture

7.4.3.6 The Grid-based Search

MDSS provides an integrated view of the information available from several suppliers by aggregating this information. It is a search mechanism, but it also enables creation of a VDD to provide access to product information in real time. MDSS provides a scalable solution when accessing a large number of SD systems, as it distributes the search across a number of machines available in a Grid cluster. However further improvements to its architecture are necessary to provide enhanced support when performing a distributed database search. This is explained below.

When performing a distributed database search, a system's physical metrics, such as CPU speed, memory and storage available and network bandwidth also have to be taken into account in order to maintain an acceptable level of performance. A database search job submitted to a DSS instance can be computationally intensive, as it may require searching several SD systems and aggregating the resulting data. Therefore several such computationally intensive jobs cannot be submitted to a given machine (running the DSS service) at the same time without first analysing if the machine would be able to allocate the required resources to the DSS instance. If several jobs are submitted, then there is a possibility that the system may become slow, fail to respond and eventually crash because of system overload. Presently MDSS does not provide any mechanism to identify whether a given DSS service node will be able to allocate the required resources (such as CPU and memory) to execute the job submitted. The jobs are presently submitted by MGS (Master Grid Services) to available DSS nodes without giving due consideration to the current state of a system's physical metrics. Thus further improvements are required to the MDSS system to address this challenge. One possible solution to address this issue, is to use cluster monitoring toolkits such as Ganglia [Gan07] to provide information on available nodes, which can potentially be used to perform the search. The Ganglia toolkit provides information on various metrics such as CPU speed, available memory, storage available and network I/O. This information can be used to identify nodes in the cluster which can handle a given job. Additionally MDSS does not provide fault tolerance and system recovery facilities in the event of unexpected hardware or software failure. Addressing these system level issues is also vital, in order to further enhance system scalability, reliability and performance. For similar reasons further improvements to the architecture are also required in terms of

7. System Testing, Verification and Validation of the MDSSF Architecture

increasing the size of the Grid cluster by adding more nodes and increasing the number of MGS service instances, so that MDSS can handle a large number of user requests at a time.

7.4.3.7 The Grid Middleware

MDSS is designed using the GT3 core component of the Globus Toolkit [Fos98] version 3.0.2, available from Globus [Glo09]. The COVITE project was undertaken in parallel with a development period of Grid technology and Grid middleware. During the development period of MDSS, version 3.0.2 was the latest Grid middleware toolkit available from Globus. MDSS was therefore developed with the latest toolkit version available at the time of its development. However, with the continual evolution and improvements in Grid technology, the corresponding Grid middleware has also evolved, and presently version 3.0.2 of the Globus Toolkit is no longer supported by Globus. Version 3.0.2 is based on OGSII (Open Grid Services Infrastructure) [Tue03] specifications. These specifications have been refactored to produce new specifications called the WSRF (Web Services Resource Framework) v1.2 specifications – currently an OASIS standard [Oas09]. The new specifications, in addition to providing other features, also take advantage of recent developments in Web Services architectures [Cza04]. The present stable version of the toolkit is thus a WSRF based Globus Toolkit which is version 4.2.1. The implication of these changes in Grid middleware to MDSS, is that MDSS will have to be reengineered to take advantage of the new middleware. This may involve redesign and implementation of the DSS component of MDSS. Since version 3.0.2 is no longer supported, further development to MDSS (in terms of enhancing features and functionality and addressing present issues) cannot be made unless underlying Grid middleware is changed. Significant effort may be required to achieve this as part of future work, however these efforts would be worthwhile as it would allow the utilisation of new features and functionalities of the new toolkit by MDSS which could be used to enhance its performance and functionality.

7.4.4 System Testing and Evaluation of the MDSSF

The fourth and final research objective of this research had been testing and evaluation of the new MDSSF information sharing architecture in terms of the research hypothesis and research objectives achieved (see Section 1.4.4). This chapter presented testing and

7. System Testing, Verification and Validation of the MDSSF Architecture

evaluation of the MDSSF with the aim to confirm the achievement of all the research objectives which were outlined to address the procurement challenges of this research. The testing of all software components of MDSSF and evaluation of MDSSF from several different perspectives in this chapter confirms the achievement of the fourth research objective.

7.5 Chapter Conclusions

This chapter presented details on system testing of the data definition, data management and data search capabilities of the architectural components of the MDSSF for meeting research objectives. The MDSSF architecture is the main contribution of this research. Like any other information sharing architecture, MDSSF also has its share of limitations which are highlighted as part of the critical evaluation of MDSSF's architectural components and the further improvements needed to address these limitations are identified. Despite these limitations the architecture of MDSSF, through its distinctive features (see Section 7.4.2) provides a novel mechanism of information sharing in the construction industry domain which aids product procurement.

8. Conclusions

8.1 Introduction

This chapter presents a review of the research and highlights the research objectives achieved against the research hypothesis. Section 8.2 identifies research objectives achieved and confirms the research hypothesis. Section 8.3 provides a summary of the contribution of this research to its field. There is a possibility that research methods developed as part of this research or components of MDSSF architecture can be used in other domains to improve or enhance collaborative working in those domains. Section 8.4 briefly identifies some of these domains. The future work in Section 8.5 identifies potential areas for further research and provides a brief description of such a research area, namely addressing data management challenges in a collaborative problem-solving environment.

8.2 Achievement of Research Objectives

This research achieved all its research objectives (see Section 1.4) in that it met the procurement challenges identified in Section 1.2. The procurement process involves obtaining desired products from a wide range of products available from a large number of product suppliers. In large projects a large quantity of various kinds of construction material are required. Chapter 1 identifies the scale of purchases required for building large artefacts such as hospitals and office blocks, and shows that this procurement is a non-trivial exercise. Although web based communication and network technology is beginning to play an increasingly important role in supporting collaboration in AEC projects, collaborative working is still restricted by the current limitations of network and communication technologies and the system architectures which are usually client/server based. In order to address the current limitations and the procurement challenges, the MDSSF information sharing architecture takes advantage of recent advances in distributed computing, particularly Grid computing, and couples it with federated database concepts such as distribution of data and autonomy of databases. The novel MDSSF architecture via its architectural components provides mechanisms for data definition, data management and data search. These mechanisms in turn provide a way to address the procurement challenges identified in Chapter 1, namely to

8. Conclusions

- Provide a single integrated means of accessing product information available from the databases of a large number of product suppliers.
- Provide up-to-date information about products which can be acquired from the external product suppliers, so that this information, such as product specifications, availability, delivery time and cost can be taken into account in procurement planning.
- Develop standard mechanisms for product data management, which address the issue of heterogeneity, which occurs because different suppliers manage and present product information differently.

In order to address the procurement challenges MDSSF created a Virtual Distributed Database (VDD) using Grid enabled distributed database search mechanisms. This benefited the industry actors, such as product suppliers, by allowing them to retain control and autonomy of their product data, while allowing them to share product information with contractors. It also benefited the contractors, as it provided them with an integrated means of accessing the product information available from the databases of a large number of product suppliers. The MDSSF allows product data management by individual suppliers, so that this information can be managed effectively and is provided to contractors via the VDD, which utilised Grid technology to provide scalable support. This addressed the second procurement challenge of providing up-to-date information about products, which can be acquired from product suppliers. In order to address the third procurement challenge, the concept of product classes (described in Chapter 5) is adopted in MDSSF. This provided a mechanism to create standard product definitions for use by product suppliers to describe products in their databases via a subscription based approach. The solution to all three procurement challenges is presented in the form of a novel federated information sharing architecture called MDSSF.

The various software components of the MDSSF information sharing architecture were implemented to test the viability of building such an infrastructure. The software components, which are presently at a research prototype stage, address the procurement

8. Conclusions

challenges at a data management level by providing mechanisms for data definition, data management and data search. Three database systems were developed, namely the PCD system, the SPCD system and the SD system for data definition and data management. These databases via database stored procedures enable the creation of product classes and their specifications by industry knowledgeable specification designers. Five different types of specification have been identified and implemented in the PCD system. These can be assigned to product classes to allow description of different product features in the most appropriate way. Once these product classes are created, the database stored procedures of SPCD provide, via a subscription based approach, a mechanism to download these product classes into SPCD by executing these procedures. This allows product suppliers to subscribe to all relevant product classes, which correspond to products they supply. Once the product classes are subscribed, product data pertaining to actual products can be described in the SD to enable local data management. A particular benefit, which the product data management level components provide, is a readily available tool for product suppliers to describe their product data. As identified in Chapter 2, the UK construction industry is highly fragmented. There are a large number of construction firms, most of which are small to medium scale enterprises. Many such enterprises still do not use IT systems for product data management, because of the complexity and cost, and so make their data available using unstructured file formats such as PDF. It is anticipated that this research could provide a cost effective solution which allows such enterprises to manage product data effectively.

The data search requirements of MDSSF's information sharing architecture were achieved by developing the Grid-based MDSS, which provides a mechanism to search a large number of supplier databases and retrieve product information in response to a contractor's query. Grid support is provided to MDSS in order to achieve system scalability. Providing Grid support to MDSS is important because it retrieves product information from a large number of supplier databases, which enables information sharing between contractors and suppliers in real time. The Grid-based MDSS addresses the computational requirements of the distributed database search. It also provides another important benefit. It enables the building of a large and comprehensive product data repository (i.e. VDD), where there is a high degree of probability that the

8. Conclusions

information available though VDD is accurate and up-to-date. This is due to the product data being decentralised and autonomously managed by the product suppliers, who are responsible for managing their own product data. It is anticipated that the product information provided by product suppliers has the potential to influence procurement planning and procurement decision making by contractors from the early design stages. Thus, due to market forces and in order to compete with other product suppliers, who may also be participating in MDSSF and supplying similar or the same products, the product supplier will provide accurate and up-to-date descriptions of their product data, if they want their products to be selected by the contractors and purchased.

The MDSSF architecture, like any other information sharing architecture also has limitations, which restrict its scope. Chapter 7 verified and validated the MDSSF architecture to identify its limitations and identify potential solutions to address these limitations. The verification and validation process identified the current limitations such as complexity and rigidity of the product data management infrastructure. The areas which require further improvements, in terms of providing additional functionality or to achieve greater efficiency and to reduce the system's complexity were also highlighted in the chapter. The verification and validation is performed for both the data definition and data management components of the MDSSF architecture. The verification and validation of data search components identified the improvements that are required to be made to make a Grid based search more efficient and effective. It identified that the improvements, which need to be made, should also take into consideration performance related factors of the distributed database search. The verification and validation also called for changes to the underlying Grid middleware to take advantage of the new versions of the middleware.

Despite its present limitations, the novel architecture of MDSSF meets the requirements, because these limitations do not affect its ability to meet the challenges. This is because these limitations correspond to architectural components and not to the architecture. The architecture meets the criteria of the hypothesis and addresses the procurement challenges by providing a novel mechanism of information sharing between construction industry actors which aids the procurement processes.

8.3 Research Contributions

8.3.1 Research Publications

The research was undertaken as part of the effort of a collaborative team where different team members looked into different aspects of the procurement challenges identified in the COVITE project proposal [Mil02]. The author's efforts concentrated on addressing procurement challenges identified in section 1.2. The MDSSF information sharing architecture, which is the main contribution of this research, addresses these procurement challenges at the data management level. As a member of the team the author made contributions to various research publications where the author's contributions to publications was the research presented in this thesis. Various aspects of this research are present in [Bur03], [Bur05], [Joi04], [Mil04], [Pah04]. The architecture of the MDSSF is presented in [Pah06b]. Further publications will be made as part of future work.

8.3.2 Novelty of the Proposed Approach

The author reviewed different information sharing models or reference architectures, which covered several different approaches to share information in an integrated way, where such information is available from different autonomous sources. Several schema integration approaches were reviewed to understand their scope and applicability to new information sharing models. Various different information sharing approaches have been suggested in the literature covering different technical approaches such as ontologies [Haj08], [Sar08], knowledge bases [Nak08], [Ahm08], mediators [Qua08], wrappers [Aka08], [Fan08], CDM [Jun08], schema integration [Fen08], [Koz07], [Chi08], FDBS [Yan07], [Bon08], [Dal08], [Ber08] and Grid-based [Kra08], [Lyn09], [Gra08], [Lyn08], [Wan08], [Ahm08a] which are used to address different types of conflicts and enable interoperability of information systems and/or provide information in an integrated way. Although all these approaches provide useful techniques to meet the identified requirements, none of the reviewed information sharing models or reference architectures fully met the requirements of the MDSSF model of information sharing. The MDSSF information sharing architecture, shares certain similarities with other information sharing models/architectures (see Chapter 4). However, clear distinctions also exist, which highlight the novel features of MDSSF. The MDSSF architecture was developed with the aim of enabling sharing of product information

8. Conclusions

between the construction industry actors, whilst addressing the procurement challenges identified in Section 1.2. The architecture provided certain distinctive features which meant it could provide a new federated information sharing model to enable a new form of collaboration between construction industry actors such as product suppliers and contractors. The distinctive features of the MDSSF architecture are presented in Section 7.4.2 and summarised here.

- MDSSF is designed to provide Grid technology support in a federated database environment.
- MDSSF enables product suppliers to compete with each other in a virtual market place, based on the product information they provide to the federation users (contractors).
- MDSSF adopts a different model of cooperation from traditional FDBS architectures. The MDSSF model of cooperation does not allow sharing of data between component DBSs supplying the data (i.e. between the SD systems of the product suppliers). In the MDSSF architecture product suppliers share their data with the contractors only.
- In MDSSF, unlike other information sharing systems (see Section 4.4) the sharing of data takes place between contractors and the supplier databases without the need to create federated, integrated or external schemas. The schema integration is not required in the MDSSF approach, because of the homogeneous nature of the product data which is exchanged in MDSSF.
- The MDSSF adopts a subscription based approach, which provides product suppliers with readily available mechanisms to manage their product data locally in a structured way by using the standard specifications of product classes to describe product features and in this way tackle the issue of heterogeneity.
- Standardisation plays a key role in the MDSSF architecture, to support data sharing in a standard and integrated way. This distinctive feature is achieved by using the concept of product classes. An important advantage of product classes is that they provide product suppliers with a readily available means to describe their product features in a standard and structured way and support its evolution.

8.4 Applicability of MDSSF in Other Domains

The MDSSF provides a way to enable collaboration between industry actors in the construction industry. However there is a possibility that this research can be useful in other domains. The components of the MDSSF or the methods developed to enable data definition, data management and data search can be used or applied in areas, where similar information is available from several information sources and is categorised according to predefined criteria and shared with its users. There are several domains where information on related products and services available from independent sources has to be accumulated, aggregated, compared and analysed to provide users with a coherent view. This view allows the users to make an informed choice by comparing the competency or suitability of products or services and choosing according to their requirements. Several organisations in industry sectors such as travel and tourism, financial services, automobile industry, insurance, healthcare and real estate provide information on products and services to their users. For example in the automobile industry, the components of the MDSSF architecture can be used to represent different kinds of new and old automobiles by using the constructs of a product class. Product classes can be defined to allow different vehicle attributes such as *make*, *model*, *car type*, *fuel type*, *mileage*, *age*, *engine size* and *safety features* to be represented using standard means through different types of specifications. Additionally, complex product attributes such as *safety features* having attributes such as *first aid box*, *airbags* and *seat belts* can be represented using a single *SpecificationGroup* specification to allow representation of the safety features of a vehicle as a single entity. Once these product classes are created, they can be used by sellers to create descriptions of vehicle features using standard methods. This will also allow buyers to search for vehicles having desired attributes. The components of MDSSF such as its Grid based MDSS can be useful in performing this search, when such information on vehicles is available from a large number of sellers.

There are several information gathering and product comparison organisations such as Compare.com [Com09], PriceRunner.com [Pri09] which regularly analyse information from different sources to provide users with a coherent view of similar products or services, but available from several competing providers. Thus, there is a possibility that approaches developed in this research have the potential to aid both information sharing

organisations and the organisations which gather and analyse information to improve or enhance collaboration. However further research is required to identify how the methods or the components of the MDSSF can be adapted to the requirements of these domains.

8.5 Future Work

The research presented in this thesis provides ample scope for further improvements in related directions to address other challenges. Chapter 7 evaluated the architectural components of the MDSSF architecture from many different perspectives and also highlighted the further work that is required to address its present limitations. Therefore one possible area of further research is addressing the current limitations of the MDSSF's architectural components. However, there are other research areas which have emerged based on the experiences gained from this research. One of these potential research areas is in the field of data management. We created a federated information sharing model to address procurement challenges and increase collaborative working. As part of future efforts, we aim to extend this research into collaborative environments and address the data management challenges associated with such environments. We now identify the challenges in this area.

Collaborative Problem-Solving Environments (PSEs) [Lee08], [Bas08] in the AEC industry and scientific domains are dynamic, geographically distributed, multi-institutional and multidisciplinary. Members from these different specialist organisations come together as virtual teams to collaboratively tackle challenges during a project's duration. Members of such transient teams are selected by their expertise and use information systems to represent and exchange their ideas and associated data with their peers in order to address the challenges. The ideas of the different members should mesh together in order to build a larger, coherent picture that represents the objective or outcome the team is trying to achieve. In the same manner, the information constructs which represent these ideas or manage information pertaining to the team's objectives or activities should also mesh together. However currently this is not often the case. A considerable amount of effort is spent addressing interoperability issues, such as resolving structural, semantic and other forms of conflict, before the existing information systems can be made to work together. The literature has a large number of

8. Conclusions

research articles that have appeared in the last two decades to address these issues, but they still persist [Bat86], [Chi08]. The current PSEs also face additional challenges. Currently, teams and their activities are dynamic in nature. They have to deal with constantly changing requirements, scope and broadening perspectives. Putting together the different pieces of information that take into account several different perspectives often requires re-arrangement and/or modification of existing information/ideas. New methods of doing things require new ways of managing information, in ways which adapt to the problem scenario and user requirements. Existing systems are built to a fixed structure and their semantics often fail to meet the evolving and dynamic requirements. Recent advances in software development methodologies propose agile techniques [Amb03], [Sho07] to build information systems. These techniques, although useful, are oriented towards conceptual modelling and software development only. Hence the existing complexities and lack of resources restrict a transient team's abilities to address the aforementioned challenges in short duration projects.

Sharing of data is a common practice, and is an important way of allowing different organisations to work together. Many organisations or institutions, which come together in a collaborative venture, have their own n-tier applications connected to back-end databases. These information systems and particularly the underlying data models are rigid in structure. Often structural changes are made to the underlying data models or altogether new data models are designed and implemented (which is a time consuming process) in order to meet the new requirements. Because the existing information systems are inflexible, they can only provide data which is in a format having a predefined set of relations and attributes. These data models cannot be used to meet the new data requirements without implementing data transformation algorithms, which interface them with the other set of data which may conform to a different data model used in a different member organisation. The occurrence of several types of heterogeneities in this process restricts members from making existing data readily available to other members. Therefore in many circumstances, they often resort to using word processors or spreadsheets to create, store and share information pertaining to several aspects of their collaborative venture. If these were stored using a structured approach, there is a higher potential for them to be readily available and more usable in the later stages of a project. Thus there is a need for new mechanisms to handle new

8. Conclusions

data requirements without necessarily making structural changes to the underlying data model.

Our future research activities aim to benefit the community by providing a new perspective to information management and representation. This new approach will not only provide the structural flexibility via the use of adaptable constructs, so that users can modify the constructs (as well as the information they hold) according to their needs, but will also have the semantic flexibility, which will allow users to represent their ideas in a manner, which closely resembles the problem scenario. An important benefit of this new approach will be that these different constructs (conforming to individual specifications or format and created or managed by different members in a team) can be interlinked and shared between team members. This will give members the freedom to represent their diverse (and evolving) ideas at the information level via diverse constructs and at the same time link them to build a coherent picture representing the different perspectives of different team members (and their contribution to the project) in a readily usable form at the information management level.

Appendix 1

Performance Criteria of the Hong Kong Based Study

The Hong Kong based study conducted by Kumaraswamy et al. [Kum00], and Kumaraswamy and Dissanayaka [Kum01] identified 11 key performance criteria for optimising project specific procurement. They are mentioned below.

1. Lower capital cost;
2. Lower life cycle costs;
3. Cost certainty;
4. Shorter pre-construction duration;
5. Time certainty;
6. Shorter construction duration;
7. Effective and efficient communication;
8. Higher quality
9. Effective and efficient decision making;
10. Dispute minimisation;
11. Overall client satisfaction (also including other aspects).

Appendix 2

Representation of Product Information in MDSSF

A2.1 Introduction

This appendix identifies how the product class concept can be used to represent product information in MDSSF architecture in its three database systems (PCD, SPCD and SD) developed as part of this research. A model of electric bed called *ETESMI/Plano*, which has different attributes (specifications), has been chosen as an example product to demonstrate the ability of MDSSF databases to handle complex product information (see Section 7.2.2). This model of electric bed product is identified from a product comparison chart document which provides information on different types of general purpose and critical care electric beds. The product comparison chart is compiled by the ECRI Institute (<http://www.ecri.org/>) - an independent and non-profit health services research agency.

The product comparison chart document was provided to the COVITE project team by its industrial partner ActivePlan Solutions Limited (APSL) [Aps09]. The aim of this appendix is to demonstrate how the attributes of the electric bed product can be represented as a product class in PCD and SPCD systems and its product description created in an SD system. The concept of product class is explained in greater detail in Chapter 5. The architecture of the three database systems is presented in Chapter 6. The *ETESMI/Plano* model of electric bed product is represented using a product class for the purpose of author's research only. This appendix is organised as follows. Section A2.2 identifies the specifications (attributes) and specification values of the *ETESMI/Plano* model of electric bed product from the product comparison chart document. In section A2.3, various specifications of the product and its values are represented using the product class concept. Sections A2.4 provides the stored procedure execution code, which were executed to create a product class for the *ETESMI/Plano* model of electric bed product. The stored procedure execution code in Section A2.5 shows how this product class can be subscribed in an SPCD system at a product supplier's end. Finally, the stored procedure execution code in Section A2.6 shows how *ETESMI/Plano* model of electric bed product description can be created in an SD system, which refers to its

Appendix 2: Representation of Product Information in MDSSF

product class subscribed in the SPCD system. Section A2.7 presents appendix summary.

A2.2 Product Attributes

Table 1: This table shows attributes (specifications) of the *ETESMI/Plano* model of electric bed product and its specification values.

Table 1

Electric Bed	
Model	ETESMI/Plano
WHERE MARKETED	Worldwide
FDA CLEARANCE	No
CE MARK (MDD)	Yes
TYPE	General purpose, acute care
PATIENT CONTROLS	
Type	Pendant, hand control box, handset
NUMBER	6,8,10
FUNCTIONS	All positions of the bed
NURSE CONTROLS	
Patient control lockouts	Yes
Walk-away down	No
Full-low indicator	No
CPR control	Yes
AUTOMATIC CONTOUR	Yes
RETRACTABLE	Yes
HYPEREXTENSION	No
TRENDELENBURG GAUGE	No
MANUAL CRANK FUNCTIONS	No
Crank Storage	NA
SPRING TYPE	
Length, cm (in)	200-210 (78.7 – 82.7)
OVERALL DIMENSIONS	
L x W, cm (in)	218 x 100 (83 x 39) 228 x 100 (90 x 39) (including bumper)
Height, cm (in)	39-89 (15.4 – 35.3)
SIDERAIL LENGTH(S)	
Fraction of overall	4/5
WEIGHT, kg (lb)	250(551.3)
CASTER DIAM, cm (in)	15(6)
CASTER FUNCTIONS	Break, steer, lock (total), anti-static
CENTRAL BRAKE SYSTEM	Yes
BUMPERS	Yes
REMOVABLE HEADBOARD	Yes
IV POLE STORAGE	No

Appendix 2: Representation of Product Information in MDSSF

IV POLE MOUNTS	2
BED SCALE	No
ELECTRICAL FEATURES	
Power required, VAC	230
Number of motors	3
Double insulation	Yes
Frame	
Grounded	Yes
Isolated (motor ground)	Yes
Nonconductive siderails	No
Isolated IV pole	
Isolated transformer	Yes
LISTED FOR USE WITH OXYGEN TENT	No
PURCHASE INFORMATION	
List price, standard configuration	Not specified
Warranty	1 year, labour and parts
Delivery time, ARO	Not specified
Year first sold	1992
Fiscal Year	January to December
OTHER SPECIFICATIONS	Range of accessories and features available.

A2.3 Representation of the Electric Bed Product Using a Product Class

This section describes how the specifications (attributes) of the *ETESMI/Plano* model of electric bed product presented in Table 1 (see Section A2.2) can be represented using the product class concept. A new product class called *Electric Bed* and its new version (i.e. version 1.0) is created in PCD system to represent *ETESMI/Plano* model of electric bed product and its different specifications (attributes). The attributes of the electric bed product have been categorised into the different specifications of the product class so that these can be represented in the PCD and SPCD system. This product shows the complexity of managing product data, as the *Electric Bed* product class version 1.0 has 42 unit specifications, 7 specification groups, 5 list specifications (out of which 2 list specifications have more than one version) and 2 sub-product class specifications. The following tables show how the product attributes are represented as product class specifications. Information on specification assignment (see Section 7.2.2.2) is also provided with the tables. The stored procedure execution code in Sections A2.4, A2.5 and A2.6 show how the specification assignment is performed when new specifications are created.

Appendix 2: Representation of Product Information in MDSSF

Table 2: This table identifies the top level product class name and provides a description of the product class.

Table 2

Product Class Name	Electric Bed
Product Class Description	This product class provides information on a range of electric beds.

Table 3: This table identifies the version ID and description of a particular version of a product class to represent a particular product i.e. *ETESMI/Plano* model of electric bed product in the present case.

Table 3

Product Class Name	Electric Bed
Product Class Version ID:	1.0
Product Class Version Description:	This version identifies specifications of ETESMI /Plano model of electric bed.

Table 4: The following table identifies a list of unit specifications assigned to *Electric Bed* product class version 1.0 (i.e. the product class for *ETESMI/Plano* model of electric bed product in the present case). The table also identifies specification values, description and measurement unit of individual specifications where relevant.

Table 4

Specification Name	Specification Value	Specification Description	Measurement Unit
Model	Plano		
Model Range	ETESMI		
FDA CLEARANCE	No		
CE MARK (MDD)	Yes		
AUTOMATIC CONTOUR	Yes		
RETRACTABLE	Yes		
HYPEREXTENSION	Yes		
TRENDELENBURG GAUGE	No		
MANUAL CRANK FUNCTIONS	No		

Appendix 2: Representation of Product Information in MDSSF

CRANK STORAGE	NA		
WEIGHT	250		Kilograms (Kg)
WEIGHT	551.3		Pound (lb)
CENTRAL BRAKE SYSTEM	Yes		
BUMPERS	Yes		
REMOVABLE HEADBOARD	Yes		
IV POLE STORAGE	No		
IV POLE MOUNTS	2		
BED SCALE	No		
LISTED FOR USE WITH OXYGEN TENT	No		
OTHER SPECIFICATIONS	Range of accessories and features available.		

Table 5: The specification *Where Marketed* is represented as a list specification as it is likely that there could be more than one value (such as names of countries) for this specification.

Table 5

List Specification Name:	WHERE MARKETED
List Specification Description:	A top level list specification.

Table 6: The following table identifies a new version of the *WHERE MARKETED* list specification (i.e. version 1.0). Presently there is only one value (i.e. *Worldwide*) for this list specification version, however it is likely that there could be more than one values for this specification, due to which this specification is represented as a list specification. The *WHERE MARKETED* list specification version 1.0 is assigned to the *Electric Bed* product class version 1.0.

Appendix 2: Representation of Product Information in MDSSF

Table 6

List Specification Name:	WHERE MARKETED
List Specification Version ID:	1.0
List Specification Version Description:	Provides information on places/countries where the product is marketed.
List Value	Measurement Unit
Worldwide	

Tables 7 and 8: In the following tables the specification *TYPE* is represented as a list specification because there are two values for this specification. Table 7 shows the top level *Type* list specification and Table 8 is the new version (i.e. version 1.0) of this list specification having two list values. The *Type* list specification version 1.0 is assigned to the *Electric Bed* product class version 1.0.

Table 7

List Specification Name:	TYPE
List Specification Description:	A top level specification.

Table 8

List Specification Name:	TYPE
List Specification Version ID:	1.0
List Specification Version Description:	Identified the type/usage of the ETESMI/Plano range of electric beds.
List Value	Measurement Unit
General purpose	
Acute care	

Tables 9 and 10: In the following tables the specification *PATIENT CONTROLS* is represented as a specification group because several specifications are categorised under this specification (see table 1). For this purpose a top level specification group *PATIENT CONTROLS* is created in Table 9. Table 10 shows the new version (i.e. version 1.0) of this specification group. The specification group *PATIENT CONTROLS* version 1.0 is assigned to the *Electric Bed* product class the version 1.0.

Table 9

Specification Group Name:	PATIENT CONTROLS
Specification Group Description:	A general description for this specification group.

Appendix 2: Representation of Product Information in MDSSF

Table 10

Specification Group Name:	PATIENT CONTROLS
Specification Group Version:	1.0
Specification Group Version Description:	A group of specification identifying the patient controls.

Table 11: The *TYPE* specification under the *PATIENT CONTROL* specification group (see Table 1) has a list of values, therefore it is represented as a list specification. An important point to consider here is that the *TYPE* list specification version 1.0 has already been created (see Table 7 & 8) to enable identification of the type information of the *ETESMI/Plano* model of electric bed. Therefore a second version (i.e. version 2.0) of this specification is created here and this version represents different types of patient controls (see Section 7.2.2.4). Table 11 shows version 2.0 of the *TYPE* list specification having a list of values. The *TYPE* list specification version 2.0 is assigned to the *PATIENT CONTROLS* specification group version 1.0.

Table 11

List Specification Name:	TYPE
List Specification Version ID:	2.0
List Specification Version Description:	Identified the type of patient controls for the ETESMI/Plano electric bed model.
List Value	Measurement Unit
Pendant	
Hand control box	
Handset (Nurse controls mounted at foot end)	

Tables 12 and 13 The specification *NUMBER* is represented as a list specification as it has more than one value (see Table 1). Table 12 shows the new list specification *NUMBER* and Table 13 is the new version (i.e. version 1.0) of the *NUMBER* list specification which is assigned to *PATIENT CONTROLS* version 1.0 specification group.

Table 12

List Specification Name:	NUMBER
List Specification Description:	A top level list specification.

Appendix 2: Representation of Product Information in MDSSF

Table 13

List Specification Name:	NUMBER
List Specification Version ID:	1.0
List Specification Version Description:	
List Value	Measurement Unit
6	
8	

Tables 14 and 15: The specification *FUNCTIONS* is represented as a specification group because several specifications are categorised under this specification (see Table 1). Table 14 shows the new *FUNCTIONS* specification group and Table 15 is the new version (i.e. version 1.0) of the *FUNCTIONS* specification group which is assigned to the *PATIENT CONTROL* version 1.0 specification group.

Table 14

List Specification Name:	FUNCTIONS
List Specification Description:	A top level list specification.

Table 15

List Specification Name:	FUNCTIONS
List Specification Version ID:	1.0
List Specification Version Description:	
List Value	Measurement Unit
High-low	
kneebreak	
backrest	

Tables 16 and 17: The specification *NURSE CONTROLS* is represented as a specification group because several specifications are categorised under this specification (see Table 1). Table 16 shows the new *NURSE CONTROLS* specification group and Table 17 is the new version (i.e. version 1.0) of this specification group, which is assigned to the *Electric Bed* version 1.0 product class.

Table 16

Specification Group Name:	NURSE CONTROLS
Specification Group Description:	A general description for this specification group.

Appendix 2: Representation of Product Information in MDSSF

Table 17

Specification Group Name:	NURSE CONTROLS
Specification Group Version:	1.0
Specification Group Version Description:	A group of specification identifying the nurse controls for the ETESMI/Plano model of electric bed.

Table 18: The *NURSE CONTROLS* specification group categorises a number of unit specifications (see Table 1). Table 18 shows these unit specifications which are assigned to *NURSE CONTROL* Version 1.0 specification group.

Table 18

Specification Name	Specification Value	Specification Description	Measurement Unit
Pt control lockouts	Yes		
Walk-away down	No		
Full-low indicator	No		
CPR control	Yes		

Table 19: A product class called *SPRING* is created for storing spring products in PCD and SPCD systems. Although the *SPRING* product class is a full fledged product class, it is considered a sub product class in context to the *Electric Bed* version 1.0 product class (see Section 5.3.5). Tables 19 and 20 create the new *SPRING* product class and its new version (i.e. version 1.0) respectively. This product class is assigned to *Electric Bed* version 1.0 product class.

Table 19

Product Class Name	SPRING
Product Class Description	A top level product class for spring products.

Table 20

Product Class Name	SPRING
Product Class Version ID:	1.0
Product Class Version Description:	A particular version of the spring product class.

Appendix 2: Representation of Product Information in MDSSF

Table 21: The specification *TYPE* represents type information of spring product and has two values (see Table 1). Therefore this specification is represented as a list specification. However two versions of this specification have already been created (see Tables 7, 8 and 11). Therefore a new version (i.e. *TYPE* version 3.0) is created and assigned to the *SPRING* product class version 1.0.

Table 21

List Specification Name:	TYPE
List Specification Version ID:	3.0
List Specification Version Description:	A list specification for holding different spring types.
List Value	Measurement Unit
Sheet metal	
Radiolucent backrest	

Table 22: This table shows a specification called *LENGTH* which is represented as a unit specification for identification of length information. This unit specification is assigned to the *SPRING* product class version 1.0. This specification is assigned twice to enable representation of two different types of measurement units which correspond to this unit specification.

Table 22

Specification Name	Specification Value	Specification Description	Measurement Unit
LENGTH	200-210		Centimetre
LENGTH	78.7–82.7		Inches

Tables 23, 24, 25 and 26: The specification *OVERALL DIMENSIONS* is represented as a specification group because several specifications are categorised under this specification (see Table 1). Table 23 shows the new *OVERALL DIMENSIONS* specification group and Table 24 is the new version (i.e. version 1.0) of this specification group, which is assigned to *Electric Bed* version 1.0 product class.

Table 23:

Specification Group Name:	OVERALL DIMENSIONS
Specification Group Description:	A top level general definition for specifying product dimensions.

Appendix 2: Representation of Product Information in MDSSF

Table 24:

Specification Group Name:	OVERALL DIMENSIONS
Specification Group Version:	1.0
Specification Group Version Description:	Specification Group version for specifying dimensions for electric beds

Tables 25 and 26: These tables show unit specifications *LENGTH*, *WIDTH* and *HEIGHT*, which are assigned to the *OVERALL DIMENSIONS* specification group version 1.0. These unit specifications have two set of values (see Table 1) and in each of these sets, there are two different types of measurement units which correspond to these unit specifications.

Table 25: Unit specifications for Specification Group *OVERALL DIMENSIONS*, version 1.0 (first set of values, in centimetres and inches).

Specification Name	Specification Value	Specification Description	Measurement Unit
LENGTH	218		Centimetre
WIDTH	100		Centimetre
HEIGHT	39-89		Centimetres
LENGTH	83		Inches
WIDTH	39		Inches
HEIGHT	15.4 – 35.3		Inches

Table 26: Unit Specifications for Specification Group *OVERALL DIMENSIONS*, version 1.0 (second set of values, in centimetres and inches).

Table 26

Specification Name	Specification Value	Specification Description	Measurement Unit
LENGTH	228	Including bumper length	Centimetre
WIDTH	100		Centimetre
HEIGHT	39-89		Centimetres
LENGTH	90	Including bumper length	Inches
WIDTH	39		Inches
HEIGHT	15.4 – 35.3		Inches

Appendix 2: Representation of Product Information in MDSSF

Tables 27, 28, 29 and 30 The specification *SIDERAIL LENGTH(S)* is represented as a specification group, because several specifications are categorised under this specification (see Table 1). Table 27 shows the new *OVERALL DIMENSIONS* specification group and Table 28 is the new version (i.e. version 1.0) of this specification group, which is assigned to *Electric Bed* version 1.0 product class.

Table 27

Specification Group Name:	SIDERAIL LENGTH(S)
Specification Group Description:	A top level specification group.

Table 28

Specification Group Name:	SIDERAIL LENGTH(S)
Specification Group Version:	1.0
Specification Group Version Description:	Specification group version for ETESMI/Plano electric bed product models.

Tables 29 and 30: These tables show *FRACTION OF OVERALL* list specification and its new version (i.e. version 1.0) which is assigned to the *SIDERAIL LENGTH(S)* version 1.0 specification group.

Table 29

List Specification Name:	FRACTION OF OVERALL
List Specification Description:	A top level list specification.

Table 30

List Specification Name:	FRACTION OF OVERALL
List Specification Version ID:	1.0
List Specification Version Description:	
List Value	Measurement Unit
4	
5	

Table 31: A product class called *CASTER* is created for storing caster products in PCD. Although the *CASTER* product class is a full fledged product class, it is considered a sub product class in context to the *Electric Bed* version 1.0 product class (see Section 5.3.5). Tables 31 and 32 show the new *CASTER* product class and its new version (i.e.

Appendix 2: Representation of Product Information in MDSSF

version 1.0) respectively. This product class is assigned to the *Electric Bed* version 1.0 product class.

Table 33 shows a unit specification called *DIAMETER* which is assigned to the *CASTER* product class version 1.0. There are two different types of measurement units, which correspond to this unit specification (see Table 1). In Tables 14 & 15, the list specification *FUNCTIONS* version 1.0 is already created, therefore in Table 34 a new version of this list specification (i.e. version 2.0) is created and assigned to the *CASTER* product class version 1.0.

Table 31

Product Class Name	CASTER
Product Class Description	A top level product class for caster products.

Table 32

Product Class Name	CASTER
Product Class Version ID:	1.0
Product Class Version Description:	A particular version of the caster product class.

Table 33

Unit Specifications for Product Class: CASTER , version 1.0			
Specification Name	Specification Value	Specification Description	Measurement Unit
DIAMETER	15		Centimetre
DIAMETER	6		Inches

Table 34

List Specification Name:	FUNCTIONS
List Specification Version ID:	2.0
List Specification Version Description:	Caster functions
List Value	Measurement Unit
Break	
Steer	
Lock (total)	
Anti-static (also track)	

Tables 35, 36 and 37: The specification *ELECTRICAL FEATURES* is represented as a specification group because several specifications are categorised under this specification (see Table 1). Table 35 shows the new *ELECTRICAL FEATURES* specification group and Table 36 is the new version (i.e. version 1.0) of this

Appendix 2: Representation of Product Information in MDSSF

specification group, which is assigned to *Electric Bed* version 1.0 product class. Table 37 shows a number of unit specifications which are assigned to the *ELECTRICAL FEATURES* version 1.0 specification group.

Table 35

Specification Group Name:	ELECTRICAL FEATURES
Specification Group Description:	A top level specification group.

Table 36

Specification Group Name:	ELECTRICAL FEATURES
Specification Group Version:	1.0
Specification Group Version Description:	Specification group version for specifying electrical features of ETESMI/Plano electric bed product models.

Table 37

Unit Specifications for Specification Group: ELECTRICAL FEATURES , version 1.0			
Specification Name	Specification Value	Specification Description	Measurement Unit
Power required	230		Volt (V)
Number of Motors	3		
Double Insulation	Yes		
Nonconductive siderails	No		
Isolated IV Pole	Yes		
Isolated transformer	Yes		

Tables 38, 39 and 40: The specification *FRAME* is represented as a specification group because several specifications are categorised under this specification (see Table 1). Table 38 shows the new *FRAME* specification group and Table 39 is the new version (i.e. version 1.0) of this specification group, which is assigned to the *ELECTRICAL FEATURES* version 1.0 specification group. Thus, in context to the *Electric Bed* product class version 1.0 the *FRAME* specification group is a sub-sub specification group. Table 40 shows the two unit specifications which are assigned to the *FRAME* version 1.0 specification group.

Appendix 2: Representation of Product Information in MDSSF

Table 38

Specification Group Name:	FRAME
Specification Group Description:	A top level specification group.

Table 39

Specification Group Name:	FRAME
Specification Group Version:	1.0
Specification Group Version Description:	Specification group version for specifying electrical features of ETESMI/Plano electric bed product models.

Table 40

Specification Name	Specification Value	Specification Description	Measurement Unit
Grounded	Yes		
Isolated (motor ground)	Yes		

Tables 41, 42 and 43 The specification *PURCHASE INFORMATION* is represented as a specification group because several specifications are categorised under this specification (see Table 1). Table 41 shows the new *PURCHASE INFORMATION* specification group and Table 42 is the new version (i.e. version 1.0) of this specification group, which is assigned to the *Electric Bed* version 1.0 product class. Table 43 shows a number of unit specifications which are assigned to the *PURCHASE INFORMATION* version 1.0 specification group.

Table 41

Specification Group Name:	PURCHASE INFORMATION
Specification Group Description:	A top level specification group.

Table 42

Specification Group Name:	PURCHASE INFORMATION
Specification Group Version:	1.0
Specification Group Version Description:	Specification group version for specifying purchase information of ETESMI/Plano electric bed product models.

Appendix 2: Representation of Product Information in MDSSF

Table 43

Unit Specifications for Specification Group: PURCHASE INFORMATION , version 1.0			
Specification Name	Specification Value	Specification Description	Measurement Unit
List price, standard configuration	Not specified		
Delivery time, ARO	Not specified		
Year first sold	1992		
Fiscal year	January to December		

A2.4 Creation of *Electric Bed* product class in the PCD System

Section A2.3 showed how different specifications (attributes) of *ETESMI/Plano* model of electric bed product can be represented using the *Electric Bed* version 1.0 product class. This section identifies the stored procedure execution code using which the *Electric Bed* version 1.0 product class was created in the PCD system. A total of 47 stored procedures were executed in the PCD system to create the *Electric Bed* version 1.0 product class and its different specifications, which are identified in Table 1 (see Section A2.2). However, only a selected list of the stored procedure execution code, their input parameters and output messages is presented in this section for brevity. The stored procedure execution code which is not provided in this section is similar to the code which is provided but executed with different input parameters to create different specifications for the product class. The source code of all the stored procedures of the PCD system is available in Appendix 5.

A2.4.1 Create product class *Electric Bed* and its version 1.0:

```

declare @IDProdClassDef bigint
declare @IDProcState tinyint
declare @message nvarchar(500)
exec proc_CreateNewProductClass
@ProdClassName      = 'Electric Bed',
@IDProdClassVer     = 1.0,
@ProdClassDesc      = 'Electric Beds for hospitals',
@ProdClassVerDesc   = 'Electric Beds for hospitals version 1.0',
@IDProdClassDef     = @IDProdClassDef      OUTPUT,
@IDProcState        = @IDProcState        OUTPUT,
@Message            = @message            OUTPUT

print 'IDProdClassDef: ' + cast(@IDProdClassDef as nvarchar)
print 'IDProcState: ' + cast(@IDProcState as nvarchar)
print @Message

```

Appendix 2: Representation of Product Information in MDSSF

Output Message:

IDProdClassDef: 10526

IDProcState: 0

Product Class Created Successfully. Product Class ID is: 10325.

Product Class Version is: 1.00.

A2.4.2 Create *Model* Specification and assign it to the *Electric Bed* Product Class Version 1.0.

The value of the input parameter *IDAssignToSpecTypeDef* of *proc_CreateNewSpecification* stored procedure is 10526. This is also the value of output parameter *IDProductClassDef* outputted by stored procedure *proc_CreateNewProductClass* executed in section A2.4.1. This means that the specification *Model* is being created as well as assigned to the *Electric Bed* product class version 1.0 in the following stored procedure. This scenario is also applicable in all the stored procedures executed in Section A2.4 where a specification is both created and assigned.

```
declare @IDProcState tinyint
declare @message nvarchar(500)
exec proc_CreateNewSpecification
@SpecName           = 'Model',
@IDAssignToSpecTypeDef = 10526,
@IDProcState        = @IDProcState OUTPUT,
@Message            = @Message OUTPUT
print 'IDProcState: ' + cast(@IDProcState as nvarchar)
print @Message
```

Output Message:

'IDProcState: 0

Specification Created Successfully. Specification ID is: 1004.

A2.4.3 Create *WHERE MARKETED* List Specification and its Version 1.0 and Assign it to the *Electric Bed* Product Class Version 1.0.

```
declare @IDListDef BIGINT
declare @IDProcState tinyint
declare @message nvarchar(500)
exec proc_CreateNewListSpecification
@ListName      = 'WHERE MARKETED',
@ListDesc      = 'A top level list specification',
@ListVerDesc   = 'Provides information on places/countries where the product is
marketed',
@ListValues    = 'Worldwide###',
@ListIDMeasUnits = '10418###',
```

Appendix 2: Representation of Product Information in MDSSF

```
@IDAssignToSpecTypeDef = 10526,  
@IDListDef             = @IDListDef   OUTPUT,  
@IDProcState          = @IDProcState  OUTPUT,  
@Message              = @message     OUTPUT  
  
print '@IDListDef: ' + cast(@IDListDef as nvarchar)  
print 'IDProcState: ' + cast(@IDProcState as nvarchar)  
print @Message
```

Output Message:

```
@IDListDef: 1083  
IDProcState: 0  
List Specification Created Successfully. List Specification ID is: 1072.  
List Specification Version is: 1.0.
```

A2.4.4 Create *PATIENT CONTROLS* Specification Group Version 1.0 and assign it to the *Electric Bed* product class version 1.0

```
declare @IDSpecGroupDef BIGINT  
declare @IDProcState tinyint  
declare @message nvarchar(500)  
exec Proc_CreateNewSpecificationGroup  
@SpecGroupName           = 'PATIENT CONTROLS',  
@SpecGroupDesc           = 'A general description for this specification group',  
@SpecGroupVerDesc        = 'A group of specifications identifying the patient  
controls',  
@IDAssignToSpecTypeDef   = 10526,  
@IDSpecGroupDef          = @IDSpecGroupDef   OUTPUT,  
@IDProcState             = @IDProcState      OUTPUT,  
@Message                 = @message          OUTPUT  
print '@IDSpecGroupDef: ' + cast(@IDSpecGroupDef as nvarchar)  
print 'IDProcState: ' + cast(@IDProcState as nvarchar)  
print @Message  
@IDSpecGroupDef: 10617  
IDProcState: 0
```

Output Message:

```
@IDSpecGroupDef: 10617  
IDProcState: 0  
Specification group created Successfully.  
Specification Group ID is: 10117. Specification Group Version is: 1.00.
```

A2.4.5 Create *TYPE* List Specification Version 1.0 and Assign it to the *Electric Bed* version 1.0 Product Class.

```
declare @IDListDef BIGINT  
declare @IDProcState tinyint  
declare @message nvarchar(500)
```

Appendix 2: Representation of Product Information in MDSSF

```
exec proc_CreateNewListSpecification
@ListName           = 'TYPE',
@ListDesc           = 'A top level list specification',
@ListVerDesc        = 'Identified the type/usage of the ETESMI/Plano range
of electric beds.',
@ListValues         = 'General purpose###Acute care',
@ListIDMeasUnits    = '10418###10418',
@IDAssignToSpecTypeDef = 10526,
@IDListDef          = @IDListDef OUTPUT,
@IDProcState        = @IDProcState OUTPUT,
@Message            = @message OUTPUT
print '@IDListDef: ' + cast(@IDListDef as nvarchar)
print 'IDProcState: ' + cast(@IDProcState as nvarchar)
print @Message
```

Output Message:

```
@IDListDef: 1084
IDProcState: 0
List Specification Version Created Successfully. List Specification ID is: 1073. List
Specification Version is: 1.00.
```

A2.4.6 Create a New Version of *TYPE* List Specification and Assign it to *PATIENT CONTROLS* Specification Group Version 1.0.

```
declare @IDListDef BIGINT
declare @IDProcState tinyint
declare @message nvarchar(500)
exec proc_CreateNewListSpecificationVersion
@IDList             = 1073,
@ListVerDesc        = 'Identified the type of patient controls for the
ETESMI/Plano electric bed model.',
@ListValues         = 'Pendant###Hand control box###Handset(Nurse
controls mounted at foot end)',
@ListIDMeasUnits    = '10418###10418###10418',
@IDAssignToSpecTypeDef = 10617,
@IDListDef          = @IDListDef OUTPUT,
@IDProcState        = @IDProcState OUTPUT,
@Message            = @Message OUTPUT

print '@IDListDef: ' + cast(@IDListDef as nvarchar)
print 'IDProcState: ' + cast(@IDProcState as nvarchar)
print @Message
```

Output Message:

```
@ IDListDef: 1085
IDProcState: 0
List Specification Version Created Successfully. List Specification ID is: 1073. List
Specification Version is: 2.00.
```

Appendix 2: Representation of Product Information in MDSSF

A2.4.7 Create *SPRING* Version 1.0 Product Class and Assign it to the *Electric Bed* version 1.0 Product Class.

The *SPRING* version 1.0 product class is a sub-product class in context to the *Electric Bed* version 1.0 product class.

```
declare @IDProdClassDef bigint
declare @IDProcState tinyint
declare @message nvarchar(500)
exec proc_CreateNewProductClass
@ProdClassName          = 'SPRING',
@IDProdClassVer         = 1.0,
@ProdClassDesc          = 'A top level product class for spring products',
@ProdClassVerDesc      = 'A particular version of the spring product class.',
@IDAssignToSpecTypeDef = 10526,
@IDProdClassDef         = @IDProdClassDef    OUTPUT,
@IDProcState           = @IDProcState      OUTPUT,
@Message               = @message         OUTPUT

print 'IDProdClassDef: ' + cast(@IDProdClassDef as nvarchar)
print 'IDProcState: ' + cast(@IDProcState as nvarchar)
print @Message
```

Output Message:

IDProdClassDef: 10527

IDProcState: 0

Product Class Created Successfully. Product Class ID is: 10326. Product Class Version is: 1.00.

A2.5 Subscription of *Electric Bed* product class in SPCD System

The stored procedures in SPCD system enable subscription of a product class at product supplier's end. This section identifies the stored procedure execution code using which the *Electric Bed* version 1.0 product class was subscribed in an SPCD system. A total of 47 stored procedures were executed to subscribe the *Electric Bed* Version 1.0 product class in an SPCD system. However, only a selected list of these stored procedures, their input parameters and output messages is presented in this section for brevity. The source code of all the stored procedures of the SPCD system is available in Appendix 5.

A2.5.1 Create *Electric Bed* Product Class Version 1.0.

```
declare @IDProcState tinyint
declare @message nvarchar(500)
exec proc_CreateNewProductClass
@IDProdClass          = 10325,
```

Appendix 2: Representation of Product Information in MDSSF

```
@IDProdClassVer      = 1.0,  
@IDProdClassDef      = 10526,  
@ProdClassName       = 'Electric Bed',  
@ProdClassDesc       = 'Electric Beds for hospitals',  
@ProdClassVerDesc    = 'Electric Beds for hospitals version 1.0',  
@IDAssignToSpecTypeDef = NULL,  
@IDProcState         = @IDProcState      OUTPUT,  
@Message              = @message          OUTPUT
```

```
print 'IDProcState: ' + cast(@IDProcState as nvarchar)  
print @Message
```

Output Message:

```
IDProcState: 0  
Product Class Downloaded Successfully. Product Class ID is: 10325.  
Product Class Version is: 1.00.
```

A2.5.2 Create *FDA CLEARANCE* Unit Specification and Assign it to *Electric Bed* Product Class Version 1.0.

```
declare @IDProcState tinyint  
declare @message nvarchar(500)  
exec Proc_CreateNewSpecification  
@IDSpec              = 1006,  
@SpecName            = 'FDA CLEARANCE',  
@SpecDesc            = NULL,  
@SpecValue           = NULL,  
@IDMeasUnit          = NULL,  
@IDAssignToSpecTypeDef = 10526,  
@IDProcState         = @IDProcState OUTPUT,  
@Message              = @Message OUTPUT
```

```
print 'IDProcState: ' + cast(@IDProcState as nvarchar)  
print @Message
```

Output Message:

```
IDProcState: 0  
Specification Downloaded Successfully.  
Specification ID is: 1006.
```

A2.5.3 Create *WHERE MARKETED* List Specification Version 1.0 and Assign it to the *Electric Bed* Product Class Version 1.0.

```
declare @IDListDef BIGINT  
declare @IDProcState tinyint  
declare @message nvarchar(500)  
exec proc_CreateNewListSpecification  
@IDList              = 1072,  
@IDListVer           = 1.0,
```

Appendix 2: Representation of Product Information in MDSSF

```
@IDListDef          = 1083,  
@ListName           = 'WHERE MARKETED',  
@ListDesc           = 'A top level list specification',  
@ListVerDesc        = 'Provides information on places/counteres where the  
                      product is markedt',  
@ListValues         = 'Worldwide###',  
@ListIDMeasUnits    = '10418###',  
@IDAssignToSpecTypeDef = 10526,  
@IDProcState        = @IDProcState  OUTPUT,  
@Message            = @message  OUTPUT
```

```
print '@IDListDef: ' + cast(@IDListDef as nvarchar)  
print 'IDProcState: ' + cast(@IDProcState as nvarchar)  
print @Message
```

Output Message:

IDProcState: 0

List Specification Created Successfully. List Specification ID is: 1072.

List Specification Version is: 1.00.

A2.5.4 Create *PATIENT CONTROLS* Specification Group Version 1.0 and Assign it to the *Electric Bed* Version 1.0.

```
declare @IDProcState tinyint  
declare @message nvarchar(500)  
exec Proc_CreateNewSpecificationGroup  
@IDSpecGroup          = 10117,  
@IDSpecGroupVer       = 1.0,  
@IDSpecGroupDef       = 10617,  
@SpecGroupName        = 'PATIENT CONTROLS',  
@SpecGroupDesc        = 'A general description for this specification group',  
@SpecGroupVerDesc     = 'A group of specifications identifying the patient  
controls',  
@IDAssignToSpecTypeDef = 10526,  
@IDProcState          = @IDProcState  OUTPUT,  
@Message              = @message  OUTPUT  
print 'IDProcState: ' + cast(@IDProcState as nvarchar)  
print @Message
```

Output Message:

IDProcState: 0

Specification group downloaded Successfully. Specification Group ID is: 10117.

Specification Group Version is: 1.00.

A2.5.5 Create *TYPE* List Specification Version 2.0 and Assign it to *PATIENT CONTROLS* Specification Group Version 1.0.

```
declare @IDListDef BIGINT  
declare @IDProcState tinyint
```

Appendix 2: Representation of Product Information in MDSSF

```
declare @message nvarchar(500)
exec proc_CreateNewListSpecification
@IDList          = 1073,
@IDListVer       = 2.0,
@IDListDef       = 1085,
@ListName        = 'TYPE',
@ListDesc        = 'A top level list specification',
@ListVerDesc     = 'Identified the type/usage of the ETESMI/Plano range of
                    electric beds.',
@ListValues      = 'Pendant###Hand control box###Handset(Nurse controls
                    mounted at foot end)',
@ListIDMeasUnits = '10418###10418###10418',
@IDAssignToSpecTypeDef = 10617,
@IDProcState     = @IDProcState OUTPUT,
@Message         = @message OUTPUT

print '@IDListDef: ' + cast(@IDListDef as nvarchar)
print 'IDProcState: ' + cast(@IDProcState as nvarchar)
print @Message
```

Output Message:

```
@IDListDef: 1085
IDProcState: 0
List Specification Created Successfully. List Specification ID is: 1073. List
Specification Version is: 2.00.
```

A2.6 Creation of *ETESMI/Plano* Model of Electric Bed in SD System

Section A2.2 identified how different specifications (attributes) of *ETESMI/Plano* model of electric bed can be represented using the product class concept. This product class is created in the PCD system and subscribed in an SPCD system at product supplier's end. After a product class is subscribed, it can be referred to create product description in an SD system. In this section, the product description of *ETESMI/Plano* model of electric bed is created in the SD system which refers to the *Electric Bed* product class version 1.0 the SPCD system. A total of 47 stored procedures were executed to create this product description in the SD system. However, only a selected list of these stored procedures, their input parameters and output messages is presented in this section for brevity. In SD system, the entities such as products, categories and specifications are identified via a globally unique identifier (GUID). The source code of all the stored procedures of SD system is available in Appendix 5.

Appendix 2: Representation of Product Information in MDSSF

A2.6.1 Create *Electric Bed* Product.

This product conforms to the *Electric Bed* product class version 1.0.

```
declare @IDProcState tinyint
declare @message nvarchar(500)
exec proc_CreateNewProduct
@IDProdClass          = 10325,
@IDProdClassVer       = 1.0,
@ProdName             = 'Electric Bed',
@ProdDesc             = 'This version identifies specification of Plano model of
                        range ETESMI',
@AssignTo             = 'Category',
@IDAssignTo           = '4F750336-5949-4C38-A029-87180D22277D',
@IDProcState          = @IDProcState OUTPUT,
@Message              = @message OUTPUT

print 'IDProcState: ' + cast(@IDProcState as nvarchar)
print @Message
```

Output Message:

```
IDProcState: 0
Product Created Successfully.
Electric bed Product GUID is: 3D38E0BE-2F4F-4526-AA7D-96CF1DA0176B
```

A2.6.2 Create *Model* Unit Specification (object) and Assign it to the *Electric Bed* product.

```
declare @IDProcState tinyint
declare @message nvarchar(500)
exec proc_CreateNewProductSpecificationObject
@ProdSpecName        = 'Model',
@ProdSpecValue       = ETESMI/Plano,
@ProdSpecDesc        = NULL,
@MeasUnitName        = 'Unspecified',
@IDProd              = '3D38E0BE-2F4F-4526-AA7D-96CF1DA0176B',
@IDProcState         = @IDProcState OUTPUT,
@Message              = @Message OUTPUT
print 'IDProcState: ' + cast(@IDProcState as nvarchar)
print @Message
```

Output Message:

```
IDProcState: 0
Specification created Successfully. Specification
GUID is: B5BFF072-BFDA-4BAC-9D36-689F78834C34
```

A2.6.3 Create *TYPE* List Specification (Object) and Assign to the *Electric Bed* product

Appendix 2: Representation of Product Information in MDSSF

```
declare @IDProcState tinyint
declare @message nvarchar(500)
exec proc_CreateNewList
@IDListClass      = 1073,
@IDListVer        = 1.0,
@ListName         = 'TYPE',
@ListDesc         = 'A top level list specification',
@ListValues       = 'General purpose###Acute care',
@ListIDMeasUnits = 'Unspecified',
@AssignTo         = 'Product',
@IDAssignTo       = '3D38E0BE-2F4F-4526-AA7D-96CF1DA0176B',
@IDProcState      = @IDProcState OUTPUT,
@Message          = @message OUTPUT

print 'IDProcState: ' + cast(@IDProcState as nvarchar)
print @Message
```

Output Message:

```
IDProcState: 0
List Specification Created Successfully.
List specification GUID is: 8AE79857-C64F-438A-8755-19D123DD9660
```

A2.6.4 Create *PATIENT CONTROLS* Specification Group (Object) and Assign it to the *Electric Bed* product

```
declare @IDProcState tinyint
declare @message nvarchar(500)
exec Proc_CreateNewSpecificationGroup
@IDSpecGroupClass = 10117,
@IDSpecGroupVer   = 1.0,
@SpecGroupName    = 'PATIENT CONTROLS',
@SpecGroupDesc    = 'A group of specifications identifying the patient controls',
@AssignTo         = 'Product',
@IDAssignTo       = '3D38E0BE-2F4F-4526-AA7D-96CF1DA0176B',
@IDProcState      = @IDProcState OUTPUT,
@Message          = @message OUTPUT
print 'IDProcState: ' + cast(@IDProcState as nvarchar)
print @Message
```

Output Message:

```
IDProcState: 0
Specification group created Successfully.
Specification group GUID is: 31FE9550-41E5-4D4A-A0B4-D5E7ADDF85E7
```

A2.6.5 Create *TYPE* List (Object) and Assign it to the *PATIENT CONTROLS* specification group (object).

```
declare @IDProcState tinyint
declare @message nvarchar(500)
```

Appendix 2: Representation of Product Information in MDSSF

```
exec proc_CreateNewList
@IDListClass      = 1073,
@IDListVer        = 2.0,
@ListName         = 'TYPE',
@ListDesc         = 'Identified the type/usage of the ETESMI/Plano range of
                    electric beds.',
@ListValues       = 'Pendant###Hand control box###Handset(Nurse controls
                    mounted at foot end)',
@ListIDMeasUnits  = 'Unspecified',
@AssignTo         = 'SpecificationGroup',
@IDAssignTo       = '31FE9550-41E5-4D4A-A0B4-D5E7ADDF85E7',
@IDProcState      = @IDProcState OUTPUT,
@Message          = @message OUTPUT

print 'IDProcState: ' + cast(@IDProcState as nvarchar)
print @Message
```

Output Message:

```
IDProcState: 0
List Created Successfully.
List GUID is: F865B8E8-5C7E-46AA-B322-E7B76FFA56A6
```

A2.7 Appendix Summary

This appendix showed how a real world product can be represented by using product class concept. The example used in this appendix is the *ETESMI/Plano* model of electric bed. The product class *Electric Bed* version 1.0 and its different specifications is created in the PCD system to represent this product. This product class is then subscribed in an SPCD system. The *ETESMI/Plano* model of electric bed product description is created in the SD system which refers to the *Electric Bed* product class version 1.0 in the SPCD system.

Information System Comparison Tables

This appendix provides information on information sharing architectures/models reviewed as part of this research and schema integration approaches. These information sharing architectures are organised into different categories according to their nature. The key features of these information sharing architectures are presented in a tabular format – one table for each category. Due to several columns occurring in these tables, which cannot fit into a single page, the tables are split across multiple pages. Where the tables are split, the first and the second column identifying the serial number and the name of the system architecture is repeated for ease of comparison. This appendix comprises of the following tables.

Table 1: Mediator and/or Wrapper Based Information Systems

Table 2. Schema Integration based Information Systems

Table 3. Grid based Information Systems

Table 4. Other Information Systems

Table 5. Schema Integration Approaches

Appendix 3: Information System Comparison Tables

**Table 1. Mediator and/or Wrapper Based Information Systems
Columns 1-5**

1	2	3	4	5
SL.	Systems/ Architectures	Description	Supports resource Autonomy?	Supports Het. resources?
1	TSIMMIS [Gar97]	A mediator based information integration system.	yes	Yes
2	DISCO [Tom98]	A distributed mediator based architecture for scaling access to heterogeneous information sources.	Yes	Yes
3	COIN [Bre97]	Provides approach to resolve semantic conflicts where the conflicts are not identified "a priori".	Yes	Yes
4	DIKE [Pal03]	Provides semi-automatic means for the construction of CISs.	Yes	Yes
5	MedMaker [Pap96]	Integration of heterogeneous information sources by declaratively specifying mediators.	#	Yes
6	XMLMedia [Gar99]	A FDBS mediator for federating heterogeneous data sources using a semi structured data model based on OEM [McH97].	Yes	Yes
7	XMF [Lee02]	A system to provide integrated view of information resources accessible via the web.	Yes	Yes
8	MIX [Bar99]	XML-based information mediator prototype	Yes	Yes
9	DIOM [Liu95],[Liu96],[Lee97]	A system to achieve interoperability by explicitly defining interfaces for information producers and consumers and matching them dynamically to meet information consumers' needs.	Yes	Yes
10	SIMS [Are93],[Are97]	Integrates information from databases and knowledge bases.	Yes	Yes
11	Infosleuth [Bay97]	A system of information sharing and processing in the dynamic and open environments of the WWW.	Yes	Yes
12	Carnot[Woe93],[Sin97]	An enterprise information integration system.	Yes	Yes
13	HERMES [Sub95]	A system for semantic integration of information from heterogeneous data sources and reasoning systems.	Yes	Yes
14	XML Data Integration System [Alm04]	An approach for mediation of XML data sources using XML as global schema. Data sources also provide data using XML schemas.	Yes	Yes
15	FIS [Bus99]	The report identifies concepts, terminology and reference architectures of FISs.	Yes	Yes

Appendix 3: Information System Comparison Tables

Table 1 (contd). Mediator and/or Wrapper Based Information Systems
Columns 1-5 (contd.)

1	2	3	4	5
SL.	Systems/ Architectures	Description	Supports resource Autonomy?	Supports Het. resources?
16	webFINDIT [Bou00], [Bou94]	A dynamic architecture for describing, locating and accessing data from web accessible databases.	Yes	Yes
17	Infomaster [Gen97]	An information integration system for accessing information sources accessible via the web.	Yes	yes
18	SPICE [Jon00], [Xu02]	The project aims to build a "catalog of life" of living organisms by providing integrated access to distributed taxonomic databases or GSDs.	Yes	yes
19	Garlic [Car95]	A system for managing heterogeneous multimedia and traditional data.	Yes	yes

Str = structured; semi-str = semi-structured, unstr = unstructured;
= Not known from reference.

Appendix 3: Information System Comparison Tables

Table 1 (contd). Mediator and/or Wrapper Based Information Systems
Columns 1, 2, 6-8

1	2	6	7	8
SL.	Systems/Architectures	Data Sources	Operational domain/ Main objective	Uses Ontologies
1	TSIMMIS [Gar97]	Str, semi-str & unstr	Integration of information from different sources which are related but "not-quite-the-same".	No
2	DISCO [Tom98]	Str	Heterogeneous data sources conforming to different data models, schema and semantics.	No
3	COIN [Bre97]	Str & semi-str	Integration of information sources accessible from the internet.	No
4	DIKE [Pal03]	Str	Friendly and flexible access to heterogeneous databases.	No
5	MedMaker [Pap96]	Str, semi-str, unstr & sources with changing schemas.	Integration of information sources which do not have well defined static schema.	No
6	XMLMedia [Gar99]	Str, unstr	Integration of multiple data sources in the internet or on intranets using XML protocols.	No
7	XMF [Lee02]	Str, semi-str	Integration of information resources accessible via the web.	No
8	MIX [Bar99]	Str, semi-str	Integration of data from web based sources belonging to different conceptual domains.	No
9	DIOM [Liu95],[Liu96],[Lee97]	Str, semi-str & unstr	Interoperability in environments with growing number of data sources.	No
10	SIMS [Are93],[Are97]	Str & KBs	Information integration in heterogeneous environments.	Yes
11	Infosleuth [Bay97]	Str, semi-str and unstr	Open, dynamic environment where applications do not have complete knowledge or control of resources.	Yes
12	Carnot[Woe93],[Sin97]	Str, semi-str, unstr, KBs & apps.	Enterprise environments for integrated management and access to information (available from files, apps, DBS, KBs, etc)	Yes
13	HERMES [Sub95]	Str, unstr & apps.	Provides methodology for integration of data sources and reasoning systems.	No
14	XML Data Integration System [Alm04]	Str	Mediation of heterogeneous XML data sources.	No
15	FIS [Bus99]	Str, semi-str & unstr	Reports developments in the field of data integration and defines criteria to characterise information integration systems.	Yes
16	webFINDIT [Bou00],[Bou94]	Str	Access to information sources available via the web.	No
17	Infomaster [Gen97]	Str, semi-str	Integration of information sources available via the web.	No
18	SPICE [Jon00],[Xu02]	Str	Integrated access to taxonomic databases or GSDs.	No
19	Garlic [Car95]	Str, semi-str, unstr, image & video data	Integrated access to heterogeneous data sources including databases, non-database information sources and multimedia data sources.	No

Str = structured; semi-str = semi-structured, unstr = unstructured; # = Not known from reference.

Appendix 3: Information System Comparison Tables

Table 1 (contd). Mediator and/or Wrapper Based Information Systems
Columns 1, 2, 9-12

1	2	9	10	11	12
SL.	Systems/Architectures	Mediator Based/ or use of mediators?	Schema Int. supported?	Provide res. wrappers?	Operations Supported
1	TSIMMIS [Gar97]	Yes	No	Yes	access
2	DISCO [Tom98]	Yes	No	Yes	access
3	COIN [Bre97]	Yes	No	Yes	access
4	DIKE [Pal03]	"mediator-like" architecture	No	Yes	access
5	MedMaker [Pap96]	Yes	No	Yes	access
6	XMLMedia [Gar99]	Yes	No.	Yes	access
7	XMF [Lee02]	Yes	Yes. Global schema defined in XML.	Yes	access
8	MIX [Bar99]	Yes	XML DTDs as structural description (i.e. schema) of mediator views.	Yes	access
9	DIOM [Liu95], [Liu96],[Lee97]	Yes	No	Yes	access
10	SIMS [Are93], [Are97]	Yes	No.	No	access, update
11	Infosleuth [Bay97]	Yes	No	No	access
12	Carnot[Woe93], [Sin97]	Yes (mediated access to resources)	No	No	access, update
13	HERMES [Sub95]	Yes	No	No	access
14	XML Data Integration System [Alm04]	Yes	Yes, Global schema defined in XML	No	access
15	FIS [Bus99]	Yes	No	Mediator based FIS types use wrappers	access, update
16	webFINDIT [Bou00], [Bou94]	No	No	Yes	access
17	Infomaster [Gen97]	No	No	Yes	access
18	SPICE [Jon00], [Xu02]	No	No	Yes	access
19	Garlic [Car95]	No	No	Yes	access, update

Appendix 3: Information System Comparison Tables

Table 1 (contd). Mediator and/or Wrapper Based Information Systems
Columns 1, 2, 13 & 14

1	2	13	14
SL.	Systems/ Architectures	Middleware/ or middleware functions	Data model
1	TSIMMIS [Gar97]	Lightweight OEM based mediators, wrappers, query language and wrapper and mediator generators.	OEM data model [Pap95].
2	DISCO [Tom98]	Wrapper interfaces, query optimiser, query processing semantics, etc.	ODMG data model [Cat00]
3	COIN [Bre97]	Context knowledge repository, mediation engine, Multi-database access engine.	Object-oriented data model of family Frame-Logic [Kif89].
4	DIKE [Pal03]	Inter-schema property extractor to address different types of heterogeneities. Data repository for representing description of information stored in input databases.	Relational
5	MedMaker [Pap96]	Mediators provide integrated views in OEM of information from underlying data sources.	OEM data model [Pap95].
6	XMLMedia [Gar99]	Three-layered architecture: translation layer (data sources & wrappers), mediation layer (for mediation capabilities) & coordination layer (client side tools).	XML
7	XMF [Lee02]	Mediators control wrappers and perform information integration.	XML
8	MIX [Bar99]	The MIX mediator and query processor resolves query by using mediator view definition.	XML
9	DIOM [Liu95], [Liu96], [Lee97]	System architecture conforms to I3 reference framework [Hul95].	ODMG-93 [Cat94] data model.
10	SIMS [Are93] [Are97]	Knowledge representation system, LIM & Prodigy analysis planner.	Relational & Loom [Loo06]
11	Infosleuth [Bay97]	Network of agents created using Java & domain models (ontologies)	Relational
12	Carnot [Woe93], [Sin97]	A distributed interpreter called ESS constructed using Rosette actor language [Tom88] & various other services in its five layer architecture.	The ESS distributed interpreter.
13	HERMES [Sub95]	Provides a general rule-based declarative language for defining mediators which access data sources and reasoning systems uniformly.	Hermes Mediator Language.
14	XML Data Integration System [Alm04]	The XML mediation layer.	XML
15	FIS [Bus99]	Federation layer of FIS provides uniform access language, schema, metadata, etc.	CDM
16	webFINDIT [Bou00], [Bou94]	webDDL for forming cluster of related information repositories. Service links for linking clusters.	CORBA [OMG06].
17	Infomaster [Gen97]	"Infomaster Facilitator" determines sources that contain the information necessary to answer user query. A KB contains rules and constraints describing information sources and translations among them.	Rules & constraints of the KB.
18	SPICE [Jon00], [Xu02]	The CAS acts as a server, formulates queries, assemble results, etc. CAS knowledge repository.	CORBA [OMG06].
19	Garlic [Car95]	Garlic Query Services and Runtime System & Metadata Repository.	ODMG-93 [Cat94] object model.

Appendix 3: Information System Comparison Tables

Table 1 (contd). Mediator and/or Wrapper Based Information Systems
Columns 1, 2 & 15

1	2	15
SL.	Systems/Architectures	Key feature(s)
1	TSIMMIS [Gar97]	A standard data model based on OEM and common query language; network of mediators which provide additional functionalities.
2	DISCO [Tom98]	Addresses issues such as "fragile mediator", "source capability" and "graceless failure" to handle increasing number of data sources.
3	COIN [Bre97]	During query time context mediators detects and resolve conflicts by comparing contexts associated with data sources.
4	DIKE [Pal03]	Extraction and exploitation of inter-schema knowledge for creating CIS, enrichment of schema descriptions, analysis of inter-schema properties to build integrated view of data available through schemas, etc.
5	MedMaker [Pap96]	Provides mechanism to handle schema evolution, structure irregularities etc. of underlying data sources.
6	XMLMedia [Gar99]	Provides the mechanism to merge structured and un-structured data sources. Provides facility to store retrieved data as XML documents in RDBMS for query and manipulation.
7	XMF [Lee02]	Information resources are described using XML (by their wrappers). Dynamic management of global schema which is also described using XML. Supports different types of wrappers for integrating different WWW information resources.
8	MIX [Bar99]	Data exchange and integration is fully managed using XML. XMAS provides grouping and ordering facilities to generate new XML "objects" from existing ones.
9	DIOM [Liu95] [Liu96] [Lee97]	Information systems interoperated without the need for global schema. Interoperability is achieved using a metadata catalog service.
10	SIMS [Are93],[Are97]	Processing of queries using knowledge of information sources stored in an ontology. Domain information such as available information sources, their structure, contents, etc described using Loom [Loo06].
11	Infosleuth [Bay97]	Specialised agents for services such as users, ontology, brokers, resources, data analysis, task execution and monitoring.
12	Carnot[Woe93] [Sin97]	Supports wide range of functions such as accessing heterogeneous legacy databases, semantic integration using ontologies, business process management, data integrity enhancement, analytical decision support, workflow management, knowledge discovery, etc.
13	HERMES [Sub95]	Supports incremental integration of new information systems. Mediators provide "sophisticated abilities to extract and produce new information from existing data".
14	XML Data Integration System [Alm04]	Global schema enables creation of homogenised view over heterogeneous XML data. Integration tool based on the XML Schema language [Fal04], [Tho04], [Bir06].
15	FIS [Bus99]	Identifies three-tier architecture of FIS.
16	webFINDIT [Bou00], [Bou94]	Allows grouping of related information repositories (as clusters) and establish inter-relationship between them.
17	Infomaster [Gen97]	Accesses heterogeneous information sources by giving the impression of a centralised and homogeneous information system. Enables creation of a virtual data warehouse.
18	SPICE [Jon00], [Xu02]	The approach adopts a CDM & all databases (via their wrappers) map to this common data model.
19	Garlic [Car95]	Query optimisation techniques for advanced query processing.

Appendix 3: Information System Comparison Tables

Table 1 (contd). Mediator and/or Wrapper Based Information Systems
Columns 1, 2, 16 & 17

1	2	16	17
SL.	Systems/ Architectures	Method(s) of resolving conflicts	Language for Data Access/Query
1	TSIMMIS [Gar97]	Wrappers perform query translation and convert results into appropriate OEM objects.	MSL, LOREL
2	DISCO [Tom98]	Maps Disco types to data source types. Mediators perform reformulation of queries into local schemas.	ODMG 2.0
3	COIN [Bre97]	Wrappers provide uniform protocol.	SQL, ODBC, HTML interface and QBE (Query-By-Example) interface.
4	DIKE [Pal03]	Wrappers for translating queries into the language supported by the local DBMS.	Web-based access layer to query available data and schemas using SQL.
5	MedMaker [Pap96]	The MSL resolves schema-domain mismatch, schematic discrepancy and supports schema evolution.	MSL
6	XMLMedia [Gar99]	Wrappers wrap data sources in XML or SQL and provide necessary translation facilities.	Object-Oriented SQL3 [Eis99], JDBC [Cat97] and XML-QL [Deu98].
7	XMF [Lee02]	Wrappers convert data into XML by using translators. XMF mediation rules resolve schematic conflicts by mapping global and local schemas.	XML based Xpath [Cla99] & API for Java based clients.
8	MIX [Bar99]	XML wrappers for different web sources which perform necessary translation.	XMAS & GUI based BBQ
9	DIOM [Liu95], [Liu96],[Lee97]	Wrappers provide translation facilities using metadata catalog.	DIOM IQL
10	SIMS [Are93], [Are97]	Reformulates queries expressed in terms of domain model to appropriate sub queries for various information sources.	Query language of Loom [Loo06] Knowledge representation system.
11	Infosleuth [Bay97]	Resource agents provide mapping from common ontology to resource schema.	Query specified in generic terms, but SQL used internally to represent queries over the ontologies. KQML [Fin94] also used internally.
12	Carnot [Woe93], [Sin97]	MIST performs semantic mediation among heterogeneous sources.	HTML, Motif & LDL++
13	HERMES [Sub95]	Via the semantic integration process & conflict handling toolkit.	Logical query language based on HERMES mediator language.
14	XML Data Integration System [Alm04]	Query translators use mappings described between global and local schemas to aid query processing.	Global queries specified as XML docs.
15	FIS [Bus99]	Application specific.	Application specific.
16	webFINDIT [Bou00], [Bou94]	Provides "interface-domain mappings" to map between clusters and database interface.	WWD-QL
17	Infomaster [Gen97]	Rules and constraints (stored in the KB) describe information sources and translation among them.	Form based interface via the web & programmatic access via ACL.
18	SPICE [Jon00], [Xu02]	The wrappers wrap the GSDs into the CDM to answer CAS queries.	Web browser-based access.
19	Garlic [Car95]	Repository wrappers translate between Garlic's protocols and repository's native protocols.	Object-oriented extension of SQL.

Appendix 3: Information System Comparison Tables

Table 2. Schema Integration based Information Systems
Columns 1-5

1	2	3	4	5
SL.	Systems/ Architectures	Description	Supports res. Autonomy?	Supports Het. resources?
20	Multibase [Hua94]	A Heterogeneous multidatabase management system.	#	Yes
21	Efendi [Rad95]	A federation services to couple heterogeneous database and file systems to create a FDBS.	Yes	Yes
22	Myraid [Hwa94] [Lim95]	A FDBS which provides "enterprise-wide" information integration of independently developed databases.	Yes	Yes
23	Pegasus [Ahm91]	A heterogeneous multidatabase management system.	Yes	Yes
24	HODFA [Kar95]	Provides an approach and a methodology for homogenisation of databases so that they can be managed using a single DBMS in a federation.	Yes	yes, but homogenises them.
25	IRO-DB [Gar95] [Fan98]	A heterogeneous federated database system.	Yes	Yes
26	DATAPLEX [Chu90]	A heterogeneous distributed database management system.	Yes	Yes
27	INFINITY [Här97]	A federated database system.	Yes	yes
28	Sheth and Larson's FDBS reference architectures [She90]	The survey paper identifies reference architectures for developing FDBS.	Yes	Yes
29	XML Data Integration with OWL [Leh04]	Provide an approach for XML data integration available via the web using Web Ontology Language OWL [Bec04].	#	Yes
30	MVDS [Duw96]	Supports integration of same information distributed across multiple sites in various ways individually tailored for different users and applications.	Yes	Yes

Str = structured; semi-str = semi-structured, unstr = unstructured;
= Not known from reference.

Appendix 3: Information System Comparison Tables

Table 2 (contd.). Schema Integration based Information Systems
Columns 1, 2, 6-9

1	2	6	7	8	9
SL.	Systems/ Architectures	Data Sources	Operational domain/ Main objective	Uses Ontologies	Mediator Based/ or use of mediators?
20	Multibase [Hua94]	Str	Integration of heterogeneous databases.	No	No
21	Efendi [Rad95]	Str & Unix File system.	Integration of heterogeneous databases in an enterprise.	No	No
22	Myraid [Hwa94] [Lim95]	Str	"enterprise-wide" information integration of independently developed database.	No	No
23	Pegasus [Ahm91]	Str	Access to heterogeneous database and non-database information sources.	No	No
24	HODFA [Kar95]	Str	Integration of autonomous "legacy localized databases" which are owned by different departments in an organisation.	No	No
25	IRO-DB [Gar95] [Fan98]	Str	Interoperability of pre-existing relational databases and new object-oriented databases.	No	No
26	DATAPLEX [Chu90]	Str	Sharing of data between heterogeneous database systems using a relational data model.	No	No
27	INFINITY [Här97]	Str	Integration of heterogeneous data from relational & object-oriented databases.	No	No
28	Sheth and Larson's FDBS reference architectures [She90]	Str	Discusses "the application of the federation concept for managing existing heterogeneous and autonomous DBSs" [She90].	No	No
29	XML Data Integration with OWL [Leh04]	Data sources described using XML schema	Integration of data sources available from the web via web forms or web services.	Yes	No
30	MVDS [Duw96]	Str	Interoperability in heterogeneous object-oriented multidatabase system.	No	No

Str = structured; semi-str = semi-structured, unstr = unstructured;
= Not known from reference.

Appendix 3: Information System Comparison Tables

Table 2 (contd.). Schema Integration based Information Systems

Columns 1, 2, 10-13

1	2	10	11	12	13
SL.	Systems/ Architectures	Schema Int. supported?	Provide res. wrappers?	Operations Supported	Middleware/ or middleware functions
20	Multibase [Hua94]	Yes	No	access, update	Schema editor for defining global schemas, global Schema repository, distributed cooperative computation model for query execution.
21	Efendi [Rad95]	Yes	No	access, update	Efendi federation kernel & schema integration facilities.
22	Myraid [Hwa94] [Lim95]	Yes. (The schema is represented as a set of integrated relations)	No	access, update	Query processing, global transaction management
23	Pegasus [Ahm91]	Yes	No	access, update	Cooperative information management layer for schema integration, global query processing, local query translation and transaction management.
24	HODFA [Kar95]	Yes	No	access, update	MASS extracts data from heterogeneous databases to a homogeneous database. Knowledge directory maintains information which aids database homogenisation.
25	IRO-DB [Gar95] [Fan98]	Yes	No	access, update	Three layer architecture supports for schema integration, global transaction management, concurrency control, recovery, distributed query decomposition and optimisation.
26	DATAPLEX [Chu90]	Yes	No	access, update	Common conceptual schema based on relational data model and definition of external schemas over it. External schemas describe user's views as subset of global conceptual schema.
27	INFINITY [Här97]	Yes	No	access, update	Federation layer provides schema architecture to resolve schematic and semantic heterogeneities in two steps.
28	Sheth and Larson's FDBS reference architectures [She90]	Yes. Loosely coupled FDBS supports multiple federated schemas & tightly coupled FDBS support one or more federated schemas.	No	access, update	Two important components applicable to all FDBS reference architectures identified are: processors and schemas.
29	XML Data Integration with OWL [Leh04]	Yes	No	access	OWL is used as global schema language for data integration and semantic mapping.
30	MVDS [Duw96]	Supports multiple integration views	No	access	MVDL provides semantically rich integration operators for resolving semantical and structural differences. The three interface architecture allows users to use Preferred DBMS or DDL.

Appendix 3: Information System Comparison Tables

Table 2 (contd.). Schema Integration based Information Systems
Columns 1, 2, 14 & 15

1	2	14	15
SL.	Systems/Architectures	Data model	Key feature(s)
20	Multibase [Hua94]	Relational	Provides high transparency of networks and local DBMSs for applications to access heterogeneous databases in network environment.
21	Efendi [Rad95]	ODMG-93 [Cat94] object model.	Integration of heterogeneous data, data migration, objects caching & redundancy control.
22	Myraid [Hwa94] [Lim95]	Relational	Supports creation of multiple federations. Transaction management provides two-phase locking protocol & a timeout mechanism for deadlock problem.
23	Pegasus [Ahm91]	object oriented as well as functional HOSQL	Access & manipulation of multiple autonomous heterogeneous distributed object-oriented, relational, and other information systems via a uniform interface.
24	HODFA [Kar95]	N.a. The paper describes architecture only.	Transformation of a collection of heterogeneous legacy information systems into homogeneous systems.
25	IRO-DB [Gar95] [Fan98]	ODMG-93 [Cat94] object model.	Achieves interoperability between pre-existing relational DBS and new object-oriented DBS. Object-oriented paradigm used both for describing heterogeneous DBS and accessing federated DBS.
26	DATAPLEX [Chu90]	Relational	Architecture provides fourteen "functionally-independent" modules with well defined interfaces to support any DBMS and file system.
27	INFINITY [Här97]	EXPRESS [ISO94a], [Wi98] data model of STEP [ISO94], [Gle89] standard & an access interface SDAI [ISO98]	The system provides not only a common data model but also a common schema structure to achieve complete homogeneity of the data models.
28	Sheth and Larson's FDBS reference architectures [She90]	The component schema (which is derived by translating local schemas) acts at the CDM of the FDBS.	Identifies various FDBS reference architectures, Five level FDBS schema architecture, FDBS evolution process, FDBD development tasks & methodology, FDBS operation tasks & case studies.
29	XML Data Integration with OWL [Leh04]	A graph structure which has items is used as data model for OWL instances.	Supports integration of data sources with XML schemas into a global schema defined using OWL.
30	MVDS [Duw96]	ODMG-93 [Cat94] object model.	The system enables creation of multiple integration views without first creating the entire global schema.

Appendix 3: Information System Comparison Tables

**Table 2 (contd.). Schema Integration based Information Systems
Columns 1, 2, 14 & 15**

1	2	16	17
SL.	Systems/ Architectures	Method(s) of resolving conflicts	Language for Data Access/Query
20	Multibase [Hua94]	"Query Translator" which converts standard SQL commands into local SQL command.	ESQL/C, ODBC API, Multibase-DIL
21	Efendi [Rad95]	Federation kernel decomposes global operations to operations of various DBS.	ODMG's ODL/OML/OQL
22	Myraid [Hwa94] [Lim95]	The relational to local query translation is performed by "gateways" which reside on local database schemas.	SQL Query Interface
23	Pegasus [Ahm91]	HOSQL provides type and function abstractions to deal with mapping & integration problems.	HOSQL
24	HODFA [Kar95]	The Coordinator DBMS manages homogenised member databases, resolves semantic conflicts and enable interoperability. Schemas of all member databases are based on the same data model of the coordinator DBMS.	N.a. The paper describes an architecture only.
25	IRO-DB [Gar95] [Fan98]	The Local layer provides local DBS adapters to address different types of heterogeneities and provides a standard ODMG interface.	OQL
26	DATAPLEX [Chu90]	The "translator" module translates between user SQL queries and native queries of the participating heterogeneous databases using a syntax transformation table and a set of rules.	SQL
27	INFINITY [Här97]	Schemas of heterogeneous data models are translated into "data-model homogenized" schemas described in EXPRESS data model by using a mapping language.	CORAL language (which is similar to SQL3 [Eis99])
28	Sheth and Larson's FDBS reference architectures [She90]	Transforming processors for transforming data and queries between different data models/schemas via mappings.	Query language of the FDBS's CDM or query language of external schema (if external schema is different from federated schema).
29	XML Data Integration with OWL [Leh04]	Semantic mapping features provided by the OWL are used for mapping heterogeneous data sources to the common global schema.	Xquery based SWQL.
30	MVDS [Duw96]	Source Schema Specification Interface maps source schema syntactic constructs and corresponding semantic content into the MVDS's intermediate CDM.	User's preferred query language. Translators map between MVDS's CDM to user specified DDL. The user specified DDL is attached to MVDS to adapt to new DBMS and DDL.

Appendix 3: Information System Comparison Tables

Table 3. Grid based Information Systems
Columns 1-5

1	2	3	4	5
SL.	Systems/ Architectures	Description	Supports res. Autonomy?	Supports Het. resources
31	OGSA-DAI [Atk05], [Ogs07]	A middleware system for exposing data sources to the Grid environment.	Yes	Yes
31	Spitfire [Gag02] [Spi03] [Bel02]	Provides middleware to access relational databases in Grid environment.	Yes	Yes
33	GeneGrid [Kel05] [Jit05]	A workflow based bioinformatics application for accessing heterogeneous applications and datasets in Grid environment.	Yes	Yes
34	Biodiversity World (BDW) [Pah06], [Pah06a]	A flexible and extensible Web Services and workflow based Grid PSE for solving problems in biodiversity and analysing biodiversity patterns.	Yes	Yes

Table 3 (contd.). Grid based Information Systems
Columns 1, 2, 6-8

1	2	6	7	8
SL.	Systems/ Architectures	Data Sources	Operational domain/ Main objective	Uses Ontologies
31	OGSA-DAI [Atk05], [Ogs07]	Str, file systems & indexed files	Access to data from different sources in a Grid environment.	Middleware can be customised to access ontology data in different formats.
31	Spitfire [Gag02] [Spi03] [Bel02]	Str	Access to RDBMS via standard protocols and well published interfaces in Grid environment.	No
33	GeneGrid [Kel05] [Jit05]	Bioinformatics apps & biological datasets	Conducting <i>in silico</i> experiments in biology for development of antibodies and new drugs.	No
34	Biodiversity World (BDW) [Pah06], [Pah06a]	Str, semi-str, & apps	Integrated access to widely dispersed and disparate resources and analytical tools & composition of these resources into workflows.	Yes

Str = structured; semi-str = semi-structured, unstr = unstructured;
= Not known from reference.

Appendix 3: Information System Comparison Tables

Table 3 (contd.). Grid based Information Systems
Columns 1, 2, 9-13

1	2	9	10	11	12	13
SL.	Systems/ Architectures	Mediator Based/ or use of mediators?	Schema Int. supported?	Provide res. wrappers?	Operations Supported	Middleware/ or middleware functions
31	OGSA-DAI [Atk05], [Ogs07]	Mediator based functionality can be built if required depending upon application domain requirements.	N.a.	Provides "data services" for users to access the underlying data sources via its operations.	access, update	Data operations specified using "perform documents" which are sent to "OGSA-DAI Core" which manages access to data resources and executes user commands.
31	Spitfire [Gag02] [Spi03] [Bel02]	No	N.a.	Provides "local layer service" for gaining access to RDBMS through JDBC [Cat97] and SOAP [Gra02] protocol.	access, update	Middleware's server component interfaces RDBMS and provides "Grid access" to the database(s).
33	GeneGrid [Kel05] [Jit05]	No	N.a.	GAM manages access to and integration of bioinformatics applications.	access, update	OGSA compliant architecture provides various components for creating integrated environment to provide access to diverse applications and databases.
34	Biodiversity World (BDW) [Pah06], [Pah06a]	No	N.a.	Yes	access	Triana workflow management system for composing and executing workflows [Tay03], Condor [Tha05] based Grid middleware, metadata repository, web-service based access to resource wrappers.

Appendix 3: Information System Comparison Tables

Table 3 (contd.). Grid based Information Systems
Columns 1, 2, 9-13

1	2	14	15
SL.	Systems/ Architectures	Data model	Key feature(s)
31	OGSA-DAI [Atk05], [Ogs07]	OGSA-DAI provides XML-based "perform documents" to access data sources in a standard way.	Provides access to data resources in Grid environment in consistent and data resource independent way, supports data integration, etc.
31	Spitfire [Gag02] [Spi03] [Bel02]	Client side API available for accessing databases through standard protocol and well-defined interfaces.	Access to RDBMS in Grid environment for Grid applications.
33	GeneGrid [Kel05] [Jit05]	Bioinformatics experiments represented in XML workflow documents.	Enables creation of a "Virtual Bioinformatics Laboratory" from diverse resources. Access to distributed computational resources for running compute-intensive experiments.
34	Biodiversity World (BDW) [Pah06], [Pah06a]	Provide datatypes to exchange of heterogeneous data in a standard way and hide data heterogeneity at the system level.	Supports different types of biodiversity experiments such as bioclimatic modelling, phylogenetics analysis, biodiversity richness analysis and conservation evaluation, etc.

Table 3 (contd.). Grid based Information Systems
Columns 1, 2, 9-13

1	2	16	17
SL.	Systems/ Architectures	Method(s) of resolving conflicts	Language for Data Access/Query
31	OGSA-DAI [Atk05], [Ogs07]	Supports data transformation using XSLT [Cla99a]. Users resolve all other conflicts.	Interaction with data services via the "perform documents" which can be generated by using OGSA-DAI's API in Java.
31	Spitfire [Gag02] [Spi03] [Bel02]	N.a.	Client applications via the use Spitfire API.
33	GeneGrid [Kel05] [Jit05]	Provides "linkers" for transforming results of one application or database operation into other formats.	Portal based access.
34	Biodiversity World (BDW) [Pah06], [Pah06a]	Data types hide data heterogeneity at system level, client side tools convert data into appropriate format for its use in workflow process.	User interaction via the Triana workflow management system.

Appendix 3: Information System Comparison Tables

Table 4. Other Information Systems
Columns 1-5

1	2	3	4	5
SL.	Systems/ Architectures	Description	Supports res. Autonomy?	Supports Het. resources?
35	InformationManifold [Lev96]	Access to heterogeneous informaiton sources available via the web.	Yes	Yes
36	DDXMI [Nam02]	A semi-automated means of integration of heterogeneous distributed databases using a metadata integration approach & XML Metadata Interchange (XMI) [XMI06] Specifications.	#	yes
37	MRDSM [Lit85]	A relational multidatabase system (loosely coupled FDBS).	Yes	Supports relational databases which may differ semantically or structurally.
38	Heimbigner and McLeod's FDBS Architecture [Hei85]	A FDBS architecture which supports a collection of independent databases in a loosely coupled federation.	Yes	No
39	Hsiao's Federated databases and systems (Data sharing Tutorial) [Hsi92]	The tutorial provides different approaches to data sharing in federated database environment.	Yes	Yes
40	E-Union [Kon04]	Provides a "platform" for Information sharing between E-commerce systems for construction material procurement.	Yes	Yes

Table 4 (contd.). Other Information Systems
Columns 1, 2, 6-8

1	2	6	7	8
SL.	Systems/ Architectures	Data Sources	Operational domain/ Main objective	Uses Ontologies
35	InformationManifold [Lev96]	Str	Access to heterogeneous information sources available via the web.	No
36	DDXMI [Nam02]	Str	Integration of heterogeneous data available in XML format from XML databases.	No
37	MRDSM [Lit85]	Str	Management of databases of MRDS (Multics Relational Data Store) [McJ06]	No
38	Heimbigner and McLeod's FDBS Architecture [Hei85]	Str	Partial and coordinated sharing among independent databases with minimum centralised control in an office information system environment.	No
39	Hsiao's Federated databases and systems (Data sharing Tutorial) [Hsi92]	Str	Interoperability of heterogeneous databases to meet organisational requirements.	No
40	E-Union [Kon04]	Str	Bring together different actors such as buyers and suppliers in construction supply chain via E-commerce systems.	No

Appendix 3: Information System Comparison Tables

Table 4 (contd.). Other Information Systems
Columns 1, 2, 9-12

1	2	9	10	11	12
SL.	Systems/ Architectures	Mediator Based/ or use of mediators?	Schema Int. supported?	Provide res. wrappers?	Operations Supported
35	InformationManifold [Lev96]	No	No	No	access
36	DDXMI [Nam02]	No	No	No	access
37	MRDSM [Lit85]	No	No	No	access, update
38	Heimbigner and McLeod's FDBS Architecture [Hei85]	No	No	No	access, update
39	Hsiao's Federated databases and systems (Data sharing Tutorial) [Hsi92]	No	No	No	access, update
40	E-Union [Kon04]	No	No	No	access

Table 4 (contd.). Other Information Systems
Columns 1, 2, 13 & 14

1	2	13	14
SL.	Systems/ Architectures	Middleware/ or middleware functions	Data model
35	InformationManifold [Lev96]	Information source description repository, Query Plan Generator, Query Execution Engine, etc.	Relational model but augmented with object-oriented features.
36	DDXMI [Nam02]	Generation of tool to perform metadata integration, which produces a DDXMI file. This file is then used to generate local queries for individual databases from user's master query.	N.a.The DDXMI file is an XML document.
37	MRDSM [Lit85]	Mltidatabase data manipulation language MDSL.	Relational
38	Heimbigner and McLeod's FDBS Architecture [Hei85]	Provides export and import schemas to specify which information to share. A federal dictionary for federation lifecycle management.	An object-oriented and semantic CDM based on the "event model" [Kin85].
39	Hsiao's Federated databases and systems (Data sharing Tutorial) [Hsi92]	Identifies five different approaches to database sharing. First is database conversion and four different mapping approaches based on database schema and transaction translation.	The tutorial proposes multimodal and multilingual capabilities in federation to support different types of DBMS and their languages.
40	E-Union [Kon04]	The web service based E-Union framework collects data from different E-commerce sites via XML based SOAP messaging protocol.	RDBMS

Appendix 3: Information System Comparison Tables

Table 4 (contd.). Other Information Systems
Columns 1, 2, 16 & 17

1	2	15	16	17
SL.	Systems/ Architectures	Key feature(s)	Method(s) of resolving conflicts	Language for Data Access/Query
35	InformationManifold [Lev96]	Supports separation of information source description and details on how to access them. Information source descriptions aid query generation, capability to query several information sources and combine their answers.	Interface programs for interacting with form based interfaces of WWW data sources which retrieve data based on input variables.	Web based interface, Query formulation using templates available for classes in the "world view" schema
36	DDXMI [Nam02]	Generation of a tool for meta-users to perform meta-data integration and store integration information in a file.	The DDXMI file provides meta-information about relationships among databases, mappings and function names for handling semantic and structural discrepancies.	XML Query Language Quilt [Cha00].
37	MRDSM [Lit85]	Supports management of databases with distinct schemas. Supports inter database queries, multiple queries which are repeated for databases, dynamic attributes, etc.	Via the use of semantic variables which aid to replace query elements (when the main query is divided into sub queries) to deal with objects named differently in different databases.	MDSL which is a SQL-like type language.
38	Heimbigner and McLeod's FDBS Architecture [Hei85]	Allows cooperation among independent systems, supports transaction sharing, negotiation between two DBS components for information sharing.	N.a. The architecture presented assumes a homogeneous data model.	Access to remote database operations by importing remote databases' export schemas into local systems.
39	Hsiao's Federated databases and systems (Data sharing Tutorial) [Hsi92]	Identifies five requirements for federated databases and systems: transparent access, local autonomy, multimodal and multilingual capabilities, multi-backend capability and efficient access and concurrency control.	Database conversion approach uses "converters" to identify semantic equivalence; In other approaches multimodal and multilingual capabilities are proposed to resolve different types of conflicts.	User's preferred language in case of database conversion approach; Also user's preferred language in other approaches except one approach which uses a universal data model and language.
40	E-Union [Kon04]	Framework joins together different construction material trading sites so that buyers can also view information provided by other websites. Enables information sharing between different sites. Allows registration of trading sites and material information. Provides material searching facility.	Material information from different sites is converted into the standard E-Union schema for data sharing.	Website based access.

Appendix 3: Information System Comparison Tables

Table 5. Schema Integration Approaches

SL.	Approach	Description
1	Progressive pairwise integration of database module schemas [Luk96]	[Luk96] describes a schema integration approach of progressive pairwise integration of database module schemas built by different designers. An information system of a large organisation consists of various software subsystems where each software subsystem provides a business function such as store control, sales, purchasing, book-keeping, production planning, production management etc. The software subsystems are individually designed by different designers. The schema integration approach presented in [Luk96] integrates module schemas (in a progressive pairwise fashion) into the main schema of the organisation where module schema corresponds to a software subsystem that provides a business function to an organisation.
2	Schema integration using collaboration [Bey97]	Beynon-Davies et al. [Bey97] discuss the importance of collaboration when performing schema integration. A demonstrator system called SISIBIS (Schema Integration System using the Issue Based Information System (IBIS)) which uses IBIS scheme to support collaborative database design is described. The IBIS project focuses on the role of computers in group work in the domain of Computer Supported Cooperative Work (CSCW) [Bey97]. The SISIBIS tool takes a collaborative and an interactive approach to represent equivalences between concepts represented in subschemas. It provides mechanisms for storing deliberations arising from collaboration in an issue base so that rationale behind integration decisions could be captured and stored.
3	Schema integration using standardised dictionary [Law01]	Schema integration requires the resolution of naming, structural and semantic conflicts. Lawrence and Barker [Law01] propose a strategy for schema integration by capturing data semantics using a standardised dictionary. It is argued that the integration process which is mostly manual can be increasingly automated using a standardized global dictionary which is organised as a hierarchy of concept terms. Concept terms are related using 'IS-A' relationship for modeling generalisation and specialisation and 'HAS-A' relationship for constructing component relationships. The dictionary is used as a binding between integration sites and eliminates naming conflicts and reduces semantic conflicts. Structural conflicts are resolved at query time by a query processor which translates from the semantic integrated view to structural queries. The major contribution of the work is a systemised method for capturing data semantics using a standardized dictionary and a model which uses this information to perform schema integration in relational databases.
4	View integration which preserves the semantics of updates [Vid94].	Vidal and Winslett [Vid94] propose a methodology of schema integration which preserves the semantics of updates during the view integration process. Schema transformation is guided by a set of transformation principles so that it can be realised in a safe an algorithmic way. The relationship between the original and transformed schema is formally specified by instance and update mappings. The instance mapping specifies how instances of the original schema are mapped to the instance of the transformed schema and vice-versa. The update mapping specifies how each update operation defined under the original schema is transformed into an update operation defined under the transformed schema. By adopting this approach the transformation preserves information and update semantics.

Appendix 3: Information System Comparison Tables

Table 5 (contd.). Schema Integration Approaches

SL.	Approach	Description
5	Assertion based approach for schema integration [Joh93]	A schema integration problem can also be analysed by using concepts from logic programming and deductive databases. This approach is suggested by Johannesson [Joh93] who identifies how equivalent constructs of conceptual schemas can be represented by integrating assertions. A method of integrating schemas automatically is proposed starting from a set of integration assertions. Integration assertions are used to describe correspondences between constructs in different conceptual schemas with corresponding information bases. It is possible to distinguish between two different types of integration assertions. The first type of assertions, called object equality assertions is used to express that different constants in different information bases denote the same entity. The second type of assertions, called extension relationship assertions, is used to describe set relationships between extensions of different predicate symbols, such as the extension of one predicate symbol being identical to, included in, or overlapping with the extension of another. For example the predicate eq(a,b) means that the entity denoted by "a" is identical to the entity denoted by "b". Here "eq" is a predicate symbol.
6	Model independent assertions for heterogeneous schema integration [Spa92]	Spaccapietra et al. [Spa92] also take assertion-based approach for heterogeneous schema integration in which the DBA points out corresponding elements in the schemas and defines the nature of correspondence between them. The interschema correspondence assertions are declarative statements that assert the existence of relationships of some form or another between two schemas. Assertions identify what semantic, descriptive and structural conflicts exist within the correspondence. The main contribution is a methodology for automatic resolution of structural conflicts by putting schema transformation knowledge into the integrator system.
7	Metadata and meta-meta information based schema integration [Tan00]	Tan et al. [Tan00] present a database schema integration approach that not only uses metadata but also uses meta-meta information to make schema integration possible. The solution requires a meta object facility that serves not only as a repository but also as a more feasible means of managing meta data. It is identified that meta objects such as names, types, structural relationships with other objects can be used to infer object equivalence or associations. In this approach database systems are wrapped as objects and the system is federated rather than the one that uses a global schema. The approach uses three repositories: the export, import and auxiliary schema repositories. An auxiliary schema defines extensions to the existing schemas for entities in the multidatabase that do not exist in the component databases. The repositories also provide program-data independence and extensibility. The schema integration facility manipulates the meta-objects to construct import schemas. Intelligent modules are used to form integration incubators to facilitate the evolution of composite schemas based on inferences, user assistance and other clues to establish correct mapping between schemas.
8	Four-layered schema integration architecture [Red94]	To achieve schema integration and database integration, the methodology adopted by Reddy et al. [Red94] uses a four-layered schema architecture consisting of local schemata, local object schemata, global schema and global view schemata. Each layer presents an integrated view of the concepts that characterise the layers below. The methodology involves acquisition of semantic knowledge relevant to objects of a local schema object. During knowledge acquisition process the attributes that define semantic meaning of local object property and their values are captured. Additionally the concepts of object equivalence class and property equivalence class are used to facilitate creation of integrated schema. The proposed methodology addresses various kinds of conflicts such as naming conflicts, scaling conflicts, type conflicts and data incompatibilities such as different levels of accuracy, asynchronous updates and lack of security.

Appendix 3: Information System Comparison Tables

Table 5 (contd.). Schema Integration Approaches

SL.	Approach	Description
9	Resolution of conflicts in MVDB federated system [Ben01]	Benchikha et al. [Ben01] propose a federated database system called MultiViewDataBase (MVDB) System that supports the concept of viewpoint to resolve naming, structural and semantic conflicts in a federated database system. It is argued that the schema can take the form of a specification which takes into account several points of view. Several points of view are required because a single abstraction which meets the need of all project participants is difficult to establish. Each point of view can represent an aspect of the data description held by an independent database called 'partial'. The entities of the universe of discourse are described in a multiple and complementary way by several partial DBs which share a basic description known as 'referential'. Each partial schema describes an aspect of the data or part of the data of the referential. These partial databases are integrated in the MVDB federated system.
10	Schema integration methodologies identified by Batini et al. [Bat86].	Causes of schema integration identified by Batini et al. [Bat86] include different perspectives of user groups in modeling the same object, equivalence among constructs of the model, incompatible design specifications, common concepts but having different representations and semantical and structural incompatibilities, etc. Batini et al. [Bat86] provides comparison of twelve different methodologies for schema integration which provide solutions to schema integration problems. For the sake of brevity they are not listed here. The fundamental problem identified by Batini et al. that is addressed by schema integration processes pertains to resolution the semantical and structural diversities of schemas.

Appendix 4
Setting-up MDSSF in a Distributed Environment

A4.1 Introduction

The new PSCD application is based on the MDSSF architecture. The testing of the PSCD application took place in a distributed computing environment in which several machines were used to test the functionality provided by its different components. The testing took place on the machines available in a local area network. A few machines from the Welsh e-Science Centre (WeSC) [Wel07] were also used for testing and development. Access to the WeSC machines for compiling, installing and running MDSSF components was achieved via the SSH secure shell. This appendix provides information on setting up the components of the MDSSF architecture in a distributed computing environment. This can be achieved by following the compiling and installation procedure described here. The installation procedure also involves installing several prerequisite software components which are also identified in this appendix. The appendix is organised as follows. Section A4.2 identifies the machines which were used to set-up the MDSSF system components in a distributed environment. Different machines were used to host its different components. Section A4.3 identifies the components and the machines on which these were deployed. Section A4.4 provides information of Web Service access URLs of MDSSF system components and section A4.5 describe the installation procedure of the system components.

A4.2 Machines used for Installing MDSSF System Components

The following seven machines were used for conducting MDSSF experiments. The MDSSF system components (see Section A4.3) were installed on these machines. Out of these seven machines, four machines were available in the local area network and three machines were provided by the WeSC.

1. tennis.cs.cf.ac.uk
2. cognac.cs.cf.ac.uk
3. violin.cs.cf.ac.uk
4. legend.cs.cf.ac.uk
5. arsenic.cs.cf.ac.uk (WeSC machine)
6. bouscat.cs.cf.ac.uk (WeSC machine)
7. agents-comsc.grid.cf.ac.uk (WeSC machine)

A4.3 MDSSF System Components

The MDSSF architecture comprises several individual system components. These system components are listed below. Figure A4.1 identifies the machines on which they were installed and run.

1. Database Search Service (DSS) System.
2. Master Grid Service (MGS)
3. Product Class Database (PCD) System
4. Supplier Database (SD) System
5. Supplier's Product Class Database (SPCD) System
6. Supplier Web Service
7. Metadata Repository

Appendix 4: Setting-up MDSSF in a Distributed Environment

MDSSF System Components	Installed in machine(s):
1. Database Search Service (DSS) System	bouscat.cs.cf.ac.uk, agents-comsc.grid.cf.ac.uk, violin.cs.cf.ac.uk, arsenic.cs.cf.ac.uk
2. Master Grid Service (MGS)	tennis.cs.cf.ac.uk
3. Product Class Database (PCD) System	tennis.cs.cf.ac.uk
4. Supplier Database (SD) System	tennis.cs.cf.ac.uk, cognac.cs.cf.ac.uk, legend.cs.cf.ac.uk
5. Supplier's Product Class Database (SPCD) System	tennis.cs.cf.ac.uk, cognac.cs.cf.ac.uk, legend.cs.cf.ac.uk
6. Supplier Web Service	tennis.cs.cf.ac.uk, cognac.cs.cf.ac.uk, legend.cs.cf.ac.uk
7. Metadata Repository	tennis.cs.cf.ac.uk

Figure A4.1 The MDSSF system components and the machines in which they are installed

A4.4 Web Service Access URLs

The components of the MDSSF architecture collaborated with other components in a distributed environment when performing a distributed database search to retrieve product data from several SD systems (supplier databases) in response to a contractor's query. In MDSSF, the components (such as the MGS) invoke the services provided by the other components (such as the DSS) via the Web Service mechanism by sending SOAP [Gra02] messages to these components (see Section 6.3). The PSCD web application submits a product search job to the MGS node by invoking its Web Service interface which is accessible through a Web Service URL. Similarly the MGS distributes the search jobs across a number of machines in a Grid environment by invoking the DSS systems Web Service interfaces through its Web Service URLs. Finally the DSS systems invoke several SD systems, which are also accessible through their Web Services interfaces having a Web Service URL. This section identifies the Web Service URLs of MDSSF system components.

A4.4.1 Product Class Database (PCD) System Component

Machine Name: tennis.cs.cf.ac.uk

Access URL: Not applicable.

A4.4.2 Supplier's Product Class Database (SPCD) and Supplier Database (SD) System Components

a) Machine Name: tennis.cs.cf.ac.uk

SD Web Service Access URL:

<http://131.251.42.40/SupplierApp/ProductService.asmx>

SPCD Access URL: Not applicable.

Appendix 4: Setting-up MDSSF in a Distributed Environment

b) Machine Name: cognac.cs.cf.ac.uk
SD Web Service Access URL:
<http://131.251.42.33/SupplierApp/ProductService.asmx>
SPCD Access URL: Not applicable.

c) Machine Name: legend.cs.cf.ac.uk
SD Web Service Access URL:
<http://131.251.49.72/SupplierApp/ProductService.asmx>
SPCD Access URL: Not applicable.

A4.4.3 Master Grid Service (MGS) System Component

a) Machine Name: tennis.cs.cf.ac.uk
Master Grid Web Service Access URL:
<http://131.251.42.40:8080/axis/servlet/AxisServlet/MGS>

A4.4.4 Database Search Service (DSS) System Component

a) Machine Name: bouscat.cs.cf.ac.uk
Access URL (GSH):
http://131.251.47.110:18080/ogsa/services/services/uk/co/activeplan/mdss_4/impl/DatabaseSearchImplProviderFactoryService

b) Machine Name: agents-comsc.grid.cf.ac.uk
Access URL (GSH):
http://131.251.128.7:18080/ogsa/services/services/uk/co/activeplan/mdss_4/impl/DatabaseSearchImplProviderFactoryService

c) Machine Name: violin.cs.cf.ac.uk
Access URL (GSH):
http://131.251.42.37:8080/ogsa/services/services/uk/co/activeplan/mdss_4/impl/DatabaseSearchImplProviderFactoryService

d) Machine Name: arsenic.cs.cf.ac.uk
Access URL (GSH):
http://131.251.42.93:8080/ogsa/services/services/uk/co/activeplan/mdss_4/impl/DatabaseSearchImplProviderFactoryService

A4.5 MDSSF System component Installation Instructions

This section provides instructions for installing and running the MDSSF system components. The source code of all the MDSSF system components is available in Appendix 5. The directory locations or the folder names of system code identified in this appendix refer to locations or folder names as it is available in the digital form.

A4.5.1 Software Tools

The installation instructions pertain to deploying and running the MDSSF components in the Windows platform. The software components were developed in the Windows XP Professional operating system environment. The MDSS system component of MDSSF can also be deployed in the Linux operating system by following the same procedure described here. The MDSS system component of the MDSSF architecture comprises one or more DSS systems and the MGS system deployed in individual machines (see Section 6.3).

Appendix 4: Setting-up MDSSF in a Distributed Environment

Download and install the following prerequisite software tools:

- Jakarta Ant 1.5.2 <http://ant.apache.org/>
- Apache Axis 1.1 Final <http://ws.apache.org/axis/>
- Jakarta Tomcat 4.1.24 <http://jakarta.apache.org/tomcat/>
- Globus Toolkit 3.0.2 (Core) <http://www-unix.globus.org/toolkit/downloads/>
- Java 2 SDK 1.4.1_02 <http://java.sun.com>
- Microsoft SQL Server 2000 <http://www.microsoft.com/sqlserver/2008/en/us/default.aspx>
- Microsoft Internet Information Server (IIS) (available as part of Windows XP professional OS)

It is not recommended to install any other version of the software and tools than the one listed above because of compatibility issues. Please review the documentation associated with the above software tools for their respective installation instructions and classpath settings.

A4.5.2 Document Conventions:

<ogsa_root> refers to the directory location where the Globus Toolkit 3.0.2 (Core) distribution is unpacked.

<tomcat_home> refers to the location of jakarta-tomcat-4.1.24 directory.

<dev_home> refers to the location of the “development” folder. The “development” folder contains all the source code for the DSS and MGS sub components of the MDSS system.

<mgs_root> refers to the location of Master Grid Service (MGS) folder.

A4.5.3 Compiling and deploying the DSS System

The steps identified in this section are performed for each of the machine where the DSS system is deployed. Figure A4.1 identifies a list of machines on which DSS system is deployed. The Grid-enabled DSS System performs the task of searching and retrieving product data from several supplier database (SD) systems via their web service interface. To invoke a supplier Web Service interface and request data from an SD system, the DSS System depends on Java proxies and skeletons for the supplier Web #service which is described using the WSDL (web service description language) [Gra02]. At the server side, the WSDL is generated by tools such as Apache Axis when an application is deployed as a web service. The client side binding for invoking the supplier Web Service can be created by using the Apache Axis *WSDL-to-Java* tool. This tool can be run by executing *org.apache.axis.wsd.WSDL2Java* command. To create bindings perform the following steps:

- a) Create a “development” folder. For example “C:\development” which will be referred as <dev_home>.
- b) Open DOS command prompt (start--> run--> cmd) and execute the following command from <dev_home> directory.

```
java org.apache.axis.wsd.WSDL2Java  
http://localhost/SupplierApp/ProductService.asmx?WSDL
```

The argument of the above command is the URL of Supplier web application WSDL file. It is assumed that the supplier web service is running in the local machine. If it is running in a remote machine then replace the above URL with the

Appendix 4: Setting-up MDSSF in a Distributed Environment

full machine name or IP address of the remote machine followed by the name of the web service and its WSDL. The above command creates a directory structure `uk\co\activeplan\SupplierWS` in the `<dev_home>` folder and all the stub files generated reside in the “SupplierWS” directory.

- c) Execute the following command from `<dev_home>` to compile the stub files:

```
javac uk\co\activeplan\SupplierWS\*.java
```

The above steps create the binding code to create and send SOAP messages to the Supplier Web Service for retrieving product data from an SD system. The binding code is used by the DSS component of MDSS System deployed in a Grid environment. To install the DSS system in a given machine as a Grid service perform the following steps:

- d) Copy `mdss` folder to `<dev_home>\uk\co\activeplan`.
- e) Compile the files in the `mdss` directory by running the following command from the `<dev_home>` directory.

```
javac uk\co\activeplan\mdss_4\*.java
```

- f) Create a java archive of all the classes in the `mdss` and `SupplierWS` directory. The following command is executed from the `<dev_home>` directory. The jar file is named as `mdss.jar` and it is created in the `<dev_home>` folder.

```
jar cvf mdss.jar uk\co\activeplan\mdss_4\*.class  
uk\co\activeplan\SupplierWS\*.class
```

- g) Create new folder “lib” in the `<dev_home>` directory and copy `mdss.jar` into the `lib` folder. The file `mdss.jar` contains all the java code of the DSS system which is exposed as a Grid service. The jar file is run through the `createBottomUpGridService` tool which comes as part of the Globus Toolkit 3.0.2 (core) distribution. The tool automatically generates the stubs, the service locators, the deployment descriptor fragment, and an operation provider that delegates its calls to the DSS system code when the Grid service is invoked for searching product supplier databases (SD systems).
- h) Copy file `build-tools-usage.xml` from `xmlFiles` folder into the `<dev_home>` folder. The `build-tools-usage.xml` file, among other things contains the location of `mdss.jar` file which is to be run through the `createBottomUpGridService` tool. In our case it is `<dev_home>\lib\mdss.jar`
- i) Copy file “build.properties” to `<dev_home>` directory. The `build.properties` file points to the location of `<ogsa_root>` directory.
- j) From the `<dev_home>` directory run the following command:

```
ant -f build-tools-usage.xml createBottomUpGridService
```

Appendix 4: Setting-up MDSSF in a Distributed Environment

This command generates all the files necessary for the deployment of the Grid service in the <dev_home>\bottomUpFiles directory.

- k) Create Grid Service Archive (GAR) of the files in the bottomUpFiles directory. The properties for creating the GAR file are set in the target “bottomUpGar” in the build-tools-usage.xml. The following command is used from <dev_home> directory to create the GAR file.

```
ant -f build-tools-usage.xml bottomUpGar
```

The above command creates mdss.gar in <dev_home>\bottomUpFiles directory. Now the DSS Grid service is ready for deployment.

- l) To deploy the DSS Grid service run the following command from <ogsa_root>.

```
ant deploy -Dgar.name=<dev_home>\bottomUpFiles\mdss.gar
```

- m) Now deploy the DSS Grid service in the tomcat container by running the following command from <ogsa_root> directory.

```
ant -Dtomcat.dir=<tomcat_home> deployTomcat
```

- n) After deploying the Grid service in the tomcat container, copy the folder “uk” from <dev_home> to <tomcat_home>\webapps\ogsa\WEB-INF\classes folder. Restart tomcat server and access the Grid service at the following URL.

<http://localhost:8080/ogsa/servlet/AxisServlet>

The above URL lists a number of Grid service currently deployed. The name of the DSS Service deployed is DatabaseSearchImplProviderFactoryService. For undeploying the DSS Grid service run the following command from <ogsa_root>\undeploy directory.

```
ant -f mdss-undeploy.xml
```

And then run the following command from <ogsa_root> to undeploy it from tomcat container.

```
ant -Dtomcat.dir=<tomcat_home> deployTomcat
```

A4.5.4 Complying and deploying the Master Grid Service (MGS)

Master Grid Service (MGS) is a Web Service which divides the total work to be done into roughly equal portions and allocates each proportion to individual Grid machines running DSS instances. The MGS accesses DSS instances using its Grid Service Handles (GSH) (i.e. its web service access URLs). The MGS is deployed in tennis.cs.cf.ac.uk machine. To compile and deploy the MGS, perform the following steps.

Appendix 4: Setting-up MDSSF in a Distributed Environment

- a) Create new folder "MGS" inside <dev_home> directory. The "MGS" folder is referred to as <mgs_root>.
- b) Copy "org" and "uk" folders from bottomUpfiles folder into the <mgs_root>.
- c) Copy "mgs_4" folder (the MGS system code folder) into the <mgs_root>\uk\co\activeplan\ folder.
- d) Copy deploy.wsdd and undeploy.wsdd into the <mgs_root>
Classes in the "org" and "uk" folder are already in the compiled form so there is no need to compile them but the java code in the mgs folder need compiling. Compile MGS using the following command from <mgs_root>.

```
javac uk\co\activeplan\mgs\*.java
```

- e) Create a java archive of classes in the mgs_4 folder and its dependent classes by executing the following jar command from <mgs_root>. The jar file is named as mgs.jar and it is created in the <mgs_root> directory.

- f)

```
jar cvf mgs.jar uk\co\activeplan\mdss_4\*.class  
uk\co\activeplan\mdss_4\bindings\*.class uk\co\activeplan\mdss\service\*.class  
uk\co\activeplan\mgs\*.class org\gridforum\ogsi\*.class
```

- g) MGS is now ready for deployment. We use AdminClient tool from Apache Axis distribution to deploy the MGS in the tomcat container. The deployment file deploy.wsdd contains details of service to be deployed. The following command is used from <mgs_root> to deploy the MGS service.

```
java org.apache.axis.client.AdminClient deploy.wsdd
```

For undeploying the MGS service run the following command from <mgs_root> directory.

```
java org.apache.axis.client.AdminClient deploy.wsdd
```

- h) Copy mgs.jar from <mgs_root> folder into <tomcat_home>\webapps\axis\WEB-INF\lib\ folder. Restart tomcat server and access MGS Service at the following URL.

<http://localhost:8080/axis/servlet/AxisServlet>

A4.5.6 Installing PCD, SPCD and SD Systems

The PCD, SPCD and SD database systems are available as backup copies. The backup copies contain database tables and database stored procedures and functions. Perform database restore operation using SQL Server Enterprise Manager utility in SQL Server 2000 RDBMS to install these database systems.

A4.5.7 Installing Product Supplier Web Service Application

A prototype product supplier Web Service application is available as part of a supplier application which can be installed and run in Microsoft IIS server in machines where

Appendix 4: Setting-up MDSSF in a Distributed Environment

SD systems are installed. The application is installed by web application installation steps in IIS server (refer to IIS server documentation). The supplier application exposes the SD system as a web service which is invoked by DSS Grid services to retrieve product data.

Appendix 5 Product Class Database (PCD) System Code

/*
1.1 Procedure Name: dbo.proc_CategoryIDGenerator

Database: PCD

Description:

CategoryIDGenerator procedure generates Category ID under which the products can be listed. The procedure takes one parameter which is the ID of the existing category. If no parameter is supplied then input parameter (@IDCategory) value is set to NULL. The ID is generated depending upon the user input. If the user supplies no parameter then a category ID is generated at the root level. If the parameter is supplied (which is the existing category or sub-category) then the new category is generated as the subcategory of supplied category/subcategory.

```
*/
CREATE PROCEDURE dbo.proc_CategoryIDGenerator
-- Input parameter.
@IDSuperCategory      NVARCHAR(255)  NULL,
@NextAvailableIDCategory NVARCHAR(255) OUTPUT,
@NextAvailableIDInternal INT          OUTPUT
AS
DECLARE
@maxIDInternal int,
@maxIDCategory NVARCHAR(255), -- Variable @maxCat stores the ID of category with highest value.
@superCategoryIDLength TINYINT, -- This variable stores the length of Category ID stored in variable @maxCat
@count TINYINT, -- Use in while loop to keep track of the number of times the loop has run or will run.
@NextIDCategory NVARCHAR(255) -- This variable stores the value of next Available Category/subcategory under given category.
IF @IDSuperCategory IS NULL
BEGIN
    IF NOT EXISTS(Select * from Category)
    BEGIN
        SELECT @NextAvailableIDCategory = '1'
        SELECT @NextAvailableIDInternal = 1
    END
    ELSE
    BEGIN
        SELECT @MaxIDInternal = MAX(IDInternal) FROM Category WHERE IDSuperCategory LIKE '0'
        Print 'Max Internal ID is: ' + CAST(@MaxIDInternal AS NVARCHAR)
        SELECT @MaxIDCategory = IDCategory FROM Category WHERE IDSuperCategory LIKE '0' AND IDInternal
        = @maxIDInternal
        Print 'MaxCat is: ' + @MaxIDCategory
        SELECT @NextAvailableIDCategory = (CAST(@MaxIDCategory AS INT) + 1)
        PRINT 'From CatIDGen Next Available Cat: ' + @NextAvailableIDCategory
        SELECT @NextAvailableIDInternal = @MaxIDInternal + 1
    END
END
--If User supplies and input parameter the following code is executed otherwise the procedure execution is --terminated here.
ELSE
BEGIN
    -- Check whether the super category supplied by the user exists.
    -- IF the supercategory exists, then @maxCat variable holds the max IDCategory under that super -- category.
    SELECT @MaxIDInternal = MAX(IDInternal) FROM Category WHERE IDSuperCategory LIKE @IDSuperCategory
    SELECT @maxIDCategory = MAX(IDCategory) FROM Category WHERE IDSuperCategory LIKE @IDSuperCategory
    AND IDInternal = @MaxIDInternal
    IF @maxIDCategory IS NOT NULL
    BEGIN
        PRINT 'LAST VALUE IN this Category IS: ' + @maxIDCategory
        SELECT @SuperCategoryIDLength = LEN(@maxIDCategory)
        SELECT @count = LEN(@maxIDCategory)
        PRINT 'Length is: ' + CAST(@SuperCategoryIDLength AS NVARCHAR)
        WHILE @count > 0
        BEGIN
            IF SUBSTRING (@maxIDCategory, @count, 1) LIKE '.'
            BEGIN
                PRINT 'Dot found at Position: ' + CAST(@count AS NVARCHAR)
                PRINT RIGHT(@maxIDCategory, (LEN(@maxIDCategory) - @count))
            END
        END
    END
END
```

Appendix 5: Product Class Database (PCD) System Code

```
        SELECT @NextIDCategory = CAST(RIGHT(@maxIDCategory, (LEN(@maxIDCategory)
        - @count) AS INT) + 1
        PRINT 'Category ID Available: ' + LEFT (@maxIDCategory, @Count) +
        @NextIDCategory
        SELECT @NextAvailableIDCategory = LEFT (@maxIDCategory, @Count) +
        @NextIDCategory
        SELECT @NextAvailableIDInternal = @MaxIDInternal + 1
        SELECT @Count = 0
    END
    ELSE
    BEGIN
        SELECT @Count = @Count - 1
    END
END
END
-- IF the supplied supercategory does not exists in the IDSuperCategory column of Category table
-- then it is checked whether
-- it exists in IDCategory column. It might happen that the supplied category might not have any
--sub categories defined under it therefore
-- it has no entry in IDSuperCategory column of Category table.

ELSE
BEGIN
    SELECT @MaxIDInternal = MAX(IDInternal) FROM Category WHERE IDCategory LIKE @IDSuperCategory
    SELECT @maxCat = MAX(IDCategory) FROM Category WHERE IDCategory LIKE @IDSuperCategory
    IF EXISTS(SELECT IDCategory FROM Category WHERE IDCategory LIKE @IDSuperCategory)
    BEGIN
        --PRINT 'This Category Exists'
        --PRINT 'Category ID Available: ' + @IDCategory + '.1'
        SELECT @NextAvailableIDCategory = @IDSuperCategory + '.1'
        SELECT @NextAvailableIDInternal = 1
    END
    --IF supplied category is not found either in IDSuperCategory or IDCategory columns
    --then the supercategory does not exists and therefore user cannot
    --a new category under the supplied category.
    ELSE
        PRINT 'Supplied Category Does not Exists'
END
END
GO
/*
1.2 Procedure Name: dbo.proc_InsertCategory
Database: PCD
Description:
This procedure is enables creation of new category values. It executes procedure CategoryIDGenerator1 to
generate vaues such as IDSuperCategory, NextAvailableIDCategory and NextAvailableIDCategoryInternal
and then uses these to create a new category in category table.
*/
CREATE PROCEDURE dbo.proc_InsertCategory
@IDSuperCategory NVARCHAR(255) = NULL,
@CategoryName NVARCHAR(255)
AS
DECLARE
    @NewCategoryID NVARCHAR(255),
    @NewIDInternal INT

    Exec CategoryIDGenerator1
    @IDSuperCategory = @IDSuperCategory,
    @NextAvailableIDCategory = @NewCategoryID OUTPUT,
    @NextAvailableIDInternal = @NewIDInternal OUTPUT

IF @IDSuperCategory IS NULL
BEGIN
    SELECT @IDSuperCategory = '0'
    PRINT 'INFO From InsertCategory Procedure'
```

Appendix 5: Product Class Database (PCD) System Code

```
PRINT 'Category ID is: ' + @NewCategoryID
PRINT 'SuperCategory ID is: ' + @IDSuperCategory
PRINT 'ID Internal is: ' + CAST(@NewIDInternal AS NVARCHAR)
Insert into Category Values(@NewCategoryID, @CategoryName, @IDSuperCategory, @NewIDInternal )
```

```
END
GO
```

```
/*
```

1.3 Procedure Name: dbo.proc_Assign

Database: PCD

Description:

This procedure assigns a specification by calling procedure proc_AssignSpecification as part of a transaction.

```
*/
```

```
CREATE Procedure dbo.proc_Assign
```

```
/* Param List */
```

```
@IDSpecTypeDef          BIGINT,
@IDAssignToSpecTypeDef  BIGINT,
@SpecType                NVARCHAR(4000) = NULL,
@IDMeasurementUnit      BIGINT = NULL,
@IDProcState             TINYINT OUTPUT,
@Message                 NVARCHAR(500) OUTPUT
```

```
AS
```

```
BEGIN
```

```
    BEGIN TRAN
```

```
    DECLARE
```

```
        @Error          INT,
        @IDEntityPart1  INT /* Holds EntityPart of @IDSpecTypeDef */
```

```
    SELECT @IDProcState = 0
```

```
    SELECT @ERROR = 0
```

```
    SELECT @IDEntityPart1 = dbo.fn_getIDEntityPart(@IDSpecTypeDef)
```

```
/* Assigning a Specification */
```

```
IF @IDEntityPart1 = 100
```

```
    BEGIN
```

```
        EXEC proc_AssignSpecification
```

```
        @IDSpecification = @IDSpecTypeDef,
```

```
        @IDAssignToSpecTypeDef = @IDAssignToSpecTypeDef,
```

```
        @SpecificationValue = @SpecTypeValue,
```

```
        @IDMeasurementUnit = @IDMeasurementUnit,
```

```
        @IDProcState = @IDProcState OUTPUT,
```

```
        @Message = @Message OUTPUT
```

```
        IF @IDProcState != 0
```

```
            BEGIN
```

```
                ROLLBACK TRAN
```

```
                SELECT @Message = @Message + ' This Procedure was called from Proc_Assign.'
```

```
                RAISERROR(@Message,1,1) WITH SETERROR
```

```
                RETURN @@ERROR
```

```
            END
```

```
        ELSE
```

```
            BEGIN
```

```
                COMMIT TRAN
```

```
                RETURN
```

```
            END
```

```
    END
```

```
END
```

```
GO
```

```
/*
```

1.4 Procedure Name: dbo.proc_AssignCategory

Database: PCD

Description:

This procedure assigns a category to a super category and sub category.

```
*/
```

```
CREATE Procedure proc_AssignCategory
```

```
@IDCategory          BIGINT,
```

Appendix 5: Product Class Database (PCD) System Code

```
@IDSuperCategory    BIGINT,
@IDProcState        TINYINT      OUTPUT,
@Message            NVARCHAR(500) OUTPUT
AS
BEGIN
    DECLARE
        @Error        INT
    SELECT @IDProcState = 0
    SELECT @ERROR = 0
    INSERT INTO Category_SuperCategory (IDCategory, IDSuperCategory)
    VALUES (@IDCategory, @IDSuperCategory)
    SELECT @ERROR = @@ERROR
    IF @Error != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @IDProcState = 1
        SELECT @Message = 'An error occurred while creating the new Category.'
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Error occurred in Procedure proc_CreateCategory.'
        SELECT @Message = @Message + 'Procedure is terminated abnormally.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    INSERT INTO Category_SubCategory (IDCategory, IDSubCategory)
    VALUES (@IDSuperCategory, @IDCategory)
    SELECT @ERROR = @@ERROR
    IF @Error != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @IDProcState = 2
        SELECT @Message = 'An error occurred while creating the new category.'
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Error occurred in Procedure proc_CreateCategory.'
        SELECT @Message = @Message + 'Procedure is terminated abnormally.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        SELECT @Message = 'Category successfully assigned.'
        RETURN
    END
END
END GO
```

/* 1.5 Procedure Name: dbo. proc_AssignListSpecification

Database: PCD

Description:

This procedure enables assigning of list specification to a product class or specification group.

```
*/
CREATE Procedure proc_AssignListSpecification
/* Param List */
@IDListDef          BIGINT,
@IDAssignToSpecTypeDef  BIGINT,
@IDProcState        TINYINT      OUTPUT,
@Message            NVARCHAR(500) OUTPUT
AS
BEGIN
    DECLARE
        @Error        INT,
        @IDEntityPart2  INT /*Holds Entity part of @IDAssignToSpecTypeDef*/

    SELECT @IDProcState = 0
    SELECT @ERROR = 0
    SELECT @IDEntityPart2 = dbo.fn_getIDEntityPart(@IDAssignToSpecTypeDef)
```

Appendix 5: Product Class Database (PCD) System Code

```
/* A List Specification can be assigned to a product class or a specification group */
IF @IDEntityPart2 NOT IN (105,106)
BEGIN
    SELECT @IDProcState = 1
    SELECT @Message = 'A List Specification can only be assigned to a Product Class or a Specification Group. '
    SELECT @Message = @Message + 'The supplied specification type to which specification group needs to be
    assigned to is invalid. '
    SELECT @Message = @Message + 'Error occured in Procedure proc_AssignListSpecification. '
    SELECT @Message = @Message + 'Procedure is terminated abnormally. '
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END

/* Assigning List specification group to a product class*/
IF @IDEntityPart2 = 105
BEGIN
    INSERT INTO ProductClassDefinition(IDProdClassDef, IDListDef)
    VALUES (@IDAssignToSpecTypeDef, @IDListDef)
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @IDProcState = 2
        SELECT @Message = 'An error occured while assigning the List Specification to the product class.'
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '. '

        SELECT @Message = @Message + 'Error occured in Procedure proc_AssignListSpecification. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        SELECT @Message = 'List Specification successfully assigned to the Product Class.'
        RETURN
    END
END

END -- End of IF @IDEntityPart2 = 105

/* Assigning list specification to a specification Group */
IF @IDEntityPart2 = 106
BEGIN
    INSERT INTO SpecificationGroupDefinition(IDSpecGroupDef, IDListDef)
    VALUES (@IDAssignToSpecTypeDef, @IDListDef)
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @IDProcState = 3
        SELECT @Message = 'An error occured while assigning the List Specification to the specification
        group.'
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '. '
        SELECT @Message = @Message + 'Error occured in Procedure proc_AssignListSpecification. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        SELECT @Message = 'List Specification successfully assigned to the Specification Group.'
        RETURN
    END
END

END --End of IF @IDEntityPart2 = 106 */
END
GO
```

Appendix 5: Product Class Database (PCD) System Code

/*

1.6 Procedure Name: dbo. proc_AssignProductClass

Database: PCD

Description:

This procedure enables assigning of product class/ sub product class to a category, product class or a specification group.

*/

```
CREATE Procedure proc_AssignProductClass
/* Param List */
@IDProdClassDef          BIGINT,
@IDAssignToSpecTypeDef   BIGINT,
@IDProcState            TINYINT      OUTPUT,
@Message                NVARCHAR(500) OUTPUT
AS
BEGIN
    DECLARE
    @Error                INT,
    @IDEntityPart2       INT /*Holds Entity part of @IDAssignToSpecTypeDef*/
    SELECT @IDProcState = 0
    SELECT @ERROR = 0
    SELECT @IDEntityPart2 = dbo.fn_getIDEntityPart(@IDAssignToSpecTypeDef)

    /* A sub product class can be assigned to a category,product class or a specification group */
    IF @IDEntityPart2 NOT IN (102,105,106)
    BEGIN
        SELECT @IDProcState = 1
        SELECT @Message = 'A product class can only be assigned to a Category, Product Class or a Specification
        Group.'
        SELECT @Message = @Message + 'The supplied specification type to which Product Class needs to be assigned
        is invalid.'
        SELECT @Message = @Message + 'Error occured in Procedure proc_AssignProductClass.'
        SELECT @Message = @Message + 'Procedure is terminated abnormally.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END

    /* Assigning product class to a product class*/
    IF @IDEntityPart2 = 105
    BEGIN
        INSERT INTO ProductClassDefinition(IDProdClassDef, IDSubclassDef)
        VALUES (@IDAssignToSpecTypeDef, @IDProdClassDef)
        SELECT @Error = @@ERROR
        IF @Error != 0
        BEGIN
            ROLLBACK TRAN
            SELECT @IDProcState = 2
            SELECT @Message = 'An error occured while assigning the Subproduct class to the product class.'
            SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '.'
            SELECT @Message = @Message + 'Error occured in Procedure proc_AssignProductClass.'
            SELECT @Message = @Message + 'Procedure is terminated abnormally.'
            RAISERROR(@Message,1,1) WITH SETERROR
            RETURN @@ERROR
        END
    ELSE
    BEGIN
        SELECT @Message = 'Subproduct class successfully assigned to the product class.'
        RETURN
    END
END -- End of IF @IDEntityPart2 = 105

/* Assigning product class to a specification Group */
IF @IDEntityPart2 = 106
BEGIN
    INSERT INTO SpecificationGroupDefinition(IDSpecGroupDef, IDProdClassDef)
    VALUES (@IDAssignToSpecTypeDef, @IDProdClassDef)
    SELECT @Error = @@ERROR
    IF @Error != 0
```

Appendix 5: Product Class Database (PCD) System Code

```

BEGIN
    ROLLBACK TRAN
    SELECT @IDProcState = 3
    SELECT @Message = 'An error occurred while assigning the product class to the specification group.'
    SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '!'
    SELECT @Message = @Message + 'Error occurred in Procedure proc_AssignProductClass.'
    SELECT @Message = @Message + 'Procedure is terminated abnormally.'
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END
ELSE
BEGIN
    SELECT @Message = 'Product class successfully assigned to the specification group.'
    RETURN
END
END --End of IF @IDEntityPart2 = 106 */
/*Assigning product class to a category */

IF @IDEntityPart2 = 102
BEGIN
    INSERT INTO Category_ProductClass(IDCategory, IDProdClassDef)
    VALUES (@IDAssignToSpecTypeDef, @IDProdClassDef)
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @IDProcState = 4
        SELECT @Message = 'An error occurred while assigning the product class to the category.'
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '!'
        SELECT @Message = @Message + 'Error occurred in Procedure proc_AssignProductClass.'
        SELECT @Message = @Message + 'Procedure is terminated abnormally.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        SELECT @Message = 'Product class successfully assigned to the category.'
        RETURN
    END
END --End of IF @IDEntityPart2 = 102 */
END -- End of Proc_AssignProductClass Procedure.
GO

```

/*

1.7 Procedure Name: dbo. proc_AssignSpecification

Database: PCD

Description:

This procedure enables assigning of specification to product class or specification group.

*/

```

CREATE Procedure proc_AssignSpecification
/* Param List */
@IDSpec          BIGINT,
@IDAssignToSpecTypeDef  BIGINT,
@SpecValue       NVARCHAR(4000) = NULL,
@IDMeasUnit      BIGINT          = NULL,
@IDProcState     TINYINT        OUTPUT,
@Message         NVARCHAR(500)  OUTPUT
AS
BEGIN
    DECLARE
    @Error          INT,
    @IDEntityPart2 INT /*Holds Entity part of @IDAssignToSpecTypeDef*/
    SELECT @IDProcState = 0
    SELECT @ERROR = 0
    SELECT @IDEntityPart2 = dbo.fn_getIDEntityPart(@IDAssignToSpecTypeDef)

    /*A specification can only be assigned to a product class or a specification group */

```


Appendix 5: Product Class Database (PCD) System Code

```
IF @IDEntityPart2 NOT IN (105, 106)
BEGIN
    SELECT @Message = 'A specification can only be assigned to a Product Class or a Specification Group.'
    SELECT @Message = @Message + 'The supplied specification type to which specification needs to be assigned is
invalid.'
    SELECT @Message = @Message + 'Error occured in Procedure proc_AssignSpecification.'
    SELECT @Message = @Message + 'Procedure is terminated abnormally.'
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END

/* If a specification value is supplied then @IDMeasUnit should not be null. A specification
should correspond to a measurement unit. */
/*
IF @SpecValue IS NOT NULL
BEGIN
    IF @IDMeasUnit IS NULL
    BEGIN
        ROLLBACK TRAN
        SELECT @IDProcState = 3
        SELECT @Message = 'An error occured while creating the new specification.'
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
NVARCHAR) + '.'
        SELECT @Message = @Message + 'Error occured in Procedure proc_CreateNewSpecification.'
        SELECT @Message = @Message + 'Procedure is terminated abnormally.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
END

/*Correspondingly @IDMeasUnit should be accompanied by a @SpecVal.*/
IF @IDMeasUnit IS NOT NULL
BEGIN
    IF @SpecValue IS NULL
    BEGIN
        ROLLBACK TRAN
        SELECT @IDProcState = 4
        SELECT @Message = 'An error occured while creating the new specification.'
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
NVARCHAR) + '.'
        SELECT @Message = @Message + 'Error occured in Procedure proc_CreateNewSpecification.'
        SELECT @Message = @Message + 'Procedure is terminated abnormally.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
END

/* Assign Specification to a product class */
IF @IDEntityPart2 = 105
BEGIN
    INSERT INTO PCDSpecificationValue(IDProdClassDef, IDSpec, SpecValue, IDMeasUnit)
VALUES(@IDAssignToSpecTypeDef, @IDSpec, @SpecValue, @IDMeasUnit)
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @IDProcState = 1
        SELECT @Message = 'An error occured while assigning the specifiction to the product class.'
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
NVARCHAR) + '.'
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '.'
        SELECT @Message = @Message + 'Error occured in Procedure proc_AssignSpecification.'
        SELECT @Message = @Message + 'Procedure is terminated abnormally.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        SELECT @Message = 'Specification successfully assigned to the product class.'
```

Appendix 5: Product Class Database (PCD) System Code

```
        RETURN
    END

END

/* Assign Specification to a specification Group */
IF @IDEntityPart2 = 106
BEGIN
    INSERT INTO SGDSpecificationValue (IDSpecGroupDef, IDSpec, SpecValue, IDMeasUnit)
    VALUES(@IDAssignToSpecTypeDef, @IDSpec, @SpecValue, @IDMeasUnit)
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @IDProcState = 2
        SELECT @Message = 'An error occurred while assigning the specification to the specification group.'
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Error occurred in Procedure proc_AssignSpecification.'
        SELECT @Message = @Message + 'Procedure is terminated abnormally.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        SELECT @Message = 'Specification successfully assigned to the specification group.'
        RETURN
    END
END -- End of IF @IDEntityPart2 = 106
END
GO
```

1.8 Procedure Name: **dbo. proc_AssignSpecificationGroup**

Database: **PCD**

Description: This procedure enables assigning of specification group to product class or a specification group.

*/

```
CREATE Procedure proc_AssignSpecificationGroup
/* Param List */
@IDSpecGroupDef          BIGINT,
@IDAssignToSpecTypeDef  BIGINT,
@IDProcState             TINYINT          OUTPUT,
@Message                 NVARCHAR(500)   OUTPUT
AS
BEGIN
    DECLARE
    @Error                INT,
    @IDEntityPart2       INT /*Holds Entity part of @IDAssignToSpecTypeDef*/

    SELECT @IDProcState = 0
    SELECT @ERROR = 0
    SELECT @IDEntityPart2 = dbo.fn_getIDEntityPart(@IDAssignToSpecTypeDef)

    /* A specification group can be assigned to a product class or a specification group */
    IF @IDEntityPart2 NOT IN (105,106)
    BEGIN
        SELECT @IDProcState = 1
        SELECT @Message = 'A specification group can only be assigned to a Product Class or a Specification Group.'
        SELECT @Message = @Message + 'The supplied specification type to which specification group needs to be
        assigned to is invalid.'
        SELECT @Message = @Message + 'Error occurred in Procedure proc_AssignSpecificationGroup.'
        SELECT @Message = @Message + 'Procedure is terminated abnormally.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END

    /* Assigning specification group to a product class*/
```

Appendix 5: Product Class Database (PCD) System Code

```
IF @IDEntityPart2 = 105
BEGIN
    INSERT INTO ProductClassDefinition(IDProdClassDef, IDSpecGroupDef)
    VALUES (@IDAssignToSpecTypeDef, @IDSpecGroupDef)
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @IDProcState = 2
        SELECT @Message = 'An error ocured while assigning the specification group to the product class.'
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Error occurred in Procedure proc_AssignSpecificationGroup. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        SELECT @Message = 'Specification group successfully assigned to the Product Class.'
        RETURN
    END
END -- End of IF @IDEntityPart2 = 105

/* Assigning specification group to a specification Group */
IF @IDEntityPart2 = 106
BEGIN
    INSERT INTO SpecificationGroupDefinition(IDSpecGroupDef, IDSubSpecGroupDef)
    VALUES (@IDAssignToSpecTypeDef, @IDSpecGroupDef)
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @IDProcState = 3
        SELECT @Message = 'An error ocured while assigning the specification group to the specification
        group.'
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Error occurred in Procedure proc_AssignSpecificationGroup. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        SELECT @Message = 'Specification group successfully assigned to the Specification Group.'
        RETURN
    END
END --End of IF @IDEntityPart2 = 106 */
END -- End of Proc_AssignSpecificationGroup Procedure.
GO

/*
1.9 Procedure Name: dbo.proc_AssignTableSpecification
Database: PCD
Description: This procedure enables assigning of table specification to product class or specification group.
*/

CREATE Procedure proc_AssignTableSpecification
/* Param List */
@IDTableVerDef          BIGINT,
@IDAssignToSpecTypeDef  BIGINT,
@IDProcState            TINYINT      OUTPUT,
@Message                NVARCHAR(500) OUTPUT
AS
BEGIN
    DECLARE
    @Error          INT,
    @IDEntityPart2 INT /*Holds Entity part of @IDAssignToSpecTypeDef*/
    SELECT @IDProcState = 0
```

Appendix 5: Product Class Database (PCD) System Code

```
SELECT @ERROR = 0
SELECT @IDEntityPart2 = dbo.fn_getIDEntityPart(@IDAssignToSpecTypeDef)

/* A Table specification can be assigned to a product class or a specification group. */
IF @IDEntityPart2 NOT IN (105,106)
BEGIN
    SELECT @IDProcState = 1
    SELECT @Message = 'A Table Specification can only be assigned to a Product Class or a Specification Group.'
    SELECT @Message = @Message + 'The supplied specification type to which specification group needs to be
assigned to is invalid.'
    SELECT @Message = @Message + 'Error occured in Procedure proc_AssignTableSpecification.'
    SELECT @Message = @Message + 'Procedure is terminated abnormally.'
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END

/* Assigning Table Specification to a product class*/
IF @IDEntityPart2 = 105
BEGIN
    INSERT INTO ProductClassDefinition(IDProdClassDef, IDTableVerDef)
    VALUES (@IDAssignToSpecTypeDef, @IDTableVerDef)
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @IDProcState = 2
        SELECT @Message = 'An error occured while assigning the table specification to the product class.'
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '.'
        SELECT @Message = @Message + 'Error occured in Procedure proc_AssignTableSpecification.'
        SELECT @Message = @Message + 'Procedure is terminated abnormally.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        SELECT @Message = 'Table Specification successfully assigned to the Product Class.'
        RETURN
    END
END -- End of IF @IDEntityPart2 = 105

/*Assigning Table Specification to a specification Group */
IF @IDEntityPart2 = 106
BEGIN
    INSERT INTO SpecificationGroupDefinition(IDSpecGroupDef, IDTableVerDef)
    VALUES (@IDAssignToSpecTypeDef, @IDTableVerDef)
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @IDProcState = 3
        SELECT @Message = 'An error occured while assigning the table specification to the specification
group.'
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '.'
        SELECT @Message = @Message + 'Error occured in Procedure proc_AssignTableSpecification.'
        SELECT @Message = @Message + 'Procedure is terminated abnormally.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        SELECT @Message = 'Table Specification successfully assigned to the Specification Group.'
        RETURN
    END
END --End of IF @IDEntityPart2 = 106 */
-- End of proc_AssignTableSpecification Procedure.
END
GO
```

Appendix 5: Product Class Database (PCD) System Code

/*

1.10 Procedure Name: dbo.proc_CallAssignCategory

Database: PCD

Description:

This procedure calls proc_AssignCategory as a part of a transaction.

*/

```
CREATE Procedure proc_CallAssignCategory
/* Param List */
@IDCategory          BIGINT,
@IDSuperCategory     BIGINT,
@IDProcState         TINYINT OUTPUT,
@Message             NVARCHAR(500) OUTPUT
AS
BEGIN
    /* This procedure calls proc_AssignCategory as a part of a transaction.*/
    BEGIN TRAN
    EXEC proc_AssignCategory
    @IDCategory        = @IDCategory,
    @IDSuperCategory   = @IDSuperCategory,
    @IDProcState       = @IDProcState OUTPUT,
    @Message           = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from Proc_CallAssignCategory.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        COMMIT TRAN
        RETURN
    END
END -- End of proc_CallAssignCategory Procedure.
GO
```

/*

1.11 Procedure Name: dbo.proc_CallAssignProductClass

Database: PCD

Description:

This procedure calls proc_AssignProductClass as a part of a transaction.

*/

```
CREATE Procedure proc_CallAssignProductClass
/* Param List */
@IDProdClassDef      BIGINT,
@IDAssignToSpecTypeDef BIGINT,
@IDProcState         TINYINT OUTPUT,
@Message             NVARCHAR(500) OUTPUT
AS
BEGIN
    /* This procedure calls proc_AssignProductClass as a part of a transaction.*/
    BEGIN TRAN
    EXEC proc_AssignProductClass
    @IDProdClassDef    = @IDProdClassDef,
    @IDAssignToSpecTypeDef = @IDAssignToSpecTypeDef,
    @IDProcState       = @IDProcState OUTPUT,
    @Message           = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from proc_CallAssignProductClass.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
END
```

Appendix 5: Product Class Database (PCD) System Code

```
ELSE
BEGIN
    COMMIT TRAN
    RETURN
END
END -- End of proc_CallAssignProductClass Procedure.
GO
```

```
/*
1.12 Procedure Name: dbo.proc_CallAssignSpecification
```

Database: PCD

Description:

This procedure calls proc_AssignSpecification as a part of a transaction.

```
*/
```

```
CREATE Procedure proc_CallAssignSpecification
/* Param List */
@IDSpec                BIGINT,
@IDAssignToSpecTypeDef BIGINT,
@SpecValue             NVARCHAR(4000) = NULL,
@IDMeasUnit            BIGINT = NULL,
@IDProcState           TINYINT OUTPUT,
@Message              NVARCHAR(500) OUTPUT
AS
BEGIN
    /* This procedure calls proc_AssignSpecification as a part of a transaction. */
    BEGIN TRAN
    EXEC proc_AssignSpecification
        @IDSpec                = @IDSpec,
        @IDAssignToSpecTypeDef = @IDAssignToSpecTypeDef,
        @SpecValue             = @SpecValue,
        @IDMeasUnit            = @IDMeasUnit,
        @IDProcState           = @IDProcState    OUTPUT,
        @Message              = @Message       OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from proc_CallAssignSpecification.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        COMMIT TRAN
        RETURN
    END
END -- End of proc_CallAssignSpecification Procedure.
GO
```

```
/*
1.13 Procedure Name: dbo.proc_CallAssignSpecificationGroup
```

Database: PCD

Description:

This procedure calls proc_AssignSpecificationGroup as a part of a transaction.

```
*/
```

```
CREATE Procedure proc_CallAssignSpecificationGroup
/* Param List */
@IDSpecGroupDef        BIGINT,
@IDAssignToSpecTypeDef BIGINT,
@IDProcState           TINYINT OUTPUT,
@Message              NVARCHAR(500) OUTPUT
AS
BEGIN
    /* This procedure calls proc_AssignSpecificationGroup as a part of a transaction. */
    BEGIN TRAN
    EXEC proc_AssignSpecificationGroup
```

Appendix 5: Product Class Database (PCD) System Code

```
@IDSpecGroupDef      = @IDSpecGroupDef,
@IDAssignToSpecTypeDef = @IDAssignToSpecTypeDef,
@IDProcState         = @IDProcState  OUTPUT,
@Message             = @Message      OUTPUT
IF @IDProcState != 0
BEGIN
    ROLLBACK TRAN
    SELECT @Message = @Message + ' This Procedure was called from proc_CallAssignSpecificationGroup.'
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END
ELSE
BEGIN
    COMMIT TRAN
    RETURN
END
END -- End of proc_CallAssignSpecificationGroup Procedure.
GO
```

1.14 Procedure Name: dbo.proc_CallAssignTableSpecification

Database: PCD

Description:

This procedure calls proc_AssignTableSpecification as a part of a transaction.

```
CREATE Procedure proc_CallAssignTableSpecification
/* Param List */
@IDTableVerDef      BIGINT,
@IDAssignToSpecTypeDef = @IDAssignToSpecTypeDef,
@IDProcState         TINYINT OUTPUT,
@Message            NVARCHAR(500) OUTPUT
AS
BEGIN
    /* This procedure calls proc_AssignTableSpecification as a part of a transaction.*/
    BEGIN TRAN
    EXEC proc_AssignTableSpecification
    @IDTableVerDef      = @IDTableVerDef,
    @IDAssignToSpecTypeDef = @IDAssignToSpecTypeDef,
    @IDProcState         = @IDProcState  OUTPUT,
    @Message             = @Message      OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from proc_CallAssignTableSpecification.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        COMMIT TRAN
        RETURN
    END
END -- End of proc_CallAssignTableSpecification Procedure.
GO
```

1.15 Procedure Name: dbo.proc_Category_ProductClass_sel

Database: PCD

Description:

This procedure provides information on all the product classes that are assigned to a category. If IDCATEGORY is not provided then all the top level product classes assigned to the categories are selected.

```
CREATE PROCEDURE proc_Category_ProductClass_sel
@IDCategory      BIGINT,
@OrderBy         NVARCHAR(255) = NULL,
```

Appendix 5: Product Class Database (PCD) System Code

```
@UpDown          NVARCHAR(255) = 'ASC',
@RecordCount     INT OUTPUT
AS
DECLARE @SQL      NVARCHAR(4000)
SET NoCount ON

/*
IF @IDCategory = -1, all the top level product classes assigned to all the categories are selected.
*/
IF (@IDCategory = -1)
BEGIN

    SET @SQL = ' SELECT c_pc.IDCategory,
c_pc.IDProdClassDef,
pcv.IDProdClass,
pcv.IDProdClassVer,
--pcv.ProdClassVerDesc,
pc.ProdClassName
--pc.ProdClassDesc
FROM dbo.Category_ProductClass c_pc , dbo.ProductClassVersion pcv , dbo.ProductClass pc
WHERE c_pc.IDProdClassDef = pcv.IDProdClassDef AND
pcv.IDProdClass = pc.IDProdClass '

END
ELSE
BEGIN

    SET @SQL = ' SELECT c_pc.IDCategory,
c_pc.IDProdClassDef,
pcv.IDProdClass,
pcv.IDProdClassVer,
--pcv.ProdClassVerDesc,
pc.ProdClassName
--pc.ProdClassDesc
FROM dbo.Category_ProductClass c_pc , dbo.ProductClassVersion pcv , dbo.ProductClass pc
WHERE c_pc.IDProdClassDef = pcv.IDProdClassDef AND
pcv.IDProdClass = pc.IDProdClass  AND
c_pc.IDCategory = @IDCategory'

END
IF (@OrderBy IS NOT NULL)
BEGIN
    SET @SQL = @SQL + ' ORDER BY ' + @OrderBy + ' ' + @UpDown
END
EXEC sp_executesql @SQL, N'@IDCategory BIGINT', @IDCategory
SELECT @RecordCount = @@rowcount
GO
```

```
/*
```

1.16 Procedure Name: `dbo.proc_Category_sel`

Database: PCD

Description:

This procedure provides information on categories such as IDCategory, CategoryName and CategoryDescription.

```
*/
```

```
CREATE PROCEDURE proc_Category_sel
@IDCategory      BIGINT = NULL,
@OrderBy         NVARCHAR(255) = NULL,
@UpDown         NVARCHAR(255) = 'ASC',
@RecordCount     INT OUTPUT
AS
DECLARE @SQL      NVARCHAR(4000)
SET NoCount ON
SET @SQL = ' SELECT dbo.Category.IDCategory,
                dbo.Category.CategoryName,
                dbo.Category.CategoryDesc
FROM Category '
```

Appendix 5: Product Class Database (PCD) System Code

```
IF (@IDCategory IS NOT NULL)
BEGIN
    SET @SQL = @SQL + ' WHERE IDCategory = @IDCategory'
END
IF (@OrderBy IS NOT NULL)
BEGIN
    SET @SQL = @SQL + ' ORDER BY ' + @OrderBy + ' ' + @UpDown
END
EXEC sp_executesql @SQL, N'@IDCategory BIGINT',@IDCategory
SELECT @RecordCount = @@rowcount
GO
```

/* 1.17 Procedure Name: dbo.proc_Category_SubCategory_sel

Database: PCD

Description:

This procedure enables selection of categories and sub categories.

*/

```
CREATE PROCEDURE proc_Category_SubCategory_sel
@IDCategory          BIGINT = NULL,
@OrderBy             NVARCHAR(255) = NULL,
@UpDown             NVARCHAR(255) = 'ASC',
@RecordCount        INT OUTPUT
AS
DECLARE @SQL          NVARCHAR(4000)
SET NoCount ON
/*
The following if block select all the top categories.
*/
IF (@IDCategory IS NULL)
BEGIN
    SET @SQL = ' SELECT dbo.Category.IDCategory,
dbo.Category.CategoryName,
dbo.Category.CategoryDesc
FROM Category
WHERE dbo.Category.IDCategory NOT IN
(SELECT dbo.Category_SuperCategory.IDCategory FROM dbo.Category_SuperCategory) '

    IF (@OrderBy IS NOT NULL)
    BEGIN
        SET @SQL = @SQL + ' ORDER BY ' + @OrderBy + ' ' + @UpDown
    END
    EXEC sp_executesql @SQL, N''
    SELECT @RecordCount = @@rowcount
    RETURN
END

--The Following if block is executed to get IDCategory, CategoryName from category table and
-- IDSuperCategory from category_SuperCategory table. By calling the following if block only one call is
-- made to the stored procedure to get categories and their sub categories.
ELSE IF(@IDCategory = -1)
BEGIN
    SET @SQL = ' select c.IDCategory, c.CategoryName, c_sc.IDSuperCategory from category c
LEFT OUTER JOIN category_SuperCategory c_sc
on c.IDCategory = c_sc.IDCategory'

END
ELSE
BEGIN
    SET @SQL = ' SELECT dbo.Category.IDCategory,
dbo.Category.CategoryName,
dbo.Category.CategoryDesc
FROM Category
WHERE dbo.Category.IDCategory IN
(SELECT dbo.Category_SubCategory.IDSubCategory FROM Category_SubCategory where IDCategory =
@IDCategory)'
```

Appendix 5: Product Class Database (PCD) System Code

```
END
IF (@OrderBy IS NOT NULL)
BEGIN
    SET @SQL = @SQL + ' ORDER BY ' + @OrderBy + ' ' + @UpDown
END
EXEC sp_executesql @SQL, N'@IDCategory BIGINT',@IDCategory
SELECT @RecordCount = @@rowcount
GO
```

```
/*
```

1.18 Procedure Name: dbo.proc_CreateCategory

Database: PCD

Description:

This procedure enables creation of category.

```
*/
```

```
CREATE Procedure proc_CreateCategory
/* Param List */
@CategoryName      NVARCHAR(255),
@IDSuperCategory   BIGINT = NULL,
@CategoryDesc      NVARCHAR(4000) = NULL,
@IDProcState       TINYINT      OUTPUT,
@Message           NVARCHAR(500) OUTPUT
AS
BEGIN
    DECLARE
        @IDCategory      BIGINT,
        @Error           INT
    SELECT @IDProcState = 0
    SELECT @ERROR = 0
    BEGIN TRAN
    /*Get a new category ID */
    EXEC dbo.proc_GetNewID
    @IDEntity = 102,
    @IDNew = @IDCategory OUTPUT,
    @Message = @Message OUTPUT,
    @IDProcState = @IDProcState OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @IDProcState = 1
        SELECT @Message = @Message + ' This Procedure was called from Proc_CreateCategory.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END

    /*Create the Category by inserting values into the Category table */
    INSERT INTO Category(IDCategory, CategoryName, CategoryDesc)
    VALUES (@IDCategory, LTRIM(RTRIM(@CategoryName)), LTRIM(RTRIM(@CategoryDesc)))
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @IDProcState = 2
        SELECT @Message = 'An error occurred while creating the new category.'
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Error occurred in Procedure proc_CreateCategory. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END

    /* Now insert values into the CategoryHierarchy table to maintain category hierarchy. */
    /*IF @SuperCategory is null then a category is top level category and the IDSuperCategory
    value for the category is 0. It also has no sub category. */
    IF @IDSuperCategory IS NULL
    BEGIN
```

Appendix 5: Product Class Database (PCD) System Code

```

SELECT @Message = 'Category successfully Created. '
SELECT @Message = @Message + 'Category ID is: ' + CAST(@IDCategory AS NVARCHAR) + '
COMMIT TRAN
RETURN

END
ELSE
BEGIN

/* If a category has a super category then the category is also a sub category
of that super category. In this case two inserts are required. First to create a
category and its super category and second to create a supercategory and its sub category
For this we call proc_AssignCategory. */

EXEC proc_AssignCategory
@IDCategory      = @IDCategory,
@IDSuperCategory = @IDSuperCategory,
@IDProcState     = @IDProcState OUTPUT,
@Message         = @Message OUTPUT
IF @IDProcState != 0
BEGIN
    ROLLBACK TRAN
    SELECT @Message = @Message + ' This Procedure was called from Proc_AssignCategory.'
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END
ELSE
BEGIN
    SELECT @Message = 'Category successfully Created. '
    SELECT @Message = @Message + 'Category ID is: ' + CAST(@IDCategory AS NVARCHAR) + '
    COMMIT TRAN
    RETURN
END
END

END -- End of Proc_CreateCategory Procedure.
GO

```

1.19 Procedure Name: dbo.proc_CreateNewListSpecification

Database: PCD

Description:

This procedure enables creation of new list specification.

*/

```

CREATE Procedure proc_CreateNewListSpecification
/* Param List */
@ListName          NVARCHAR(255),
@IDListVer         MONEY = NULL,
@ListDesc          NVARCHAR(4000) = NULL,
@ListVerDesc       NVARCHAR(4000) = NULL,
@ListValues        NVARCHAR(4000),
@ListIDMeasUnits   NVARCHAR(4000),
@IDAssignToSpecTypeDef BIGINT = NULL,
@IDListDef         BIGINT OUTPUT,
@IDProcState       TINYINT OUTPUT,
@Message           NVARCHAR(500) OUTPUT
AS
BEGIN
    DECLARE
        @IDList          BIGINT,
        --@IDListDef     BIGINT,
        @Error          INT
    SELECT @IDProcState = 0
    SELECT @ERROR = 0
    BEGIN TRAN
    /*Get an new List ID */
    EXEC dbo.proc_GetNewID
    @IDEntity = 107,
    @IDNew    = @IDList OUTPUT,

```

Appendix 5: Product Class Database (PCD) System Code

```
@Message = @Message OUTPUT,
@IDProcState = @IDProcState OUTPUT
IF @IDProcState != 0
BEGIN
    ROLLBACK TRAN
    SELECT @Message = @Message + ' This Procedure was called from proc_CreateNewListSpecification.'
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END

/* Select version ID as 1 if it is not provided */
IF @IDListVer IS NULL
BEGIN
    SELECT @IDListVer = 1
END
/*Get a new List definition ID */
EXEC dbo.proc_GetNewID
@IDEntity = 108,
@IDNew = @IDListDef OUTPUT,
@Message = @Message OUTPUT,
@IDProcState = @IDProcState OUTPUT
IF @IDProcState != 0
BEGIN
    ROLLBACK TRAN
    SELECT @Message = @Message + ' This Procedure was called from proc_CreateNewListSpecification.'
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END

/*Create a List specification by inserting values into ListSpecification, List version
tables. The order in which the values are inserted into the
table should be maintained. First values should be inserted into the ListSpecification table
then into ListVersion table because IDList in ListVersion table
references IDList in ListSpecification.*/

INSERT INTO ListSpecification(IDList, ListName,ListDesc)
VALUES (@IDList, LTRIM(RTRIM(@ListName)), LTRIM(RTRIM(@ListDesc)))

SELECT @Error = @@ERROR
IF @Error != 0
BEGIN
    ROLLBACK TRAN
    SELECT @IDProcState = 1
    SELECT @Message = 'An error occurred while creating the new list specification.'
    SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
    SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + ' '
    SELECT @Message = @Message + 'Error occurred in Procedure proc_CreateNewListSpecification.'
    SELECT @Message = @Message + 'Procedure is terminated abnormally.'
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END

INSERT INTO ListVersion(IDList, IDListVer, IDListDef, ListVerDesc)
VALUES(@IDList,@IDListVer,@IDListDef, LTRIM(RTRIM(@ListVerDesc)))
SELECT @Error = @@ERROR
IF @Error != 0
BEGIN
    ROLLBACK TRAN
    SELECT @IDProcState = 2
    SELECT @Message = 'An error occurred while creating the new list specification.'
    SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
    SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + ' '
    SELECT @Message = @Message + 'Error occurred in Procedure proc_CreateNewListSpecification.'
    SELECT @Message = @Message + 'Procedure is terminated abnormally.'
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END
```

Appendix 5: Product Class Database (PCD) System Code

```
/* Now assign the list specification. The following procedure is called for assigning the list specification. A list specification
can be assigned to a product class or specification group. */
IF @IDAssignToSpecTypeDef IS NOT NULL
BEGIN
    EXEC proc_AssignListSpecification
        @IDListDef = @IDListDef,
        @IDAssignToSpecTypeDef = @IDAssignToSpecTypeDef,
        @IDProcState = @IDProcState OUTPUT,
        @Message = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from
        Proc_CreateNewListSpecification.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
END

/* Now insert values in the ListDefiniton Table. For this we call proc_InsertListValues
procedure */
EXEC dbo.proc_InsertListValues
    @IDListDef = @IDListDef,
    @ListValues = @ListValues,
    @ListIDMeasUnits = @ListIDMeasUnits,
    @Message = @Message OUTPUT,
    @IDProcState = @IDProcState OUTPUT
IF @IDProcState != 0
BEGIN
    ROLLBACK TRAN
    SELECT @Message = @Message + ' This Procedure was called from proc_CreateNewListSpecification.'
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END
ELSE
BEGIN
    /*If everything goes well until this point it means that all the values are inserted
    properly into the above tables. Now we need to commit the transaction. */
    COMMIT TRAN
    SELECT @Message = 'List Specification Created Successfully. List Specification ID is: ' + CAST(@IDList AS
    NVARCHAR) + ' '
    SELECT @Message = @Message + 'List Specification Version is: ' + CAST(@IDListVer AS NVARCHAR) + ' '
    RETURN
END
END -- End of proc_CreateNewListSpecification Procedure.
GO
```

1.20 Procedure Name: dbo.proc_CreateNewListSpecificationVersion

Database: PCD

Description:

This procedure enables creation of new list specification version.

*/

```
CREATE Procedure proc_CreateNewListSpecificationVersion
/* Param List */
@IDList BIGINT,
@IDListVer MONEY = NULL,
@ListVerDesc NVARCHAR(4000) = NULL,
@ListValues NVARCHAR(4000),
@ListIDMeasUnits NVARCHAR(4000) = NULL,
@IDAssignToSpecTypeDef BIGINT = NULL,
@IDListDef BIGINT OUTPUT,
@IDProcState TINYINT OUTPUT,
@Message NVARCHAR(500) OUTPUT
AS
BEGIN
    DECLARE
```

Appendix 5: Product Class Database (PCD) System Code

```
@IDExistingLatestVer MONEY,
--@IDListDef          BIGINT,
@Error               INT
SELECT @IDProcState = 0
SELECT @ERROR = 0
BEGIN TRAN

/*Check whether the supplied List specification exists. If yes, get the
existing latest version of that List specification in the database */

SELECT @IDExistingLatestVer = MAX(IDListVer) FROM ListVersion
WHERE IDList = @IDList
IF @IDExistingLatestVer IS NULL
BEGIN
    ROLLBACK TRAN
    SELECT @IDProcState = 1
    SELECT @Message = 'List Specification ID supplied does not exists. '
    SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + ' '
    SELECT @Message = @Message + 'Error occurred in Procedure proc_CreateNewListSpecificationVersion. '
    SELECT @Message = @Message + 'Procedure is terminated abnormally. '
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END

/*IDListVer supplied should be always greate than existing latest version in the database */

IF @IDListVer IS NULL
BEGIN
    SELECT @IDListVer = FLOOR(@IDExistingLatestVer) + 1
END
ELSE
BEGIN
    IF @IDListVer < @IDExistingLatestVer
    BEGIN
        ROLLBACK TRAN
        SELECT @IDProcState = 2
        SELECT @Message = 'Supplied IDListVersion ' + CAST(@IDListVer AS NVARCHAR) + ' is less
than existing latest version in the database. '
        SELECT @Message = @Message + 'Existing lateste version is: ' + CAST(@IDExistingLatestVer AS
NVARCHAR) + ' '
        SELECT @Message = @Message + 'Please provide a version ID greater than the existing latest version
or leave it blank for the system to generate it for you. '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
NVARCHAR) + ' '
        SELECT @Message = @Message + 'Error occurred in Procedure
proc_CreateNewListSpecificationVersion. '
        SELECT @Message = @Message + 'Procedure is abnormally terminated.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
END

/*Create new IDListDef for the new version of the product class.*/

EXEC dbo.proc_GetNewID
@IDEntity = 108,
@IDNew = @IDListDef OUTPUT,
@Message = @Message OUTPUT,
@IDProcState = @IDProcState OUTPUT

IF @IDProcState != 0
BEGIN
    ROLLBACK TRAN
    SELECT @Message = @Message + ' This Procedure was called from proc_CreateNewListSpecificationVersion.'
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END

/*Create a list specification version by inserting values into the ListVersion table.*/
```

Appendix 5: Product Class Database (PCD) System Code

```
INSERT INTO ListVersion(IDList, IDListVer, IDListDef, ListVerDesc)
VALUES (@IDList, @IDListVer, @IDListDef, LTRIM(RTRIM(@ListVerDesc)))
SELECT @Error = @@ERROR
IF @Error != 0
BEGIN
    ROLLBACK TRAN
    SELECT @IDProcState = 3
    SELECT @Message = 'An error occurred while creating the new List Specification version.'
    SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
    SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + ' '
    SELECT @Message = @Message + 'Error occurred in Procedure proc_CreateNewListSpecificationVersion.'
    SELECT @Message = @Message + 'Procedure is terminated abnormally.'
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END

/* Now assign the list specification. The following procedure is called for
   assigning the list specification. A list specification can be assigned
   to a product class or specification group. */

IF @IDAssignToSpecTypeDef IS NOT NULL
BEGIN
    EXEC proc_AssignListSpecification
    @IDListDef = @IDListDef,
    @IDAssignToSpecTypeDef = @IDAssignToSpecTypeDef,
    @IDProcState = @IDProcState OUTPUT,
    @Message = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from
        Proc_CreateNewListSpecification.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
END

/* Now insert values in the ListDefiniton Table. For this we call proc_InsertListValues
   procedure */

EXEC dbo.proc_InsertListValues
@IDListDef = @IDListDef,
@ListValues = @ListValues,
@ListIDMeasUnits = @ListIDMeasUnits,
@Message = @Message OUTPUT,
@IDProcState = @IDProcState OUTPUT

IF @IDProcState != 0
BEGIN
    ROLLBACK TRAN
    SELECT @Message = @Message + ' This Procedure was called from proc_CreateNewListSpecificationVersion.'
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END
ELSE
BEGIN
    /*If everything goes well until this point it means that all the values are inserted
    properly into the above tables. Now we need to commit the transaction. */

    COMMIT TRAN
    SELECT @Message = 'List Specification Version Created Successfully. List Specification ID is: ' +
    CAST(@IDList AS NVARCHAR) + ' '
    SELECT @Message = @Message + 'List Specification Version is: ' + CAST(@IDListVer AS NVARCHAR) + ' '
    RETURN
END
END -- End of proc_CreateNewListSpecificationVersion Procedure.
GO
```

Appendix 5: Product Class Database (PCD) System Code

1.21 Procedure Name: dbo.proc_CreateNewProductClass

Database: PCD

Description:

This procedure enables the creation of new product class.

*/

```
CREATE Procedure proc_CreateNewProductClass
```

```
/* Param List */
```

```
@ProdClassName          NVARCHAR(255),
@IDProdClassVer          MONEY = NULL,
@ProdClassDesc           NVARCHAR(4000) = NULL,
@ProdClassVerDesc        NVARCHAR(4000) = NULL,
@IDAssignToSpecTypeDef   BIGINT = NULL,
@IDProdClassDef           BIGINT OUTPUT,
@IDProcState             TINYINT OUTPUT,
@Message                 NVARCHAR(500) OUTPUT
```

```
AS
BEGIN
```

```
DECLARE
```

```
@IDProdClass            BIGINT,
--@IDProdClassDef        BIGINT,
@Error                  INT
SELECT @IDProcState     = 0
SELECT @ERROR           = 0
```

```
BEGIN TRAN
```

```
/*Get an new ProductClass ID */
```

```
EXEC dbo.proc_GetNewID
```

```
@IDEntity = 103,
@IDNew     = @IDProdClass OUTPUT,
@Message   = @Message OUTPUT,
@IDProcState = @IDProcState OUTPUT
IF @IDProcState != 0
BEGIN
```

```
ROLLBACK TRAN
```

```
SELECT @Message = @Message + ' This Procedure was called from Proc_CreateNewProductClass.'
RAISERROR(@Message,1,1) WITH SETERROR
RETURN @@ERROR
```

```
END
```

```
/* Select version ID as 1 if it is not provided */
```

```
IF @IDProdClassVer IS NULL
```

```
BEGIN
    SELECT @IDProdClassVer = 1
END
```

```
/*Get a new product class definition ID */
```

```
EXEC dbo.proc_GetNewID
```

```
@IDEntity = 105,
@IDNew     = @IDProdClassDef OUTPUT,
@Message   = @Message OUTPUT,
@IDProcState = @IDProcState OUTPUT
```

```
IF @IDProcState != 0
```

```
BEGIN
    ROLLBACK TRAN
    SELECT @Message = @Message + ' This Procedure was called from Proc_CreateNewProductClass.'
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END
```

```
/*Create a product class by inserting values into ProductClass, product class version
and Category_ProductClass tables. The order in which the values are inserted into the
table should be maintained. First values should be inserted into the ProductClass table
then into ProductClassVersion table because IDProdClass in ProductClassVersion table
```

Appendix 5: Product Class Database (PCD) System Code

references IDProdClass in ProductClass. Finally the product class should be assigned.
A product class can be assigned to a product class, category or specification group.
For assigning the product class we call proc_AssignProductclass procedure.*/

```
INSERT INTO ProductClass(IDProdClass, ProdClassName,ProdClassDesc)
VALUES (@IDProdClass, LTRIM(RTRIM(@ProdClassName)), LTRIM(RTRIM(@ProdClassDesc)) )

SELECT @Error = @@ERROR
IF @Error != 0
BEGIN
    ROLLBACK TRAN
    SELECT @IDProcState = 1
    SELECT @Message = 'An error occured while creating the new product class.'
    SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
    SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + ' '
    SELECT @Message = @Message + 'Error occurred in Procedure proc_CreateNewProductClass.'
    SELECT @Message = @Message + 'Procedure is terminated abnormally.'
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END

INSERT INTO ProductClassVersion(IDProdClass, IDProdClassVer, IDProdClassDef, ProdClassVerDesc)
VALUES(@IDProdClass,@IDProdClassVer,@IDProdClassDef, LTRIM(RTRIM(@ProdClassVerDesc)) )
SELECT @Error = @@ERROR
IF @Error != 0
BEGIN
    ROLLBACK TRAN
    SELECT @IDProcState = 2
    SELECT @Message = 'An error occured while creating the new product class.'
    SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
    SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + ' '
    SELECT @Message = @Message + 'Error occurred in Procedure proc_CreateNewProductClass.'
    SELECT @Message = @Message + 'Procedure is terminated abnormally.'
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END

/* Assigning product class */
EXEC proc_AssignProductClass
@IDProdClassDef = @IDProdClassDef,
@IDAssignToSpecTypeDef = @IDAssignToSpecTypeDef,

@IDProcState = @IDProcState OUTPUT,
@Message = @Message OUTPUT
IF @IDProcState != 0
BEGIN
    ROLLBACK TRAN
    SELECT @Message = @Message + ' This Procedure was called from proc_CreateNewProductClass.'
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END
ELSE
BEGIN
    /*If everything goes well until this point it means that all the values are inserted
    properly into the above tables. Now we need to commit the transaction. */

    COMMIT TRAN
    SELECT @Message = 'Product Class Created Successfully. Product Class ID is: ' + CAST(@IDProdClass AS
    NVARCHAR) + ' '
    SELECT @Message = @Message + 'Product Class Version is: ' + CAST(@IDProdClassVer AS NVARCHAR) +
    ' '
    RETURN
END
END -- End of Proc_CreateNewProductClass Procedure.
GO
```

Appendix 5: Product Class Database (PCD) System Code

1.22 Procedure Name: dbo.proc_CreateNewProductClassVersion

Database: PCD

Description:

This procedure enables the creation of new product class version.

*/

```
CREATE Procedure proc_CreateNewProductClassVersion
/* Param List */
@IDProdClass          BIGINT,
@IDProdClassVer       MONEY = NULL,
@ProdClassVerDesc     NVARCHAR(4000) = NULL,
@IDAssignToSpecTypeDef BIGINT = NULL,
@IDProdClassDef       BIGINT OUTPUT,
@IDProcState          TINYINT OUTPUT,
@Message              NVARCHAR(500) OUTPUT
AS
BEGIN

    DECLARE
    @IDExistingLatestVer MONEY,
    --@IDProdClassDef    BIGINT,
    @Error              INT
    SELECT @IDProcState = 0
    SELECT @ERROR = 0
    BEGIN TRAN

    /*Check whether the supplied product class exists. If yes, get the
    existing latest version of that product class in the database */
    SELECT @IDExistingLatestVer = MAX(IDProdClassVer) FROM ProductClassVersion
    WHERE IDProdClass = @IDProdClass

    IF @IDExistingLatestVer IS NULL
    BEGIN
        ROLLBACK TRAN
        SELECT @IDProcState = 1
        SELECT @Message = 'Product class ID supplied does not exists. '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + '. '
        SELECT @Message = @Message + 'Error occurred in Procedure proc_CreateNewProductClassVersion. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END

    /*IDProdClassVer supplied should be always greate than existing latest version in the database */

    IF @IDProdClassVer IS NULL
    BEGIN
        SELECT @IDProdClassVer = FLOOR(@IDExistingLatestVer) + 1
    END
    ELSE
    BEGIN
        IF @IDProdClassVer < @IDExistingLatestVer
        BEGIN
            ROLLBACK TRAN
            SELECT @IDProcState = 2
            SELECT @Message = 'Supplied IDProductClassVersion ' + CAST(@IDProdClassVer AS
            NVARCHAR) + ' is less than existing latest version in the database. '
            SELECT @Message = @Message + 'Existing lateste version is: ' + CAST(@IDExistingLatestVer AS
            NVARCHAR) + '. '
            SELECT @Message = @Message + 'Please provide a version ID greater than the existing latest version
            or leave it blank for the system to generate it for you. '
            SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
            NVARCHAR) + '. '
            SELECT @Message = @Message + 'Error occured in Procedure proc_CreateNewProductClassVersion.
            '
            SELECT @Message = @Message + 'Procedure is abnormally terminated.'
            RAISERROR(@Message,1,1) WITH SETERROR
        END
    END
END
```

Appendix 5: Product Class Database (PCD) System Code

```
        RETURN @@ERROR
    END
END

/*Create new IDProdClassDef for the new version of the product class.*/

EXEC dbo.proc_GetNewID
@IDEntity = 105,
@IDNew = @IDProdClassDef OUTPUT,
@Message = @Message OUTPUT,
@IDProcState = @IDProcState OUTPUT

IF @IDProcState != 0
BEGIN
    ROLLBACK TRAN
    SELECT @Message = @Message + ' This Procedure was called from Proc_CreateNewProductClassVersion.'
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END

/*Create a product class by inserting values into ProductClassVersion and Category_ProductClass tables*/

INSERT INTO ProductClassVersion(IDProdClass, IDProdClassVer, IDProdClassDef, ProdClassVerDesc)
VALUES
(@IDProdClass, @IDProdClassVer, @IDProdClassDef, LTRIM(RTRIM(@ProdClassVerDesc)))
SELECT @Error = @@ERROR
IF @Error != 0
BEGIN
    ROLLBACK TRAN
    SELECT @IDProcState = 3
    SELECT @Message = 'An error occured while creating the new product class version.'
    SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
    SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + ' '
    SELECT @Message = @Message + 'Error occured in Procedure proc_CreateNewProductClassVersion. '
    SELECT @Message = @Message + 'Procedure is terminated abnormally. '
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END

/* Assigning product class */

EXEC proc_AssignProductClass
@IDProdClassDef = @IDProdClassDef,
@IDAssignToSpecTypeDef = @IDAssignToSpecTypeDef,
@IDProcState = @IDProcState OUTPUT,
@Message = @Message OUTPUT
IF @IDProcState != 0
BEGIN
    ROLLBACK TRAN
    SELECT @Message = @Message + ' This Procedure was called from proc_CreateNewProductClass.'
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END
ELSE
BEGIN
    /*If everything goes well until this point it means that all the values are inserted
    properly into the above tables. Now we need to commit the transaction.*/

    COMMIT TRAN
    SELECT @Message = 'Product Class Version created successfully. Product Class ID is: ' + CAST(@IDProdClass
    AS NVARCHAR) + ' '
    SELECT @Message = @Message + 'Product Class Version is: ' + CAST(@IDProdClassVer AS NVARCHAR) +
    ' ' RETURN
END
END -- End of Proc_CreateNewProductClassVersion Procedure.
GO
```

Appendix 5: Product Class Database (PCD) System Code

/* 1.23 Procedure Name: dbo.proc_CreateNewSpecification

Database: PCD

Description:

This procedure enables creation of new specification.

*/

```
CREATE Procedure proc_CreateNewSpecification
/* Param List */
@SpecName          NVARCHAR(255),
@SpecDesc          NVARCHAR(4000) = NULL,
@SpecValue         NVARCHAR(4000) = NULL,
@IDMeasUnit        BIGINT = NULL,
@IDAssignToSpecTypeDef  BIGINT = NULL,
@IDProcState       TINYINT OUTPUT,
@Message           NVARCHAR(500) OUTPUT
AS
BEGIN
    DECLARE
        @Error          INT,
        @IDSpec         BIGINT,
        @IDEntityPart   BIGINT
    SELECT @IDProcState = 0
    SELECT @ERROR = 0
    BEGIN TRAN

    /*Get an new specification ID */

    EXEC dbo.proc_GetNewID
    @IDEntity = 100,
    @IDNew = @IDSpec OUTPUT,
    @Message = @Message OUTPUT,
    @IDProcState = @IDProcState OUTPUT

    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from Proc_CreateNewSpecification.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END

    /*Create the Specification by inserting values into the Specification table */

    INSERT INTO Specification(IDSpec, SpecName, SpecDesc)
    VALUES (@IDSpec, LTRIM(RTRIM(@SpecName)), LTRIM(RTRIM(@SpecDesc)))

    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @IDProcState = 1
        SELECT @Message = 'An error occured while creating the new specification.'
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Error occured in Procedure proc_CreateNewSpecification.'
        SELECT @Message = @Message + 'Procedure is terminated abnormally.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END

    /* Now assign the specification to Specification Definition under which it was created if
    @AssignToSpecTypeDef is not null. Otherwise commit the transaction and return as
    the user has not specified which specification type def the new specification should
    be assigned to. */

    IF @IDAssignToSpecTypeDef IS NULL
    BEGIN
```

Appendix 5: Product Class Database (PCD) System Code

```

        COMMIT TRAN
        SELECT @Message = 'Specification Created Successfully. Specification ID is: ' +
        CAST(@IDSpec AS NVARCHAR) + '. '
    RETURN
END
ELSE
BEGIN
    EXEC proc_AssignSpecification
        @IDSpec                = @IDSpec,
        @IDAssignToSpecTypeDef = @IDAssignToSpecTypeDef,
        @SpecValue             = @SpecValue,
        @IDMeasUnit            = @IDMeasUnit,

        @IDProcState           = @IDProcState    OUTPUT,
        @Message                = @Message       OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from proc_CallAssignSpecification.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
END
ELSE
BEGIN
    COMMIT TRAN
    SELECT @Message = 'Specification Created Successfully. Specification ID is: ' +
    CAST(@IDSpec AS NVARCHAR) + '. '
    RETURN
END
END -- End of IF @IDEntityPart = 106
END -- End of Proc_CreateNewSpecification Procedure.
GO

```

/*
1.24 Procedure Name: dbo.proc_CreateNewSpecificationGroup

Database: PCD

Description:

This procedure enables creation of new specification group.

*/

```

CREATE Procedure proc_CreateNewSpecificationGroup
/* Param List */
@SpecGroupName          NVARCHAR(255),
@IDSpecGroupVer         MONEY = NULL,
@SpecGroupDesc          NVARCHAR(4000) = NULL,
@SpecGroupVerDesc       NVARCHAR(4000) = NULL,
@IDAssignToSpecTypeDef  BIGINT = NULL,
@IDSpecGroupDef         BIGINT OUTPUT,
@IDProcState            TINYINT OUTPUT,
@Message                NVARCHAR(500) OUTPUT

```

```

AS
BEGIN
    BEGIN TRAN
    DECLARE
        @IDSpecGroup          BIGINT,
        --@IDSpecGroupDef     BIGINT,
        @Error                INT,
        @IDEntityPart         INT
    SELECT @IDProcState      = 0
    SELECT @ERROR           = 0

    /*create a new SpecificationGroup ID */

    EXEC dbo.proc_GetNewID
        @IDEntity = 101,
        @IDNew    = @IDSpecGroup OUTPUT,
        @Message  = @Message OUTPUT,

```

Appendix 5: Product Class Database (PCD) System Code

```
@IDProcState = @IDProcState OUTPUT
IF @IDProcState != 0
BEGIN
    ROLLBACK TRAN
    SELECT @Message = @Message + ' This Procedure was called from Proc_CreateNewSpecificationGroup.'
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END

/*Select version ID as 1 if it is not provided */

IF @IDSpecGroupVer IS NULL
BEGIN
    SELECT @IDSpecGroupVer = 1
END

/*Create new specification group definition ID */
EXEC dbo.proc_GetNewID
@IDEntity = 106,
@IDNew = @IDSpecGroupDef OUTPUT,
@Message = @Message OUTPUT,
@IDProcState = @IDProcState OUTPUT

IF @IDProcState != 0
BEGIN
    ROLLBACK TRAN
    SELECT @Message = @Message + ' This Procedure was called from Proc_CreateNewSpecificationGroup.'
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END

/*Create a specification group by inserting values into SpecificationGroup and
SpecificationGroupVersion tables. The order in which the values are inserted into the
table should be maintained. First values should be inserted into the SpecificationGroup table
then into SpecificationGroupVersion table because IDSpecGroup in SpecificationGroupVersion table
references IDSpecGroup in SpecificationGroup table.*/

INSERT INTO SpecificationGroup (IDSpecGroup, SpecGroupName, SpecGroupDesc)
VALUES (@IDSpecGroup, LTRIM(RTRIM(@SpecGroupName)), LTRIM(RTRIM(@SpecGroupDesc)))
SELECT @Error = @@ERROR
IF @Error != 0
BEGIN
    ROLLBACK TRAN
    SELECT @IDProcState = 1
    SELECT @Message = 'An error occurred while creating the new specification group.'
    SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
    SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + ' '
    SELECT @Message = @Message + 'Error occurred in Procedure proc_CreateNewSpecificationGroup.'
    SELECT @Message = @Message + 'Procedure is terminated abnormally.'
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END

INSERT INTO SpecificationGroupVersion (IDSpecGroup, IDSpecGroupVer, IDSpecGroupDef, SpecGroupVerDesc)
VALUES (@IDSpecGroup, @IDSpecGroupVer, @IDSpecGroupDef, LTRIM(RTRIM(@SpecGroupVerDesc)))
SELECT @Error = @@ERROR
IF @Error != 0
BEGIN
    ROLLBACK TRAN
    SELECT @IDProcState = 1
    SELECT @Message = 'An error occurred while creating the new specification group.'
    SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
    SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + ' '
    SELECT @Message = @Message + 'Error occurred in Procedure proc_CreateNewSpecificationGroup.'
    SELECT @Message = @Message + 'Procedure is terminated abnormally.'
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END
```

Appendix 5: Product Class Database (PCD) System Code

```
/* Now assign the specification group. The following procedure is called for
   assigning the specification group */
IF @IDAssignToSpecTypeDef IS NOT NULL
BEGIN
    EXEC proc_AssignSpecificationGroup
        @IDSpecGroupDef = @IDSpecGroupDef,
        @IDAssignToSpecTypeDef = @IDAssignToSpecTypeDef,
        @IDProcState = @IDProcState OUTPUT,
        @Message = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from
        Proc_CreateNewSpecificationGroup.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
END

COMMIT TRAN
SELECT @Message = 'Specification group created Successfully. Specification Group ID is: ' + CAST(@IDSpecGroup AS
NVARCHAR) + '.'
SELECT @Message = @Message + 'Specification Group Version is: ' + CAST(@IDSpecGroupVer AS NVARCHAR) + '.'
RETURN
END -- End of Proc_CreateNewSpecificationGroup Procedure.
GO
```

```
/*
```

1.25 Procedure Name: dbo.proc_CreateNewSpecificationGroupVersion

Database: PCD

Description:

This procedure enables creation of new specification group version.

```
*/
```

```
CREATE Procedure proc_CreateNewSpecificationGroupVersion
/* Param List */
@IDSpecGroup          BIGINT,
@IDSpecGroupVer       MONEY = NULL,
@SpecGroupVerDesc     NVARCHAR(4000) = NULL,
@IDAssignToSpecTypeDef BIGINT = NULL,
@IDSpecGroupDef       BIGINT OUTPUT,
@IDProcState          TINYINT OUTPUT,
@Message              NVARCHAR(500) OUTPUT

AS
BEGIN

    DECLARE
    @IDExistingLatestVer MONEY,
    --@IDSpecGroupDef    BIGINT,
    @Error              INT
    SELECT @IDProcState = 0
    SELECT @ERROR = 0

    BEGIN TRAN

    /*Check whether the supplied specification group exists. If yes, get the
    existing latest version of that specification group in the database */

    SELECT @IDExistingLatestVer = MAX(IDSpecGroupVer) FROM SpecificationGroupVersion
    WHERE IDSpecGroup = @IDSpecGroup

    IF @IDExistingLatestVer IS NULL
    BEGIN
        ROLLBACK TRAN
        SELECT @IDProcState = 1
        SELECT @Message = 'Specification Group ID supplied does not exists.'
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + '.'
    END
```

Appendix 5: Product Class Database (PCD) System Code

```
SELECT @Message = @Message + 'Error occured in Procedure proc_CreateNewSpecificationGroupVersion.'
SELECT @Message = @Message + 'Procedure is terminated abnormally.'
RAISERROR(@Message,1,1) WITH SETERROR
RETURN @@ERROR
END

/*IDSpecGroupVer supplied should always be greate than existing latest version in the database */

IF @IDSpecGroupVer IS NULL
BEGIN
    SELECT @IDSpecGroupVer = FLOOR(@IDExistingLatestVer) + 1
END
ELSE
BEGIN
    IF @IDSpecGroupVer < @IDExistingLatestVer
    BEGIN
        ROLLBACK TRAN
        SELECT @IDProcState = 2
        SELECT @Message = 'Supplied IDSpecificationGroupVersion ' + CAST(@IDSpecGroupVer AS
        NVARCHAR) + ' is less than existing latest version in the database.'
        SELECT @Message = @Message + 'Existing lateste version is: ' + CAST(@IDExistingLatestVer AS
        NVARCHAR) + '.'
        SELECT @Message = @Message + 'Please provide a version ID greater than the existing latest version
        or leave it blank for the system to generate it for you.'
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
        NVARCHAR) + '.'
        SELECT @Message = @Message + 'Error occured in Procedure
        proc_CreateNewSpecificationGroupVersion.'
        SELECT @Message = @Message + 'Procedure is abnormally terminated.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
END

/*Create new IDPSpecGroupDef for the new version of the specification group.*/
EXEC dbo.proc_GetNewID
@IDEntity = 106,
@IDNew = @IDSpecGroupDef OUTPUT,
@Message = @Message OUTPUT,
@IDProcState = @IDProcState OUTPUT

IF @IDProcState != 0
BEGIN
    ROLLBACK TRAN
    SELECT @Message = @Message + ' This Procedure was called from
    Proc_CreateNewSpecificationGroupVersion.'
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END

/*Create a specification group by inserting values into the SpecificationGroupVersion table*/
INSERT INTO SpecificationGroupVersion(IDSpecGroup, IDSpecGroupVer, IDSpecGroupDef, SpecGroupVerDesc)
VALUES (@IDSpecGroup, @IDSpecGroupVer, @IDSpecGroupDef, LTRIM(RTRIM(@SpecGroupVerDesc)) )
SELECT @Error = @@ERROR
IF @Error != 0
BEGIN
    ROLLBACK TRAN
    SELECT @IDProcState = 3
    SELECT @Message = 'An error occured while creating the new specification group version.'
    SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '.'
    SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + '.'
    SELECT @Message = @Message + 'Error occured in Procedure proc_CreateNewSpecificationGroupVersion.'
    SELECT @Message = @Message + 'Procedure is terminated abnormally.'
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END

COMMIT TRAN
```


Appendix 5: Product Class Database (PCD) System Code

```
SELECT @Message = 'Specification group Version created successfully. Specification group ID is: ' +  
CAST(@IDSpecGroup AS NVARCHAR) + '.'  
SELECT @Message = @Message + 'Specification group Version is: ' + CAST(@IDSpecGroupVer AS NVARCHAR) + '.'
```

```
END -- End of Proc_CreateNewSpecificationGroupVersion Procedure.  
GO
```

1.26 Procedure Name: **dbo.proc_CreateNewTableSpecification**

Database: PCD

Description:

This procedure enables creation of table specification group.

*/

```
CREATE Procedure proc_CreateNewTableSpecification
```

```
/* Param List */
```

```
@TableSpecName          NVARCHAR(255),  
@IDTableSpecVer         MONEY          = NULL,  
@NumOfRows              INT            = NULL,  
@NumOfColumns           INT,  
@ColumnValues           NVARCHAR(4000),  
@ColIDMeasUnits         NVARCHAR(4000) = NULL,  
@RowContent             NVARCHAR(100),  
@TableSpecDesc          NVARCHAR(4000) = NULL,  
@TableSpecVerDesc       NVARCHAR(4000) = NULL,  
@IDAssignToSpecTypeDef  BIGINT         = NULL,  
@IDTableVerDef          BIGINT         OUTPUT,  
@IDProcState            TINYINT        OUTPUT,  
@Message                NVARCHAR(500)  OUTPUT
```

```
AS  
BEGIN
```

```
DECLARE  
@IDTableSpec            BIGINT,  
--@IDTableVerDef       BIGINT,  
@Error                 INT  
SELECT @IDProcState = 0  
SELECT @ERROR = 0
```

```
BEGIN TRAN  
/*Get an new Table Specification ID */  
EXEC dbo.proc_GetNewID  
@IDEntity = 109,  
@IDNew    = @IDTableSpec OUTPUT,  
@Message  = @Message OUTPUT,  
@IDProcState = @IDProcState OUTPUT
```

```
IF @IDProcState != 0  
BEGIN  
    ROLLBACK TRAN  
    SELECT @Message = @Message + ' This Procedure was called from proc_CreateNewTableSpecification.'  
    RAISERROR(@Message,1,1) WITH SETERROR  
    RETURN @@ERROR  
END
```

```
/* Select version ID as 1 if it is not provided */
```

```
IF @IDTableSpecVer IS NULL  
BEGIN  
    SELECT @IDTableSpecVer = 1  
END
```

```
/*Get a new table version definition ID */
```

```
EXEC dbo.proc_GetNewID  
@IDEntity = 110,  
@IDNew    = @IDTableVerDef  OUTPUT,
```

Appendix 5: Product Class Database (PCD) System Code

```
@Message = @Message      OUTPUT,
@IDProcState = @IDProcState  OUTPUT

IF @IDProcState != 0
BEGIN
    ROLLBACK TRAN
    SELECT @Message = @Message + ' This Procedure was called from proc_CreateNewTableSpecification.'
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END

/*Create the table specification by inserting values into TableSpecification and TableVersion
tables. The order in which the values are inserted into the table should be maintained. First values s
ould be inserted into the
TableSpecification table
then into the TableVersion table because IDTableSpec in TableVersion table
references IDTableSpec in TableSpecification. */

INSERT INTO TableSpecification(IDTableSpec, TableSpecName, TableSpecDesc)
VALUES (@IDTableSpec, LTRIM(RTRIM(@TableSpecName)), LTRIM(RTRIM(@TableSpecDesc)))

SELECT @Error = @@ERROR
IF @Error != 0
BEGIN
    ROLLBACK TRAN
    SELECT @IDProcState = 1
    SELECT @Message = 'An error occurred while creating the new table specification.'
    SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
    SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + ' '
    SELECT @Message = @Message + 'Error occurred in Procedure proc_CreateNewTableSpecification.'
    SELECT @Message = @Message + 'Procedure is terminated abnormally.'
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END

INSERT INTO TableVersion(IDTableSpec, IDTableSpecVer, IDTableVerDef,
NumOfRows, NumOfColumns, TableVerDesc)
VALUES(@IDTableSpec,@IDTableSpecVer,@IDTableVerDef, @NumOfRows, @NumOfColumns,
LTRIM(RTRIM(@TableSpecVerDesc)))
SELECT @Error = @@ERROR
IF @Error != 0
BEGIN
    ROLLBACK TRAN
    SELECT @IDProcState = 2
    SELECT @Message = 'An error occurred while creating the new table specification.'
    SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
    SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + ' '
    SELECT @Message = @Message + 'Error occurred in Procedure proc_CreateNewTableSpecification.'
    SELECT @Message = @Message + 'Procedure is terminated abnormally.'
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END

/* Assign the table specification by calling the following procedure */
/* A Table specification can be assigned to a product class or a specification group */
IF @IDAssignToSpecTypeDef IS NOT NULL
BEGIN
    EXEC proc_AssignTableSpecification
    @IDTableVerDef = @IDTableVerDef,
    @IDAssignToSpecTypeDef = @IDAssignToSpecTypeDef,
    @IDProcState = @IDProcState OUTPUT,
    @Message = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from
        proc_CreateNewTableSpecification.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
END
```

Appendix 5: Product Class Database (PCD) System Code

```
END  
END
```

```
/* Now check how many columns are supplied by the user. Accordingly invoke the procedure  
that handle that many columns */
```

```
IF @NumOFColumns = 2  
BEGIN
```

```
EXEC proc_InsertRow2  
@IDTableVerDef = @IDTableVerDef,  
@ColumnValues = @ColumnValues,  
@ColIDMeasUnits = @ColIDMeasUnits,  
@RowContent = @RowContent,  
@IDProcState = @IDProcState OUTPUT,  
@Message = @Message OUTPUT
```

```
IF @IDProcState != 0  
BEGIN
```

```
ROLLBACK TRAN  
SELECT @Message = @Message + ' This Procedure was called from  
proc_CreateNewTableSpecification.'  
RAISERROR(@Message,1,1) WITH SETERROR  
RETURN @@ERROR
```

```
END  
ELSE  
BEGIN
```

```
COMMIT TRAN  
SELECT @Message = 'Table Specification Created Successfully. Table Specification ID is: ' +  
CAST(@IDTableSpec AS NVARCHAR) + '.'  
SELECT @Message = @Message + 'Table Specification Version is: ' + CAST(@IDTableSpecVer AS  
NVARCHAR) + '.'  
RETURN
```

```
END
```

```
END
```

```
IF @NumOFColumns = 3  
BEGIN
```

```
EXEC proc_InsertRow3  
@IDTableVerDef = @IDTableVerDef,  
@ColumnValues = @ColumnValues,  
@ColIDMeasUnits = @ColIDMeasUnits,  
@RowContent = @RowContent,  
@IDProcState = @IDProcState OUTPUT,  
@Message = @Message OUTPUT  
IF @IDProcState != 0  
BEGIN
```

```
ROLLBACK TRAN  
SELECT @Message = @Message + ' This Procedure was called from  
proc_CreateNewTableSpecification.'  
RAISERROR(@Message,1,1) WITH SETERROR  
RETURN @@ERROR
```

```
END  
ELSE  
BEGIN
```

```
COMMIT TRAN  
SELECT @Message = 'Table Specification Created Successfully. Table Specification ID is: ' +  
CAST(@IDTableSpec AS NVARCHAR) + '.'  
SELECT @Message = @Message + 'Table Specification Version is: ' + CAST(@IDTableSpecVer AS  
NVARCHAR) + '.'  
RETURN
```

```
END
```

```
END
```

```
IF @NumOFColumns = 4  
BEGIN
```

```
EXEC proc_InsertRow4  
@IDTableVerDef = @IDTableVerDef,  
@ColumnValues = @ColumnValues,  
@ColIDMeasUnits = @ColIDMeasUnits,
```

Appendix 5: Product Class Database (PCD) System Code

```
@RowContent = @RowContent,
@IDProcState = @IDProcState OUTPUT,
@Message = @Message OUTPUT
IF @IDProcState != 0
BEGIN
    ROLLBACK TRAN
    SELECT @Message = @Message + ' This Procedure was called from
proc_CreateNewTableSpecification.'
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END
ELSE
BEGIN
    COMMIT TRAN
    SELECT @Message = 'Table Specification Created Successfully. Table Specification ID is: ' +
CAST(@IDTableSpec AS NVARCHAR) + '.'
    SELECT @Message = @Message + 'Table Specification Version is: ' + CAST(@IDTableSpecVer AS
NVARCHAR) + '.'
    RETURN
END
END

IF @NumOfColumns = 5
BEGIN
    EXEC proc_InsertRow5
    @IDTableVerDef = @IDTableVerDef,
    @ColumnValues = @ColumnValues,
    @ColIDMeasUnits = @ColIDMeasUnits,
    @RowContent = @RowContent,
    @IDProcState = @IDProcState OUTPUT,
    @Message = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from
proc_CreateNewTableSpecification.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        COMMIT TRAN
        SELECT @Message = 'Table Specification Created Successfully. Table Specification ID is: ' +
CAST(@IDTableSpec AS NVARCHAR) + '.'
        SELECT @Message = @Message + 'Table Specification Version is: ' + CAST(@IDTableSpecVer AS
NVARCHAR) + '.'
        RETURN
    END
END

IF @NumOfColumns = 6
BEGIN
    EXEC proc_InsertRow6
    @IDTableVerDef = @IDTableVerDef,
    @ColumnValues = @ColumnValues,
    @ColIDMeasUnits = @ColIDMeasUnits,
    @RowContent = @RowContent,
    @IDProcState = @IDProcState OUTPUT,
    @Message = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from
proc_CreateNewTableSpecification.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
```

Appendix 5: Product Class Database (PCD) System Code

```
        COMMIT TRAN
        SELECT @Message = 'Table Specification Created Successfully. Table Specification ID is: ' +
        CAST(@IDTableSpec AS NVARCHAR) + '.'
        SELECT @Message = @Message + 'Table Specification Version is: ' + CAST(@IDTableSpecVer AS
        NVARCHAR) + '.'
        RETURN
    END
END

IF @NumOfColumns = 7
BEGIN
    EXEC proc_InsertRow7
    @IDTableVerDef = @IDTableVerDef,
    @ColumnValues = @ColumnValues,
    @ColIDMeasUnits = @ColIDMeasUnits,
    @RowContent = @RowContent,
    @IDProcState = @IDProcState OUTPUT,
    @Message = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from
        proc_CreateNewTableSpecification.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        COMMIT TRAN
        SELECT @Message = 'Table Specification Created Successfully. Table Specification ID is: ' +
        CAST(@IDTableSpec AS NVARCHAR) + '.'
        SELECT @Message = @Message + 'Table Specification Version is: ' + CAST(@IDTableSpecVer AS
        NVARCHAR) + '.'
        RETURN
    END
END

IF @NumOfColumns = 8
BEGIN
    EXEC proc_InsertRow8
    @IDTableVerDef = @IDTableVerDef,
    @ColumnValues = @ColumnValues,
    @ColIDMeasUnits = @ColIDMeasUnits,
    @RowContent = @RowContent,
    @IDProcState = @IDProcState OUTPUT,
    @Message = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from
        proc_CreateNewTableSpecification.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        COMMIT TRAN
        SELECT @Message = 'Table Specification Created Successfully. Table Specification ID is: ' +
        CAST(@IDTableSpec AS NVARCHAR) + '.'
        SELECT @Message = @Message + 'Table Specification Version is: ' + CAST(@IDTableSpecVer AS
        NVARCHAR) + '.'
        RETURN
    END
END

IF @NumOfColumns = 9
BEGIN
    EXEC proc_InsertRow9
    @IDTableVerDef = @IDTableVerDef,
```

Appendix 5: Product Class Database (PCD) System Code

```
@ColumnValues = @ColumnValues,
@ColIDMeasUnits = @ColIDMeasUnits,
@RowContent = @RowContent,
@IDProcState = @IDProcState OUTPUT,
@Message = @Message OUTPUT
IF @IDProcState != 0
BEGIN
    ROLLBACK TRAN
    SELECT @Message = @Message + ' This Procedure was called from
proc_CreateNewTableSpecification.'
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END
ELSE
BEGIN
    COMMIT TRAN
    SELECT @Message = 'Table Specification Created Successfully. Table Specification ID is: ' +
CAST(@IDTableSpec AS NVARCHAR) + '.'
    SELECT @Message = @Message + 'Table Specification Version is: ' + CAST(@IDTableSpecVer AS
NVARCHAR) + '.'
    RETURN
END
END

IF @NumOfColumns = 10
BEGIN
    EXEC proc_InsertRow10
    @IDTableVerDef = @IDTableVerDef,
    @ColumnValues = @ColumnValues,
    @ColIDMeasUnits = @ColIDMeasUnits,
    @RowContent = @RowContent,
    @IDProcState = @IDProcState OUTPUT,
    @Message = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from
proc_CreateNewTableSpecification.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        COMMIT TRAN
        SELECT @Message = 'Table Specification Created Successfully. Table Specification ID is: ' +
CAST(@IDTableSpec AS NVARCHAR) + '.'
        SELECT @Message = @Message + 'Table Specification Version is: ' + CAST(@IDTableSpecVer AS
NVARCHAR) + '.'
        RETURN
    END
END
END -- End of Proc_CreateNewTableSpecification Procedure.
GO
```

1.27 Procedure Name: **dbo.proc_CreateNewTableSpecificationVersion**

Database: PCD

Description:

This table enables creation of new table specification version.

*/

```
CREATE Procedure proc_CreateNewTableSpecificationVersion
/* Param List */
@IDTableSpec          BIGINT,
@IDTableSpecVer      MONEY = NULL,
@NumOfRows           INT = NULL,
@NumOfColumns        INT,
@ColumnValues         NVARCHAR(4000),
```

Appendix 5: Product Class Database (PCD) System Code

```

@ColIDMeasUnits          NVARCHAR(4000) = NULL,
@RowContent              NVARCHAR(100),
@TableVerDesc            NVARCHAR(4000) = NULL,
@IDAssignToSpecTypeDef  BIGINT      = NULL,
@IDTableVerDef           BIGINT      OUTPUT,
@IDProcState             TINYINT     OUTPUT,
@Message                 NVARCHAR(500) OUTPUT

AS
BEGIN

DECLARE
@IDExistingLatestVer  MONEY,
--@IDTableVerDef      BIGINT,
@Error               INT
SELECT @IDProcState = 0
SELECT @ERROR = 0

BEGIN TRAN

/*Check whether the supplied Table Specification exists. If yes, get the
existing latest version of that table specification the database */

SELECT @IDExistingLatestVer = MAX(IDTableSpecVer) FROM TableVersion
WHERE IDTableSpec = @IDTableSpec
IF @IDExistingLatestVer IS NULL
BEGIN
    ROLLBACK TRAN
    SELECT @IDProcState = 1
    SELECT @Message = 'Table Specification ID supplied does not exists. '
    SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + '. '
    SELECT @Message = @Message + 'Error occurred in Procedure proc_CreateNewTableSpecificationVersion. '
    SELECT @Message = @Message + 'Procedure is terminated abnormally. '
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END

/*IDTableSpecVer supplied should always be greate than existing latest version in the database */

IF @IDTableSpecVer IS NULL
BEGIN
    SELECT @IDTableSpecVer = FLOOR(@IDExistingLatestVer) + 1
END
ELSE
BEGIN
    IF @IDTableSpecVer < @IDExistingLatestVer
    BEGIN
        ROLLBACK TRAN
        SELECT @IDProcState = 2
        SELECT @Message = 'Supplied IDTableVersion ' + CAST(@IDTableSpecVer AS NVARCHAR) + '
is less than existing latest version in the database. '
        SELECT @Message = @Message + 'Existing lateste version is: ' + CAST(@IDExistingLatestVer AS
NVARCHAR) + '. '
        SELECT @Message = @Message + 'Please provide a version ID greater than the existing latest version
or leave it blank for the system to generate it for you. '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
NVARCHAR) + '. '
        SELECT @Message = @Message + 'Error occured in Procedure
proc_CreateNewTableSpecificationVersion. '
        SELECT @Message = @Message + 'Procedure is abnormally terminated.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
END

/*Create new IDTableVerDef for the new version of the table specification.*/

EXEC dbo.proc_GetNewID
@IDEntity = 110,

```

Appendix 5: Product Class Database (PCD) System Code

```
@IDNew = @IDTableVerDef OUTPUT,  
@Message = @Message OUTPUT,  
@IDProcState = @IDProcState OUTPUT  
  
IF @IDProcState != 0  
BEGIN  
    ROLLBACK TRAN  
    SELECT @Message = @Message + ' This Procedure was called from proc_CreateNewTableSpecificationVersion.'  
    RAISERROR(@Message,1,1) WITH SETERROR  
    RETURN @@ERROR  
END  
  
/*Create a table specification version by inserting values into the TableVersion table*/  
  
INSERT INTO TableVersion(IDTableSpec, IDTableSpecVer, IDTableVerDef,  
NumOfRows, NumOfColumns, TableVerDesc)  
VALUES (@IDTableSpec, @IDTableSpecVer, @IDTableVerDef, @NumOfRows,  
@NumOfColumns, LTRIM(RTRIM(@TableVerDesc)))  
SELECT @Error = @@ERROR  
IF @Error != 0  
BEGIN  
    ROLLBACK TRAN  
    SELECT @IDProcState = 3  
    SELECT @Message = 'An error occurred while creating the new table specification version.'  
    SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '. '  
    SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + '. '  
    SELECT @Message = @Message + 'Error occurred in Procedure proc_CreateNewTableSpecificationVersion.'  
    SELECT @Message = @Message + 'Procedure is terminated abnormally.'  
    RAISERROR(@Message,1,1) WITH SETERROR  
    RETURN @@ERROR  
END  
  
/* Assign the table specification by calling the following procedure */  
/* A Table specification can be assigned to a product class or a specification group */  
IF @IDAssignToSpecTypeDef IS NOT NULL  
BEGIN  
    EXEC proc_AssignTableSpecification  
    @IDTableVerDef = @IDTableVerDef,  
    @IDAssignToSpecTypeDef = @IDAssignToSpecTypeDef,  
    @IDProcState = @IDProcState OUTPUT,  
    @Message = @Message OUTPUT  
    IF @IDProcState != 0  
    BEGIN  
        ROLLBACK TRAN  
        SELECT @Message = @Message + ' This Procedure was called from  
proc_CreateNewTableSpecification.'  
        RAISERROR(@Message,1,1) WITH SETERROR  
        RETURN @@ERROR  
    END  
END  
  
/* Now check how many columns are supplied by the user. Accordingly invoke the procedure  
that handle that many columns */  
  
IF @NumOfColumns = 2  
BEGIN  
    EXEC proc_InsertRow2  
    @IDTableVerDef = @IDTableVerDef,  
    @ColumnValues = @ColumnValues,  
    @ColIDMeasUnits = @ColIDMeasUnits,  
    @RowContent = @RowContent,  
    @IDProcState = @IDProcState OUTPUT,  
    @Message = @Message OUTPUT  
    IF @IDProcState != 0  
    BEGIN  
        ROLLBACK TRAN  
        SELECT @Message = @Message + ' This Procedure was called from  
proc_CreateNewTableSpecificationVersion.'  
        RAISERROR(@Message,1,1) WITH SETERROR
```


Appendix 5: Product Class Database (PCD) System Code

```
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        COMMIT TRAN
        SELECT @Message = 'Table Specification version Created Successfully. Table Specification ID is: ' +
            CAST(@IDTableSpec AS NVARCHAR) + '.'
        SELECT @Message = @Message + 'Table Specification Version is: ' + CAST(@IDTableSpecVer AS
            NVARCHAR) + '.'
        RETURN
    END
END

IF @NumOfColumns = 3
BEGIN
    EXEC proc_InsertRow3
    @IDTableVerDef = @IDTableVerDef,
    @ColumnValues = @ColumnValues,
    @ColIDMeasUnits = @ColIDMeasUnits,
    @RowContent = @RowContent,
    @IDProcState = @IDProcState OUTPUT,
    @Message = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from
            proc_CreateNewTableSpecificationVersion.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        COMMIT TRAN
        SELECT @Message = 'Table Specification version Created Successfully. Table Specification ID is: ' +
            CAST(@IDTableSpec AS NVARCHAR) + '.'
        SELECT @Message = @Message + 'Table Specification Version is: ' + CAST(@IDTableSpecVer AS
            NVARCHAR) + '.'
        RETURN
    END
END

IF @NumOfColumns = 4
BEGIN
    EXEC proc_InsertRow4
    @IDTableVerDef = @IDTableVerDef,
    @ColumnValues = @ColumnValues,
    @ColIDMeasUnits = @ColIDMeasUnits,
    @RowContent = @RowContent,
    @IDProcState = @IDProcState OUTPUT,
    @Message = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from
            proc_CreateNewTableSpecificationVersion.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        COMMIT TRAN
        SELECT @Message = 'Table Specification version Created Successfully. Table Specification ID is: ' +
            CAST(@IDTableSpec AS NVARCHAR) + '.'
        SELECT @Message = @Message + 'Table Specification Version is: ' + CAST(@IDTableSpecVer AS
            NVARCHAR) + '.'
        RETURN
    END
END
```

Appendix 5: Product Class Database (PCD) System Code

```
IF @NumOFColumns = 5
BEGIN
    EXEC proc_InsertRow5
    @IDTableVerDef = @IDTableVerDef,
    @ColumnValues = @ColumnValues,
    @ColIDMeasUnits = @ColIDMeasUnits,
    @RowContent = @RowContent,
    @IDProcState = @IDProcState OUTPUT,
    @Message = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from
        proc_CreateNewTableSpecificationVersion.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        COMMIT TRAN
        SELECT @Message = 'Table Specification version Created Successfully. Table Specification ID is: ' +
        CAST(@IDTableSpec AS NVARCHAR) + ' .'
        SELECT @Message = @Message + 'Table Specification Version is: ' + CAST(@IDTableSpecVer AS
        NVARCHAR) + ' .'
        RETURN
    END
END

END

IF @NumOFColumns = 6
BEGIN
    EXEC proc_InsertRow6
    @IDTableVerDef = @IDTableVerDef,
    @ColumnValues = @ColumnValues,
    @ColIDMeasUnits = @ColIDMeasUnits,
    @RowContent = @RowContent,
    @IDProcState = @IDProcState OUTPUT,
    @Message = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from
        proc_CreateNewTableSpecificationVersion.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        COMMIT TRAN
        SELECT @Message = 'Table Specification version Created Successfully. Table Specification ID is: ' +
        CAST(@IDTableSpec AS NVARCHAR) + ' .'
        SELECT @Message = @Message + 'Table Specification Version is: ' + CAST(@IDTableSpecVer AS
        NVARCHAR) + ' .'
        RETURN
    END
END

END

IF @NumOFColumns = 7
BEGIN
    EXEC proc_InsertRow7
    @IDTableVerDef = @IDTableVerDef,
    @ColumnValues = @ColumnValues,
    @ColIDMeasUnits = @ColIDMeasUnits,
    @RowContent = @RowContent,
    @IDProcState = @IDProcState OUTPUT,
    @Message = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
```

Appendix 5: Product Class Database (PCD) System Code

```
        SELECT @Message = @Message + ' This Procedure was called from
        proc_CreateNewTableSpecificationVersion.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        COMMIT TRAN
        SELECT @Message = 'Table Specification version Created Successfully. Table Specification ID is: ' +
        CAST(@IDTableSpec AS NVARCHAR) + '. '
        SELECT @Message = @Message + 'Table Specification Version is: ' + CAST(@IDTableSpecVer AS
        NVARCHAR) + '. '
        RETURN
    END
END
END

IF @NumOfColumns = 8
BEGIN
    EXEC proc_InsertRow8
    @IDTableVerDef = @IDTableVerDef,
    @ColumnValues = @ColumnValues,
    @ColIDMeasUnits = @ColIDMeasUnits,
    @RowContent = @RowContent,
    @IDProcState = @IDProcState OUTPUT,
    @Message = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from
        proc_CreateNewTableSpecificationVersion.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        COMMIT TRAN
        SELECT @Message = 'Table Specification version Created Successfully. Table Specification ID is: ' +
        CAST(@IDTableSpec AS NVARCHAR) + '. '
        SELECT @Message = @Message + 'Table Specification Version is: ' + CAST(@IDTableSpecVer AS
        NVARCHAR) + '. '
        RETURN
    END
END
END

IF @NumOfColumns = 9
BEGIN
    EXEC proc_InsertRow9
    @IDTableVerDef = @IDTableVerDef,
    @ColumnValues = @ColumnValues,
    @ColIDMeasUnits = @ColIDMeasUnits,
    @RowContent = @RowContent,
    @IDProcState = @IDProcState OUTPUT,
    @Message = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from
        proc_CreateNewTableSpecificationVersion.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        COMMIT TRAN
        SELECT @Message = 'Table Specification version Created Successfully. Table Specification ID is: ' +
        CAST(@IDTableSpec AS NVARCHAR) + '. '
        SELECT @Message = @Message + 'Table Specification Version is: ' + CAST(@IDTableSpecVer AS
        NVARCHAR) + '. '
        RETURN
    END
END
END
```

Appendix 5: Product Class Database (PCD) System Code

```
        END
    END
    IF @NumOfColumns = 10
    BEGIN
        EXEC proc_InsertRow10
            @IDTableVerDef = @IDTableVerDef,
            @ColumnValues = @ColumnValues,
            @ColIDMeasUnits = @ColIDMeasUnits,
            @RowContent = @RowContent,
            @IDProcState = @IDProcState OUTPUT,
            @Message = @Message OUTPUT
        IF @IDProcState != 0
        BEGIN
            ROLLBACK TRAN
            SELECT @Message = @Message + ' This Procedure was called from
            proc_CreateNewTableSpecificationVersion.'
            RAISERROR(@Message,1,1) WITH SETERROR
            RETURN @@ERROR
        END
        ELSE
        BEGIN
            COMMIT TRAN
            SELECT @Message = 'Table Specification version Created Successfully. Table Specification ID is: ' +
            CAST(@IDTableSpec AS NVARCHAR) + '.'
            SELECT @Message = @Message + 'Table Specification Version is: ' + CAST(@IDTableSpecVer AS
            NVARCHAR) + '.'
            RETURN
        END
    END
END
END -- End of proc_CreateNewTableSpecificationVersion Procedure.
GO
```

1.28 Procedure Name: dbo. proc_CreateNewTableSpecificationVersionVersion

Database: PCD

Description:

This procedure enables creation of a new version of a table specification version.

```
*/
CREATE Procedure proc_CreateNewTableSpecificationVersionVersion
/* Param List */
@IDTableSpec          BIGINT,
@IDTableSpecVer      MONEY = NULL,
@NumOfRows           INT = NULL,
@NumOfColumns        INT,
@ColumnValues        NVARCHAR(4000),
@ColIDMeasUnits      NVARCHAR(4000) = NULL,
@RowContent          NVARCHAR(100),
@TableVerDesc        NVARCHAR(4000) = NULL,
@IDAssignToSpecTypeDef BIGINT = NULL,
@IDProcState         TINYINT OUTPUT,
@Message             NVARCHAR(500) OUTPUT

AS
BEGIN
    DECLARE
    @IDExistingLatestVer MONEY,
    @IDTableVerDef      BIGINT,
    @Error              INT
    SELECT @IDProcState = 0
    SELECT @ERROR = 0

    BEGIN TRAN

    /*Check whether the supplied Table Specification exists. If yes, get the
    existing latest version of that table specification the database */
```

Appendix 5: Product Class Database (PCD) System Code

```
SELECT @IDExistingLatestVer = MAX(IDTableSpecVer) FROM TableVersion
WHERE IDTableSpec = @IDTableSpec

IF @IDExistingLatestVer IS NULL
BEGIN
    ROLLBACK TRAN
    SELECT @IDProcState = 1
    SELECT @Message = 'Table Specification ID supplied does not exists. '
    SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + '. '
    SELECT @Message = @Message + 'Error occurred in Procedure
proc_CreateNewTableSpecificationVersionVersion. '
    SELECT @Message = @Message + 'Procedure is terminated abnormally. '
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END

/*IDTableSpecVer supplied should always be greate than existing latest version in the database */

IF @IDTableSpecVer IS NULL
BEGIN
    SELECT @IDTableSpecVer = FLOOR(@IDExistingLatestVer) + 1
END
ELSE
BEGIN
    IF @IDTableSpecVer < @IDExistingLatestVer
    BEGIN
        ROLLBACK TRAN
        SELECT @IDProcState = 2
        SELECT @Message = 'Supplied IDTableVersion ' + CAST(@IDTableSpecVer AS NVARCHAR) + '
is less than existing latest version in the database. '
        SELECT @Message = @Message + 'Existing lateste version is: ' + CAST(@IDExistingLatestVer AS
NVARCHAR) + '. '
        SELECT @Message = @Message + 'Please provide a version ID greater than the existing latest version
or leave it blank for the system to generate it for you. '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
NVARCHAR) + '. '
        SELECT @Message = @Message + 'Error occurred in Procedure
proc_CreateNewTableSpecificationVersionVersion. '
        SELECT @Message = @Message + 'Procedure is abnormally terminated.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
END

/*Create new IDTableVerDef for the new version of the table specification.*/

EXEC dbo.proc_GetNewID
@IDEntity = 110,
@IDNew = @IDTableVerDef OUTPUT,
@Message = @Message OUTPUT,
@IDProcState = @IDProcState OUTPUT

IF @IDProcState != 0
BEGIN
    ROLLBACK TRAN
    SELECT @Message = @Message + ' This Procedure was called from
proc_CreateNewTableSpecificationVersionVersion. '
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END

/*Create a table specification version by inserting values into the TableVersion table*/

INSERT INTO TableVersion(IDTableSpec, IDTableSpecVer, IDTableVerDef,
NumOfRows, NumOfColumns, TableVerDesc)
VALUES (@IDTableSpec, @IDTableSpecVer, @IDTableVerDef, @NumOfRows,
```

Appendix 5: Product Class Database (PCD) System Code

```
@NumOfColumns, LTRIM(RTRIM(@TableVerDesc)))
SELECT @Error = @@ERROR
IF @Error != 0
BEGIN
    ROLLBACK TRAN
    SELECT @IDProcState = 3
    SELECT @Message = 'An error occured while creating the new table specification version.'
    SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
    SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + ' '
    SELECT @Message = @Message + 'Error occured in Procedure
proc_CreateNewTableSpecificationVersionVersion.'
    SELECT @Message = @Message + 'Procedure is terminated abnormally.'
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END

/* Now check how many columns are supplied by the user. Accordingly invoke the procedure
that handle that many columns */

IF @NumOFColumns = 2
BEGIN
    EXEC proc_InsertRow2
    @IDTableVerDef = @IDTableVerDef,
    @ColumnValues = @ColumnValues,
    @ColIDMeasUnits = @ColIDMeasUnits,
    @RowContent = @RowContent,
    @IDProcState = @IDProcState OUTPUT,
    @Message = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from
proc_CreateNewTableSpecificationVersion.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        COMMIT TRAN
        SELECT @Message = 'Table Specification Created Successfully. Table Specification ID is: ' +
CAST(@IDTableSpec AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Table Specification Version is: ' + CAST(@IDTableSpecVer AS
NVARCHAR) + ' '
        RETURN
    END
END

IF @NumOFColumns = 3
BEGIN
    EXEC proc_InsertRow3
    @IDTableVerDef = @IDTableVerDef,
    @ColumnValues = @ColumnValues,
    @ColIDMeasUnits = @ColIDMeasUnits,
    @RowContent = @RowContent,
    @IDProcState = @IDProcState OUTPUT,
    @Message = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from
proc_CreateNewTableSpecificationVersion.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        COMMIT TRAN
        SELECT @Message = 'Table Specification Created Successfully. Table Specification ID is: ' +
CAST(@IDTableSpec AS NVARCHAR) + ' '

```

Appendix 5: Product Class Database (PCD) System Code

```
        SELECT @Message = @Message + 'Table Specification Version is: ' + CAST(@IDTableSpecVer AS
        NVARCHAR) + '.'
        RETURN
    END
END

IF @NumOfColumns = 4
BEGIN
    EXEC proc_InsertRow4
    @IDTableVerDef = @IDTableVerDef,
    @ColumnValues = @ColumnValues,
    @ColIDMeasUnits = @ColIDMeasUnits,
    @RowContent = @RowContent,
    @IDProcState = @IDProcState OUTPUT,
    @Message = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from
        proc_CreateNewTableSpecificationVersion.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        COMMIT TRAN
        SELECT @Message = 'Table Specification Created Successfully. Table Specification ID is: ' +
        CAST(@IDTableSpec AS NVARCHAR) + '.'
        SELECT @Message = @Message + 'Table Specification Version is: ' + CAST(@IDTableSpecVer AS
        NVARCHAR) + '.'
        RETURN
    END
END

IF @NumOfColumns = 5
BEGIN
    EXEC proc_InsertRow5
    @IDTableVerDef = @IDTableVerDef,
    @ColumnValues = @ColumnValues,
    @ColIDMeasUnits = @ColIDMeasUnits,
    @RowContent = @RowContent,
    @IDProcState = @IDProcState OUTPUT,
    @Message = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from
        proc_CreateNewTableSpecificationVersion.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        COMMIT TRAN
        SELECT @Message = 'Table Specification Created Successfully. Table Specification ID is: ' +
        CAST(@IDTableSpec AS NVARCHAR) + '.'
        SELECT @Message = @Message + 'Table Specification Version is: ' + CAST(@IDTableSpecVer AS
        NVARCHAR) + '.'
        RETURN
    END
END

IF @NumOfColumns = 6
BEGIN
    EXEC proc_InsertRow6
    @IDTableVerDef = @IDTableVerDef,
    @ColumnValues = @ColumnValues,
    @ColIDMeasUnits = @ColIDMeasUnits,
    @RowContent = @RowContent,
```

Appendix 5: Product Class Database (PCD) System Code

```
@IDProcState = @IDProcState OUTPUT,  
@Message = @Message OUTPUT  
IF @IDProcState != 0  
BEGIN  
    ROLLBACK TRAN  
    SELECT @Message = @Message + ' This Procedure was called from  
proc_CreateNewTableSpecificationVersion.'  
    RAISERROR(@Message,1,1) WITH SETERROR  
    RETURN @@ERROR  
END  
ELSE  
BEGIN  
    COMMIT TRAN  
    SELECT @Message = 'Table Specification Created Successfully. Table Specification ID is: '+  
CAST(@IDTableSpec AS NVARCHAR) + '.'  
    SELECT @Message = @Message + 'Table Specification Version is: ' + CAST(@IDTableSpecVer AS  
NVARCHAR) + '.'  
    RETURN  
END  
END  
END  
IF @NumOfColumns = 7  
BEGIN  
    EXEC proc_InsertRow7  
    @IDTableVerDef = @IDTableVerDef,  
    @ColumnValues = @ColumnValues,  
    @CollIDMeasUnits = @CollIDMeasUnits,  
    @RowContent = @RowContent,  
    @IDProcState = @IDProcState OUTPUT,  
    @Message = @Message OUTPUT  
    IF @IDProcState != 0  
    BEGIN  
        ROLLBACK TRAN  
        SELECT @Message = @Message + ' This Procedure was called from  
proc_CreateNewTableSpecificationVersion.'  
        RAISERROR(@Message,1,1) WITH SETERROR  
        RETURN @@ERROR  
    END  
    ELSE  
    BEGIN  
        COMMIT TRAN  
        SELECT @Message = 'Table Specification Created Successfully. Table Specification ID is: '+  
CAST(@IDTableSpec AS NVARCHAR) + '.'  
        SELECT @Message = @Message + 'Table Specification Version is: ' + CAST(@IDTableSpecVer AS  
NVARCHAR) + '.'  
        RETURN  
    END  
END  
END  
IF @NumOfColumns = 8  
BEGIN  
    EXEC proc_InsertRow8  
    @IDTableVerDef = @IDTableVerDef,  
    @ColumnValues = @ColumnValues,  
    @CollIDMeasUnits = @CollIDMeasUnits,  
    @RowContent = @RowContent,  
    @IDProcState = @IDProcState OUTPUT,  
    @Message = @Message OUTPUT  
    IF @IDProcState != 0  
    BEGIN  
        ROLLBACK TRAN  
        SELECT @Message = @Message + ' This Procedure was called from  
proc_CreateNewTableSpecificationVersion.'  
        RAISERROR(@Message,1,1) WITH SETERROR  
        RETURN @@ERROR  
    END  
    ELSE  
    BEGIN  
        COMMIT TRAN
```


Appendix 5: Product Class Database (PCD) System Code

```
        SELECT @Message = 'Table Specification Created Successfully. Table Specification ID is: ' +
        CAST(@IDTableSpec AS NVARCHAR) + '. '
        SELECT @Message = @Message + 'Table Specification Version is: ' + CAST(@IDTableSpecVer AS
        NVARCHAR) + '. '
        RETURN
    END
END

IF @NumOfColumns = 9
BEGIN
    EXEC proc_InsertRow9
    @IDTableVerDef = @IDTableVerDef,
    @ColumnValues = @ColumnValues,
    @ColIDMeasUnits = @ColIDMeasUnits,
    @RowContent = @RowContent,
    @IDProcState = @IDProcState OUTPUT,
    @Message = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from
        proc_CreateNewTableSpecificationVersion.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        COMMIT TRAN
        SELECT @Message = 'Table Specification Created Successfully. Table Specification ID is: ' +
        CAST(@IDTableSpec AS NVARCHAR) + '. '
        SELECT @Message = @Message + 'Table Specification Version is: ' + CAST(@IDTableSpecVer AS
        NVARCHAR) + '. '
        RETURN
    END
END

IF @NumOfColumns = 10
BEGIN
    EXEC proc_InsertRow10
    @IDTableVerDef = @IDTableVerDef,
    @ColumnValues = @ColumnValues,
    @ColIDMeasUnits = @ColIDMeasUnits,
    @RowContent = @RowContent,
    @IDProcState = @IDProcState OUTPUT,
    @Message = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from
        proc_CreateNewTableSpecificationVersion.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        COMMIT TRAN
        SELECT @Message = 'Table Specification Created Successfully. Table Specification ID is: ' +
        CAST(@IDTableSpec AS NVARCHAR) + '. '
        SELECT @Message = @Message + 'Table Specification Version is: ' + CAST(@IDTableSpecVer AS
        NVARCHAR) + '. '
        RETURN
    END
END
END -- End of proc_CreateNewTableSpecificationVersion Procedure.
GO
```

Appendix 5: Product Class Database (PCD) System Code

```
/*
1.29 Procedure Name: dbo.proc_GetNewID
Database: PCD
Description:
This procedure enables generation of new ID for different entities such as product class and specification.
*/

CREATE PROCEDURE dbo.proc_GetNewID
@IDEntity      INT,
@IDNew         BIGINT OUTPUT,
@Message       NVARCHAR(500) OUTPUT,
@IDProcState   TINYINT OUTPUT
AS
BEGIN
    DECLARE
    @RowsAffected INT

    SELECT @IDProcState = 0
    SELECT @Message = 'Procuedure Executed Successfully.'
    IF NOT EXISTS (SELECT [IDEntity] FROM Entity WHERE [IDEntity] = @IDEntity)
    BEGIN
        SELECT @IDProcState = 1
        SELECT @Message = 'An Entity with supplied IDENTITY ' + CAST(@IDEntity AS NVARCHAR) + ' does not exist.'
        SELECT @Message = @Message + 'Error occured in Procedure proc_getNewID.'
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + '.'
        SELECT @Message = @Message + 'Procedure is abnormally terminated.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        SELECT @IDNew = IDAvailable FROM ENTITY WHERE [IDEntity] = @IDENTITY
    END

    /* Call fn_IncrementID function to increment the IDNew. @IDNew parameter is the ID to be incremented
    and DEFAULT parameter is the number by which it is to be incremented. The DEFAULT increment
    number set is 1 */

    UPDATE Entity SET IDAvailable = dbo.fn_IncrementID(@IDNew, DEFAULT)
    WHERE [IDEntity] = @IDEntity
    SELECT @RowsAffected = @@RowCount

    --PRINT 'Rows affected are ' + CAST(@RowsAffected AS NVARCHAR)
    --PRINT '@Error id is: ' + CAST(@Error AS NVARCHAR)
    /* When the above update fails the following error is raised.*/
    IF @RowsAffected = 0
    BEGIN
        SELECT @IDProcState = 2
        SELECT @Message = 'An unknown error occured in Procedure Proc_GetNewID.'
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + '.'
        SELECT @Message = @Message + 'Please contact your system administrator.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
END
GO
```

```
/*
1.30 Procedure Name: dbo.proc_InsertListValues
Database: PCD
```

```
Description:
This table enables creation of list values for list specification.
*/
```

Appendix 5: Product Class Database (PCD) System Code

```
CREATE Procedure proc_InsertListValues
/* Param List */
@IDListDef          BIGINT,
@ListValues         NVARCHAR(4000),
@ListIDMeasUnits   NVARCHAR(4000) = NULL,
@IDProcState       TINYINT OUTPUT,
@Message           NVARCHAR(500) OUTPUT

AS
BEGIN

    DECLARE
    @Error          INT
    SELECT @IDProcState = 0
    SELECT @ERROR = 0
    SELECT @Message = ''

    /*Create a temporary table and insert the values from the table returned
    by the function fn_getListValuesTable */

    CREATE TABLE #TempTableListValues (ListID INT, ListValue NVARCHAR(255) )
    INSERT INTO #TempTableListValues
    SELECT ColumnID, ColVal FROM dbo.fn_getColumnValuesTable(@ListValues)

    /*Create another temporary table and insert IDMeas units from the table returned
    by the function fn_getListValuesTable */

    CREATE TABLE #TempTableIDMeasUnit (MeasID INT, IDMeasUnit BIGINT )
    INSERT INTO #TempTableIDMeasUnit
    SELECT ColumnID, CAST(ColVal AS BIGINT) FROM dbo.fn_getColumnValuesTable(@ListIDMeasUnits)
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        SELECT @IDProcState = 1
        SELECT @Message = 'An error ocured while inserting the list specification. '
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '. '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + '. '
        SELECT @Message = @Message + 'Error occurred in Procedure proc_InsertListValues. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END

    /* Now insert the values into the List Definition table */
    INSERT INTO ListDefinition (IDListDef, ListValue, IDMeasUnit)
    SELECT @IDListDef, a.ListValue, b.IDMeasUnit
    FROM #TempTableListValues a, #TempTableIDMeasUnit b
    WHERE a.ListID = b.MeasID

    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        SELECT @IDProcState = 2
        SELECT @Message = 'An error ocured while inserting the list specification. '
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '. '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + '. '
        SELECT @Message = @Message + 'Error occurred in Procedure proc_InsertListValues. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        DROP TABLE #TempTableListValues
        DROP TABLE #TempTableIDMeasUnit
        RETURN
    END
END
END -- End of proc_InsertListValues Procedure.
GO
```

Appendix 5: Product Class Database (PCD) System Code

```
/*
1.31 Procedure Name: dbo.proc_InsertRow2
Database: PCD
Description:
This procedure enables creation of table specification values having 2 columns.
*/

CREATE Procedure proc_InsertRow2
/* Param List */
@IDTableVerDef          BIGINT,
@ColumnValues           NVARCHAR(4000),
@ColIDMeasUnits        NVARCHAR(4000) = NULL,
@RowContent            NVARCHAR(100),
@IDProcState           TINYINT OUTPUT,
@Message              NVARCHAR(500) OUTPUT

AS
BEGIN
    DECLARE
    @Error              INT,
    @IDRow             INT,
    @NumOfRows         INT,
    @Column1Val        NVARCHAR(255),
    @Column2Val        NVARCHAR(255)

    SELECT @IDProcState = 0
    SELECT @ERROR = 0
    SELECT @Message = ''

    SELECT @NumOfRows = COUNT(*) FROM dbo.fn_getColumnValuesTable(@ColumnValues)
    PRINT '@NumOfRows: ' + CAST(@NumOfRows AS NVARCHAR)
    /* The function fn_getColumnValuesTable(@ColumnValues) should return two rows only */

    IF @NumOfRows != 2
    BEGIN
        SELECT @IDProcState = 1
        SELECT @Message = 'An error occurred while inserting the table row. '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Error occurred in Procedure proc_InsertRow2. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN
    END

    /*Create a temporary table and insert the values from the table returned
    by the function fn_getColumnValuesTable */

    CREATE TABLE #TempTable (ColumnID INT, ColumnValue NVARCHAR(255) )
    INSERT INTO #TempTable
    SELECT ColumnID,ColVal FROM dbo.fn_getColumnValuesTable(@ColumnValues)
    SELECT @Column1Val = ColumnValue FROM #TempTable WHERE ColumnID = 1
    SELECT @Column2Val = ColumnValue FROM #TempTable WHERE ColumnID = 2

    /*Check whether the row is contains column values or column specifications
    A row can contain column values or column specifications(headers).*/

    IF @RowContent LIKE 'ColVals'
    BEGIN
        /* Check for the existing rows with same IDTableVerDef to keep track of the number
        of rows the IDTableVerDef has if the @RowContent = 'ColumnValues' */
        SELECT @IDRow = MAX(IDRow) FROM TableDefinition2 WHERE IDTableVerDef = @IDTableVerDef
        IF @IDRow IS NULL
            SELECT @IDRow = 1
        ELSE
            SELECT @IDRow = @IDRow + 1

        INSERT INTO TableDefinition2(IDTableVerDef, IDRow, Column1Val, Column2Val, RowContent)

```

Appendix 5: Product Class Database (PCD) System Code

```
VALUES(@IDTableVerDef, @IDRow, LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),
LTRIM(RTRIM(@RowContent)))
SELECT @Error = @@ERROR
IF @Error != 0
BEGIN
    SELECT @IDProcState = 2
    SELECT @Message = 'An error ocured while inserting the table row. '
    SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
    SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
    NVARCHAR) + ' '
    SELECT @Message = @Message + 'Error ocured in Procedure proc_InsertRow2. '
    SELECT @Message = @Message + 'Procedure is terminated abnormally. '
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END
ELSE
BEGIN
    DROP TABLE #TempTable
    RETURN
END
END --End of IF @RowContent LIKE 'TableRow'
ELSE
BEGIN
    SELECT @IDRow = 0
    INSERT INTO TableDefinition2(IDTableVerDef, IDRow, Column1Val, Column2Val, RowContent)
    VALUES(@IDTableVerDef, @IDRow, LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),
    LTRIM(RTRIM(@RowContent)))
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        SELECT @IDProcState = 3
        SELECT @Message = 'An error ocured while inserting the table row. '
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
        NVARCHAR) + ' '
        SELECT @Message = @Message + 'Error ocured in Procedure proc_InsertRow2. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        DROP TABLE #TempTable
        RETURN
    END
END
END -- End of Proc_InsertRow2 Procedure.
GO
```

```
/*
1.32 Procedure Name: dbo.proc_InsertRow3
Database: PCD
Description:
This procedure enables creation of table specification values having 3 columns.
*/
```

```
CREATE Procedure proc_InsertRow3
/* Param List */
@IDTableVerDef          BIGINT,
@ColumnValues           NVARCHAR(4000),
@CollIDMeasUnits       NVARCHAR(4000) = NULL,
@RowContent            NVARCHAR(100),
@IDProcState           TINYINT OUTPUT,
@Message               NVARCHAR(500) OUTPUT

AS
BEGIN
    DECLARE
```

Appendix 5: Product Class Database (PCD) System Code

```
@Error          INT,
@IDRow          INT,
@NumOfRows     INT,
@Column1Val     NVARCHAR(255),
@Column2Val     NVARCHAR(255),
@Column3Val     NVARCHAR(255)
SELECT @IDProcState = 0
SELECT @ERROR = 0
SELECT @Message = ''
SELECT @NumOfRows = COUNT(*) FROM dbo.fn_getColumnValuesTable(@ColumnValues)
PRINT '@NumOfRows: ' + CAST(@NumOfRows AS NVARCHAR)
/* The function fn_getColumnValuesTable(@ColumnValues) should return three rows only */
IF @NumOfRows != 3
BEGIN
    SELECT @IDProcState = 1
    SELECT @Message = 'An error occured while inserting the table row. '
    SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + '. '
    SELECT @Message = @Message + 'Error occured in Procedure proc_InsertRow3. '
    SELECT @Message = @Message + 'Procedure is terminated abnormally. '
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN
END

/*Create a temporary table and insert the values from the table returned
by the function fn_getColumnValuesTable */

CREATE TABLE #TempTable (ColumnID INT, ColumnValue NVARCHAR(255) )
INSERT INTO #TempTable
SELECT ColumnID,ColVal FROM dbo.fn_getColumnValuesTable(@ColumnValues)

SELECT @Column1Val = ColumnValue FROM #TempTable WHERE ColumnID = 1
SELECT @Column2Val = ColumnValue FROM #TempTable WHERE ColumnID = 2
SELECT @Column3Val = ColumnValue FROM #TempTable WHERE ColumnID = 3

/*Check whether the row is contains column values or column specifications
A row can contain column values or column specifications(headers).*/

IF @RowContent LIKE 'ColVals'
BEGIN
    /* Check for the existing rows with same IDTableVerDef to keep track of the number
of rows the IDTableVerDef has if the @RowContent = 'ColumnValues' */
    SELECT @IDRow = MAX(IDRow) FROM TableDefinition3 WHERE IDTableVerDef = @IDTableVerDef
    IF @IDRow IS NULL
        SELECT @IDRow = 1
    ELSE
        SELECT @IDRow = @IDRow + 1
        INSERT INTO TableDefinition3(IDTableVerDef, IDRow, Column1Val, Column2Val, Column3Val,
RowContent)
VALUES(@IDTableVerDef, @IDRow,
LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),
LTRIM(RTRIM(@Column3Val)), LTRIM(RTRIM(@RowContent)) )
        SELECT @Error = @@ERROR
        IF @Error != 0
            BEGIN
                SELECT @IDProcState = 2
                SELECT @Message = 'An error occured while inserting the table row. '
                SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '. '
                SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
NVARCHAR) + '. '
                SELECT @Message = @Message + 'Error occured in Procedure proc_InsertRow3. '
                SELECT @Message = @Message + 'Procedure is terminated abnormally. '
                RAISERROR(@Message,1,1) WITH SETERROR
                RETURN @@ERROR
            END
        ELSE
            BEGIN
                DROP TABLE #TempTable
                RETURN
            END
END
```

Appendix 5: Product Class Database (PCD) System Code

```

        END
    END --End of IF @RowContent LIKE 'TableRow'
    ELSE
    BEGIN
        SELECT @IDRow = 0
        INSERT INTO TableDefinition3(IDTableVerDef, IDRow, Column1Val, Column2Val,Column3Val,
        RowContent)
        VALUES(@IDTableVerDef, @IDRow,
        LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),
        LTRIM(RTRIM(@Column3Val)),LTRIM(RTRIM(@RowContent)))
        SELECT @Error = @@ERROR
        IF @Error != 0
        BEGIN
            SELECT @IDProcState = 3
            SELECT @Message = 'An error ocured while inserting the table row.'
            SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '
            '

            SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
            NVARCHAR) + ' '
            SELECT @Message = @Message + 'Error occurred in Procedure proc_InsertRow3.'
            SELECT @Message = @Message + 'Procedure is terminated abnormally.'
            RAISERROR(@Message,1,1) WITH SETERROR
            RETURN @@ERROR
        END
    END
    BEGIN
        DROP TABLE #TempTable
        RETURN
    END
END

END
END -- End of Proc_InsertRow3 Procedure.
GO

/*
1.33 Procedure Name: dbo.proc_InsertRow4
Database: PCD
Description:
This procedure enables creation of table specification values having 4 columns.
*/

CREATE Procedure proc_InsertRow4
/* Param List */
@IDTableVerDef      BIGINT,
@ColumnValues        NVARCHAR(4000),
@ColIDMeasUnits     NVARCHAR(4000) = NULL,
@RowContent         NVARCHAR(100),
@IDProcState        TINYINT OUTPUT,
@Message            NVARCHAR(500) OUTPUT
AS
BEGIN
    DECLARE
    @Error            INT,
    @IDRow           INT,
    @NumOfRows       INT,
    @Column1Val      NVARCHAR(255),
    @Column2Val      NVARCHAR(255),
    @Column3Val      NVARCHAR(255),
    @Column4Val      NVARCHAR(255)

    SELECT @IDProcState = 0
    SELECT @ERROR = 0
    SELECT @Message = ''

    SELECT @NumOfRows = COUNT(*) FROM dbo.fn_getColumnValuesTable(@ColumnValues)
    --PRINT '@NumOfRows: ' + CAST(@NumOfRows AS NVARCHAR)
    /* The function fn_getColumnValuesTable(@ColumnValues) should return two rows only */

```

Appendix 5: Product Class Database (PCD) System Code

```

IF @NumOfRows != 4
BEGIN
    SELECT @IDProcState = 1
    SELECT @Message = 'An error occured while inserting the table row. '
    SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + ' '
    SELECT @Message = @Message + 'Error occured in Procedure proc_InsertRow4. '
    SELECT @Message = @Message + 'Procedure is terminated abnormally. '
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN
END

/*Create a temporary table and insert the values from the table returned
by the function fn_getColumnValuesTable */

CREATE TABLE #TempTable (ColumnID INT, ColumnValue NVARCHAR(255) )
INSERT INTO #TempTable
SELECT ColumnID,ColVal FROM dbo.fn_getColumnValuesTable(@ColumnValues)

SELECT @Column1Val = ColumnValue FROM #TempTable WHERE ColumnID = 1
SELECT @Column2Val = ColumnValue FROM #TempTable WHERE ColumnID = 2
SELECT @Column3Val = ColumnValue FROM #TempTable WHERE ColumnID = 3
SELECT @Column4Val = ColumnValue FROM #TempTable WHERE ColumnID = 4

/*Check whether the row contains column values or column specifications
A row can only contain column values or column specifications(headers).*/

IF @RowContent LIKE 'ColVals'
BEGIN
    /* Check for the existing rows with same IDTableVerDef to keep track of the number
    of rows the IDTableVerDef has if the @RowContent = 'ColumnValues' */
    SELECT @IDRow = MAX(IDRow) FROM TableDefinition4 WHERE IDTableVerDef = @IDTableVerDef
    IF @IDRow IS NULL
        SELECT @IDRow = 1
    ELSE
        SELECT @IDRow = @IDRow + 1
        INSERT INTO TableDefinition4(IDTableVerDef, IDRow, Column1Val,
        Column2Val,Column3Val,Column4Val, RowContent)
        VALUES(@IDTableVerDef, @IDRow,
        LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),
        LTRIM(RTRIM(@Column3Val)),
        LTRIM(RTRIM(@Column4Val)), LTRIM(RTRIM(@RowContent)) )
        SELECT @Error = @@ERROR

        IF @Error != 0
        BEGIN
            SELECT @IDProcState = 2
            SELECT @Message = 'An error occured while inserting the table row. '
            SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
            SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
            NVARCHAR) + ' '
            SELECT @Message = @Message + 'Error occured in Procedure proc_InsertRow4. '
            SELECT @Message = @Message + 'Procedure is terminated abnormally. '
            RAISERROR(@Message,1,1) WITH SETERROR
            RETURN @@ERROR
        END
    ELSE
    BEGIN
        DROP TABLE #TempTable
        RETURN
    END
END --End of IF @RowContent LIKE 'TableRow'
ELSE
BEGIN
    SELECT @IDRow = 0
    INSERT INTO TableDefinition4(IDTableVerDef, IDRow, Column1Val, Column2Val, Column3Val,
    Column4Val, RowContent)

```


Appendix 5: Product Class Database (PCD) System Code

```
VALUES(@IDTableVerDef, @IDRow, LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),
LTRIM(RTRIM(@Column3Val)),LTRIM(RTRIM(@Column4Val)),LTRIM(RTRIM(@RowContent)) )
SELECT @Error = @@ERROR
IF @Error != 0
BEGIN
    SELECT @IDProcState = 3
    SELECT @Message = 'An error occurred while inserting the table row.'
    SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
    SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
    NVARCHAR) + ' '
    SELECT @Message = @Message + 'Error occurred in Procedure proc_InsertRow4.'
    SELECT @Message = @Message + 'Procedure is terminated abnormally.'
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END
ELSE
BEGIN
    DROP TABLE #TempTable
    RETURN
END
END
END -- End of Proc_InsertRow4 Procedure.
GO
```

```
/*
1.34 Procedure Name: dbo.proc_InsertRow5
Database: PCD
Description:
This procedure enables creation of table specification values having 5 columns.
*/
```

```
CREATE Procedure proc_InsertRow5
/* Param List */
@IDTableVerDef          BIGINT,
@ColumnValues            NVARCHAR(4000),
@ColIDMeasUnits         NVARCHAR(4000) = NULL,
@RowContent             NVARCHAR(100),
@IDProcState            TINYINT OUTPUT,
@Message                NVARCHAR(500) OUTPUT

AS
BEGIN
    DECLARE
    @Error                INT,
    @IDRow                INT,
    @NumOfRows            INT,
    @Column1Val           NVARCHAR(255),
    @Column2Val           NVARCHAR(255),
    @Column3Val           NVARCHAR(255),
    @Column4Val           NVARCHAR(255),
    @Column5Val           NVARCHAR(255)

    SELECT @IDProcState = 0
    SELECT @ERROR = 0
    SELECT @Message = ''

    SELECT @NumOfRows = COUNT(*) FROM dbo.fn_getColumnValuesTable(@ColumnValues)
    --PRINT '@NumOfRows: ' + CAST(@NumOfRows AS NVARCHAR)
    /* The function fn_getColumnValuesTable(@ColumnValues) should return two rows only */

    IF @NumOfRows != 5
    BEGIN
        SELECT @IDProcState = 1
        SELECT @Message = 'An error occurred while inserting the table row.'
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Error occurred in Procedure proc_InsertRow5.'
        SELECT @Message = @Message + 'Procedure is terminated abnormally.'
    END
END
```

Appendix 5: Product Class Database (PCD) System Code

```

        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN
    END

/*Create a temporary table and insert the values from the table returned
by the function fn_getColumnValuesTable */

CREATE TABLE #TempTable (ColumnID INT, ColumnValue NVARCHAR(255) )
INSERT INTO #TempTable
SELECT ColumnID,ColVal FROM dbo.fn_getColumnValuesTable(@ColumnValues)

SELECT @Column1Val = ColumnValue FROM #TempTable WHERE ColumnID = 1
SELECT @Column2Val = ColumnValue FROM #TempTable WHERE ColumnID = 2
SELECT @Column3Val = ColumnValue FROM #TempTable WHERE ColumnID = 3
SELECT @Column4Val = ColumnValue FROM #TempTable WHERE ColumnID = 4
SELECT @Column5Val = ColumnValue FROM #TempTable WHERE ColumnID = 5

/*Check whether the row contains column values or column specifications
A row can only contain column values or column specifications(headers).*/
IF @RowContent LIKE 'ColVals'
BEGIN
    /* Check for the existing rows with same IDTableVerDef to keep track of the number
    of rows the IDTableVerDef has if the @RowContent = 'ColumnValues' */
    SELECT @IDRow = MAX(IDRow) FROM TableDefinition2 WHERE IDTableVerDef = @IDTableVerDef
    IF @IDRow IS NULL
        SELECT @IDRow = 1
    ELSE
        SELECT @IDRow = @IDRow + 1
        INSERT INTO TableDefinition5(IDTableVerDef, IDRow, Column1Val,
        Column2Val,Column3Val,Column4Val, Column5Val, RowContent)
        VALUES(@IDTableVerDef, @IDRow,
        LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),LTRIM(RTRIM(@Column3Val))
        ,LTRIM(RTRIM(@Column4Val)),LTRIM(RTRIM(@Column5Val)),
        LTRIM(RTRIM(@RowContent)) )
        SELECT @Error = @@ERROR
        IF @Error != 0
        BEGIN
            SELECT @IDProcState = 2
            SELECT @Message = 'An error occured while inserting the table row. '
            SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '
            '

            SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
            NVARCHAR) + '
            '
            SELECT @Message = @Message + 'Error occured in Procedure proc_InsertRow5. '
            SELECT @Message = @Message + 'Procedure is terminated abnormally. '
            RAISERROR(@Message,1,1) WITH SETERROR
            RETURN @@ERROR
        END
    ELSE
    BEGIN
        DROP TABLE #TempTable
        RETURN
    END
END --End of IF @RowContent LIKE 'TableRow'
ELSE
BEGIN
    SELECT @IDRow = 0
    INSERT INTO TableDefinition5(IDTableVerDef, IDRow, Column1Val, Column2Val, Column3Val,
    Column4Val, Column5Val, RowContent)
    VALUES(@IDTableVerDef, @IDRow, LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),
    LTRIM(RTRIM(@Column3Val)),LTRIM(RTRIM(@Column4Val)),
    LTRIM(RTRIM(@Column5Val)),
    LTRIM(RTRIM(@RowContent)) )
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        SELECT @IDProcState = 3
        SELECT @Message = 'An error occured while inserting the table row. '
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '

```

Appendix 5: Product Class Database (PCD) System Code

```
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
        NVARCHAR) + ', '
        SELECT @Message = @Message + 'Error occured in Procedure proc_InsertRow5. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        DROP TABLE #TempTable
        RETURN
    END
END
END -- End of Proc_InsertRow5 Procedure.
GO
```

/*

1.35 Procedure Name: dbo.proc_InsertRow6

Database: PCD

Description:

This procedure enables creation of table specification values having 6 columns.

*/

```
CREATE Procedure proc_InsertRow6
/* Param List */
@IDTableVerDef      BIGINT,
@ColumnValues        NVARCHAR(4000),
@ColIDMeasUnits     NVARCHAR(4000) = NULL,
@RowContent         NVARCHAR(100),
@IDProcState        TINYINT OUTPUT,
@Message            NVARCHAR(500) OUTPUT
AS
BEGIN
    DECLARE
        @Error          INT,
        @IDRow          INT,
        @NumOfRows      INT,
        @Column1Val     NVARCHAR(255),
        @Column2Val     NVARCHAR(255),
        @Column3Val     NVARCHAR(255),
        @Column4Val     NVARCHAR(255),
        @Column5Val     NVARCHAR(255),
        @Column6Val     NVARCHAR(255)

    SELECT @IDProcState = 0
    SELECT @ERROR = 0
    SELECT @Message = ''

    SELECT @NumOfRows = COUNT(*) FROM dbo.fn_getColumnValuesTable(@ColumnValues)
    --PRINT '@NumOfRows: ' + CAST(@NumOfRows AS NVARCHAR)
    /* The function fn_getColumnValuesTable(@ColumnValues) should return two rows only */

    IF @NumOfRows != 6
    BEGIN
        SELECT @IDProcState = 1
        SELECT @Message = 'An error occured while inserting the table row. '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + ', '
        SELECT @Message = @Message + 'Error occured in Procedure Proc_InsertRow6. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN
    END

    /*Create a temporary table and insert the values from the table returned
    by the function fn_getColumnValuesTable */
```

Appendix 5: Product Class Database (PCD) System Code

```
CREATE TABLE #TempTable (ColumnID INT, ColumnValue NVARCHAR(255) )
INSERT INTO #TempTable
SELECT ColumnID,ColVal FROM dbo.fn_getColumnValuesTable(@ColumnValues)

SELECT @Column1Val = ColumnValue FROM #TempTable WHERE ColumnID = 1
SELECT @Column2Val = ColumnValue FROM #TempTable WHERE ColumnID = 2
SELECT @Column3Val = ColumnValue FROM #TempTable WHERE ColumnID = 3
SELECT @Column4Val = ColumnValue FROM #TempTable WHERE ColumnID = 4
SELECT @Column5Val = ColumnValue FROM #TempTable WHERE ColumnID = 5
SELECT @Column6Val = ColumnValue FROM #TempTable WHERE ColumnID = 6

/*Check whether the row contains column values or column specifications
A row can only contain column values or column specifications(headers).*/

IF @RowContent LIKE 'ColVals'
BEGIN
    /* Check for the existing rows with same IDTableVerDef to keep track of the number
of rows the IDTableVerDef has if the @RowContent = 'ColumnValues' */
    SELECT @IDRow = MAX(IDRow) FROM TableDefinition6 WHERE IDTableVerDef = @IDTableVerDef
    IF @IDRow IS NULL
        SELECT @IDRow = 1
    ELSE
        SELECT @IDRow = @IDRow + 1

    INSERT INTO TableDefinition6(IDTableVerDef, IDRow, Column1Val, Column2Val,Column3Val,Column4Val,
        Column5Val, Column6Val, RowContent)
    VALUES(@IDTableVerDef, @IDRow,
    LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),LTRIM(RTRIM(@Column3Val)),
    LTRIM(RTRIM(@Column4Val)),LTRIM(RTRIM(@Column5Val)),
    LTRIM(RTRIM(@Column6Val)),LTRIM(RTRIM(@RowContent)) )
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        SELECT @IDProcState = 2
        SELECT @Message = 'An error ocured while inserting the table row.'
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '.'
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
        NVARCHAR) + '.'
        SELECT @Message = @Message + 'Error occurred in Procedure Proc_InsertRow6.'
        SELECT @Message = @Message + 'Procedure is terminated abnormally.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        DROP TABLE #TempTable
        RETURN
    END
END
END --End of IF @RowContent LIKE 'TableRow'
ELSE
BEGIN
    SELECT @IDRow = 0
    INSERT INTO TableDefinition6(IDTableVerDef, IDRow, Column1Val, Column2Val, Column3Val,
    Column4Val, Column5Val, Column6Val,RowContent)
    VALUES(@IDTableVerDef, @IDRow, LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),
    LTRIM(RTRIM(@Column3Val)),LTRIM(RTRIM(@Column4Val)),LTRIM(RTRIM(@Column5Val)),
    LTRIM(RTRIM(@Column6Val)),LTRIM(RTRIM(@RowContent)) )
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        SELECT @IDProcState = 3
        SELECT @Message = 'An error ocured while inserting the table row.'
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '.'
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
        NVARCHAR) + '.'
        SELECT @Message = @Message + 'Error occurred in Procedure Proc_InsertRow6.'
        SELECT @Message = @Message + 'Procedure is terminated abnormally.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
END
```

Appendix 5: Product Class Database (PCD) System Code

```
        END
        ELSE
        BEGIN
            DROP TABLE #TempTable
            RETURN
        END
    END
END -- End of Proc_InsertRow6 Procedure.
GO
```

```
/*
1.36 Procedure Name: dbo.proc_InsertRow7
Database: PCD
Description:
This procedure enables creation of table specification values having 7 columns.
*/
```

```
CREATE Procedure Proc_InsertRow7
/* Param List */
@IDTableVerDef          BIGINT,
@ColumnValues            NVARCHAR(4000),
@ColIDMeasUnits        NVARCHAR(4000) = NULL,
@RowContent             NVARCHAR(100),
@IDProcState           TINYINT OUTPUT,
@Message               NVARCHAR(500) OUTPUT
AS
BEGIN
    DECLARE
    @Error              INT,
    @IDRow             INT,
    @NumOfRows         INT,
    @Column1Val        NVARCHAR(255),
    @Column2Val        NVARCHAR(255),
    @Column3Val        NVARCHAR(255),
    @Column4Val        NVARCHAR(255),
    @Column5Val        NVARCHAR(255),
    @Column6Val        NVARCHAR(255),
    @Column7Val        NVARCHAR(255)

    SELECT @IDProcState = 0
    SELECT @ERROR = 0
    SELECT @Message = ''

    SELECT @NumOfRows = COUNT(*) FROM dbo.fn_getColumnValuesTable(@ColumnValues)
    --PRINT 'NumOfRows: ' + CAST(@NumOfRows AS NVARCHAR)
    /* The function fn_getColumnValuesTable(@ColumnValues) should return two rows only */

    IF @NumOfRows != 7
    BEGIN
        SELECT @IDProcState = 1
        SELECT @Message = 'An error occurred while inserting the table row.'
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + ','
        SELECT @Message = @Message + 'Error occurred in Procedure Proc_InsertRow7.'
        SELECT @Message = @Message + 'Procedure is terminated abnormally.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN
    END

    /*Create a temporary table and insert the values from the table returned
    by the function fn_getColumnValuesTable */

    CREATE TABLE #TempTable (ColumnID INT, ColumnValue NVARCHAR(255) )
    INSERT INTO #TempTable
    SELECT ColumnID,ColVal FROM dbo.fn_getColumnValuesTable(@ColumnValues)

    SELECT @Column1Val = ColumnValue FROM #TempTable WHERE ColumnID = 1
    SELECT @Column2Val = ColumnValue FROM #TempTable WHERE ColumnID = 2
    SELECT @Column3Val = ColumnValue FROM #TempTable WHERE ColumnID = 3
```

Appendix 5: Product Class Database (PCD) System Code

```

SELECT @Column4Val = ColumnValue FROM #TempTable WHERE ColumnID = 4
SELECT @Column5Val = ColumnValue FROM #TempTable WHERE ColumnID = 5
SELECT @Column6Val = ColumnValue FROM #TempTable WHERE ColumnID = 6
SELECT @Column7Val = ColumnValue FROM #TempTable WHERE ColumnID = 7

/*Check whether the row contains column values or column specifications
A row can only contain column values or column specifications(headers).*/

IF @RowContent LIKE 'ColVals'
BEGIN
    /* Check for the existing rows with same IDTableVerDef to keep track of the number
of rows the IDTableVerDef has if the @RowContent = 'ColumnValues' */
    SELECT @IDRow = MAX(IDRow) FROM TableDefinition7 WHERE IDTableVerDef = @IDTableVerDef
    IF @IDRow IS NULL
        SELECT @IDRow = 1
    ELSE
        SELECT @IDRow = @IDRow + 1

    INSERT INTO TableDefinition7(IDTableVerDef, IDRow, Column1Val, Column2Val,Column3Val,Column4Val,
        Column5Val, Column6Val,Column7Val, RowContent)
    VALUES(@IDTableVerDef, @IDRow,
    LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),LTRIM(RTRIM(@Column3Val)),
    LTRIM(RTRIM(@Column4Val)),LTRIM(RTRIM(@Column5Val)), LTRIM(RTRIM(@Column6Val)),
    LTRIM(RTRIM(@Column7Val)),LTRIM(RTRIM(@RowContent)) )
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        SELECT @IDProcState = 2
        SELECT @Message = 'An error occured while inserting the table row. '
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '. '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
        NVARCHAR) + '. '
        SELECT @Message = @Message + 'Error occured in Procedure Proc_InsertRow7. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        DROP TABLE #TempTable
        RETURN
    END
END --End of IF @RowContent LIKE 'TableRow'
ELSE
BEGIN
    SELECT @IDRow = 0
    INSERT INTO TableDefinition7(IDTableVerDef, IDRow, Column1Val, Column2Val, Column3Val,
    Column4Val, Column5Val, Column6Val,Column7Val, RowContent)
    VALUES(@IDTableVerDef, @IDRow, LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),
    LTRIM(RTRIM(@Column3Val)),LTRIM(RTRIM(@Column4Val)),LTRIM(RTRIM(@Column5Val)),
    LTRIM(RTRIM(@Column6Val)),LTRIM(RTRIM(@Column7Val)),LTRIM(RTRIM(@RowContent)) )
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        SELECT @IDProcState = 3
        SELECT @Message = 'An error occured while inserting the table row. '
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '. '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
        NVARCHAR) + '. '
        SELECT @Message = @Message + 'Error occured in Procedure Proc_InsertRow7. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        DROP TABLE #TempTable
        RETURN
    END
END
END

```

Appendix 5: Product Class Database (PCD) System Code

```
END
END -- End of Proc_InsertRow7 Procedure.
GO
```

```
*/
1.37 Procedure Name: dbo.proc_InsertRow8
```

Database: PCD

Description:

This procedure enables creation of table specification values having 8 columns.

```
*/
```

```
CREATE Procedure Proc_InsertRow8
```

```
/* Param List */
```

```
@IDTableVerDef      BIGINT,
@ColumnValues        NVARCHAR(4000),
@ColIDMeasUnits     NVARCHAR(4000) = NULL,
@RowContent         NVARCHAR(100),
@IDProcState        TINYINT OUTPUT,
@Message            NVARCHAR(500) OUTPUT
AS
```

```
BEGIN
```

```
    DECLARE
```

```
    @Error           INT,
    @IDRow           INT,
    @NumOfRows       INT,
    @Column1Val      NVARCHAR(255),
    @Column2Val      NVARCHAR(255),
    @Column3Val      NVARCHAR(255),
    @Column4Val      NVARCHAR(255),
    @Column5Val      NVARCHAR(255),
    @Column6Val      NVARCHAR(255),
    @Column7Val      NVARCHAR(255),
    @Column8Val      NVARCHAR(255)
```

```
    SELECT @IDProcState = 0
    SELECT @ERROR = 0
    SELECT @Message = ''
```

```
    SELECT @NumOfRows = COUNT(*) FROM dbo.fn_getColumnValuesTable(@ColumnValues)
    --PRINT 'NumOfRows: ' + CAST(@NumOfRows AS NVARCHAR)
    /* The function fn_getColumnValuesTable(@ColumnValues) should return two rows only */
```

```
    IF @NumOfRows != 8
```

```
    BEGIN
```

```
        SELECT @IDProcState = 1
        SELECT @Message = 'An error occurred while inserting the table row. '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + '.'
        SELECT @Message = @Message + 'Error occurred in Procedure Proc_InsertRow8. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN
```

```
    END
```

```
    /*Create a temporary table and insert the values from the table returned
    by the function fn_getColumnValuesTable */
```

```
    CREATE TABLE #TempTable (ColumnID INT, ColumnValue NVARCHAR(255))
    INSERT INTO #TempTable
    SELECT ColumnID, ColVal FROM dbo.fn_getColumnValuesTable(@ColumnValues)
```

```
    SELECT @Column1Val = ColumnValue FROM #TempTable WHERE ColumnID = 1
    SELECT @Column2Val = ColumnValue FROM #TempTable WHERE ColumnID = 2
    SELECT @Column3Val = ColumnValue FROM #TempTable WHERE ColumnID = 3
    SELECT @Column4Val = ColumnValue FROM #TempTable WHERE ColumnID = 4
    SELECT @Column5Val = ColumnValue FROM #TempTable WHERE ColumnID = 5
    SELECT @Column6Val = ColumnValue FROM #TempTable WHERE ColumnID = 6
```

Appendix 5: Product Class Database (PCD) System Code

```

SELECT @Column7Val = ColumnValue FROM #TempTable WHERE ColumnID = 7
SELECT @Column8Val = ColumnValue FROM #TempTable WHERE ColumnID = 8

/*Check whether the row contains column values or column specifications
A row can only contain column values or column specifications(headers).*/

IF @RowContent LIKE 'ColVals'
BEGIN
    /* Check for the existing rows with same IDTableVerDef to keep track of the number
of rows the IDTableVerDef has if the @RowContent = 'ColumnValues' */
    SELECT @IDRow = MAX(IDRow) FROM TableDefinition8 WHERE IDTableVerDef = @IDTableVerDef
    IF @IDRow IS NULL
        SELECT @IDRow = 1
    ELSE
        SELECT @IDRow = @IDRow + 1
        INSERT INTO TableDefinition8(IDTableVerDef, IDRow, Column1Val,
Column2Val,Column3Val,Column4Val,
Column5Val, Column6Val,Column7Val, Column8Val, RowContent)
VALUES(@IDTableVerDef, @IDRow,
LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),
LTRIM(RTRIM(@Column3Val)),
LTRIM(RTRIM(@Column4Val)),LTRIM(RTRIM(@Column5Val)),
LTRIM(RTRIM(@Column6Val)),
LTRIM(RTRIM(@Column7Val)),LTRIM(RTRIM(@Column8Val)),LTRIM(RTRIM(@RowContent)
))
        SELECT @Error = @@ERROR
        IF @Error != 0
            BEGIN
                SELECT @IDProcState = 2
                EJECT @Message = 'An error occured while inserting the table row. '
                SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '
                '
                SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
NVARCHAR) + ' '
                SELECT @Message = @Message + 'Error occured in Procedure Proc_InsertRow8. '
                SELECT @Message = @Message + 'Procedure is terminated abnormally. '
                RAISERROR(@Message,1,1) WITH SETERROR
                RETURN @@ERROR
            END
        ELSE
            BEGIN
                DROP TABLE #TempTable
                RETURN
            END
    END --End of IF @RowContent LIKE 'TableRow'
ELSE
    BEGIN
        SELECT @IDRow = 0
        INSERT INTO TableDefinition8(IDTableVerDef, IDRow, Column1Val, Column2Val, Column3Val,
Column4Val, Column5Val,
Column6Val,Column7Val, Column8Val, RowContent)
VALUES(@IDTableVerDef, @IDRow,
LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),
LTRIM(RTRIM(@Column3Val)),LTRIM(RTRIM(@Column4Val)),
LTRIM(RTRIM(@Column5Val)),
LTRIM(RTRIM(@Column6Val)),LTRIM(RTRIM(@Column7Val)),
LTRIM(RTRIM(@Column8Val)),
LTRIM(RTRIM(@RowContent)))
        SELECT @Error = @@ERROR
        IF @Error != 0
            BEGIN
                SELECT @IDProcState = 3
                SELECT @Message = 'An error occured while inserting the table row. '
                SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '
                '
                SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
NVARCHAR) + ' '
                SELECT @Message = @Message + 'Error occured in Procedure Proc_InsertRow8. '
                SELECT @Message = @Message + 'Procedure is terminated abnormally. '
            END
    END

```

Appendix 5: Product Class Database (PCD) System Code

```
                RAISERROR(@Message,1,1) WITH SETERROR
                RETURN @@ERROR
            END
            ELSE
            BEGIN
                DROP TABLE #TempTable
                RETURN
            END
        END
    END
END -- End of Proc_InsertRow8 Procedure.
GO

/*
1.38 Procedure Name: dbo.proc_InsertRow9
Database: PCD
Description:
This procedure enables creation of table specification values having 9 columns.
*/
CREATE Procedure Proc_InsertRow9
/* Param List */
@IDTableVerDef        BIGINT,
@ColumnValues          NVARCHAR(4000),
@ColIDMeasUnits       NVARCHAR(4000) = NULL,
@RowContent           NVARCHAR(100),
@IDProcState          TINYINT OUTPUT,
@Message              NVARCHAR(500) OUTPUT

AS
BEGIN
    DECLARE
    @Error              INT,
    @IDRow              INT,
    @NumOfRows          INT,
    @Column1Val         NVARCHAR(255),
    @Column2Val         NVARCHAR(255),
    @Column3Val         NVARCHAR(255),
    @Column4Val         NVARCHAR(255),
    @Column5Val         NVARCHAR(255),
    @Column6Val         NVARCHAR(255),
    @Column7Val         NVARCHAR(255),
    @Column8Val         NVARCHAR(255),
    @Column9Val         NVARCHAR(255)

    SELECT @IDProcState = 0
    SELECT @ERROR = 0
    SELECT @Message = ''

    SELECT @NumOfRows = COUNT(*) FROM dbo.fn_getColumnValuesTable(@ColumnValues)
    --PRINT '@NumOfRows: ' + CAST(@NumOfRows AS NVARCHAR)
    /* The function fn_getColumnValuesTable(@ColumnValues) should return two rows only */

    IF @NumOfRows != 9
    BEGIN
        SELECT @IDProcState = 1
        SELECT @Message = 'An error occurred while inserting the table row.'
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + '.'
        SELECT @Message = @Message + 'Error occurred in Procedure Proc_InsertRow9.'
        SELECT @Message = @Message + 'Procedure is terminated abnormally.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN
    END

    /*Create a temporary table and insert the values from the table returned
    by the function fn_getColumnValuesTable */

    CREATE TABLE #TempTable (ColumnID INT, ColumnValue NVARCHAR(255) )
    INSERT INTO #TempTable
    SELECT ColumnID,ColVal FROM dbo.fn_getColumnValuesTable(@ColumnValues)
```

Appendix 5: Product Class Database (PCD) System Code

```
SELECT @Column1Val = ColumnValue FROM #TempTable WHERE ColumnID = 1
SELECT @Column2Val = ColumnValue FROM #TempTable WHERE ColumnID = 2
SELECT @Column3Val = ColumnValue FROM #TempTable WHERE ColumnID = 3
SELECT @Column4Val = ColumnValue FROM #TempTable WHERE ColumnID = 4
SELECT @Column5Val = ColumnValue FROM #TempTable WHERE ColumnID = 5
SELECT @Column6Val = ColumnValue FROM #TempTable WHERE ColumnID = 6
SELECT @Column7Val = ColumnValue FROM #TempTable WHERE ColumnID = 7
SELECT @Column8Val = ColumnValue FROM #TempTable WHERE ColumnID = 8
SELECT @Column9Val = ColumnValue FROM #TempTable WHERE ColumnID = 9

/*Check whether the row contains column values or column specifications
A row can only contain column values or column specifications(headers).*/

IF @RowContent LIKE 'ColVals'
BEGIN
    /* Check for the existing rows with same IDTableVerDef to keep track of the number
of rows the IDTableVerDef has if the @RowContent = 'ColumnValues' */
    SELECT @IDRow = MAX(IDRow) FROM TableDefinition9 WHERE IDTableVerDef = @IDTableVerDef
    IF @IDRow IS NULL
        SELECT @IDRow = 1
    ELSE
        SELECT @IDRow = @IDRow + 1
        INSERT INTO TableDefinition9(IDTableVerDef, IDRow, Column1Val,
Column2Val,Column3Val,Column4Val,
Column5Val, Column6Val,Column7Val, Column8Val, Column9Val,RowContent)
VALUES(@IDTableVerDef, @IDRow,
LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),LTRIM(RTRIM(@Column3Val))
,
LTRIM(RTRIM(@Column4Val)),LTRIM(RTRIM(@Column5Val)),
LTRIM(RTRIM(@Column6Val)),
LTRIM(RTRIM(@Column7Val)),LTRIM(RTRIM(@Column8Val)),
LTRIM(RTRIM(@Column9Val)),
LTRIM(RTRIM(@RowContent)) )
        SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        SELECT @IDProcState = 2
        SELECT @Message = 'An error occurred while inserting the table row. '
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
NVARCHAR) + ' '
        SELECT @Message = @Message + 'Error occurred in Procedure Proc_InsertRow9. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        DROP TABLE #TempTable
        RETURN
    END
END
--End of IF @RowContent LIKE 'TableRow'
ELSE
BEGIN
    SELECT @IDRow = 0
    INSERT INTO TableDefinition9(IDTableVerDef, IDRow, Column1Val, Column2Val, Column3Val,
Column4Val, Column5Val, Column6Val,Column7Val, Column8Val,
Column9Val, RowContent)
VALUES(@IDTableVerDef, @IDRow, LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),
LTRIM(RTRIM(@Column3Val)),LTRIM(RTRIM(@Column4Val)),LTRIM(RTRIM(@Column5Val)),
LTRIM(RTRIM(@Column6Val)),LTRIM(RTRIM(@Column7Val)),LTRIM(RTRIM(@Column8Val)),
LTRIM(RTRIM(@Column9Val)),LTRIM(RTRIM(@RowContent)) )
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        SELECT @IDProcState = 3
        SELECT @Message = 'An error occurred while inserting the table row. '
```

Appendix 5: Product Class Database (PCD) System Code

```
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ', '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
        NVARCHAR) + ', '
        SELECT @Message = @Message + 'Error occured in Procedure Proc_InsertRow9. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        DROP TABLE #TempTable
        RETURN
    END
END
END -- End of Proc_InsertRow9 Procedure.
GO
```

/*

1.39 Procedure Name: dbo.proc_InsertRow10

Database: PCD

Description:

This procedure enables creation of table specification values having 10 columns.

*/

```
CREATE Procedure proc_InsertRow10
```

```
/* Param List */
```

```
@IDTableVerDef          BIGINT,
@ColumnValues            NVARCHAR(4000),
@ColIDMeasUnits         NVARCHAR(4000) = NULL,
@RowContent             NVARCHAR(100),
@IDProcState           TINYINT OUTPUT,
@Message               NVARCHAR(500) OUTPUT
```

```
AS
```

```
BEGIN
```

```
    DECLARE
```

```
    @Error              INT,
    @IDRow              INT,
    @NumOfRows          INT,
    @Column1Val         NVARCHAR(255),
    @Column2Val         NVARCHAR(255),
    @Column3Val         NVARCHAR(255),
    @Column4Val         NVARCHAR(255),
    @Column5Val         NVARCHAR(255),
    @Column6Val         NVARCHAR(255),
    @Column7Val         NVARCHAR(255),
    @Column8Val         NVARCHAR(255),
    @Column9Val         NVARCHAR(255),
    @Column10Val        NVARCHAR(255)
```

```
    SELECT @IDProcState = 0
    SELECT @ERROR = 0
    SELECT @Message = ''
```

```
    SELECT @NumOfRows = COUNT(*) FROM dbo.fn_getColumnValuesTable(@ColumnValues)
    --PRINT @NumOfRows: ' + CAST(@NumOfRows AS NVARCHAR)
    /* The function fn_getColumnValuesTable(@ColumnValues) should return two rows only */
```

```
    IF @NumOfRows != 10
```

```
    BEGIN
```

```
        SELECT @IDProcState = 1
        SELECT @Message = 'An error occured while inserting the table row. '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + ', '
        SELECT @Message = @Message + 'Error occured in Procedure Proc_InsertRow10. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN
    END
```

```
END
```

Appendix 5: Product Class Database (PCD) System Code

/*Create a temporary table and insert the values from the table returned
by the function fn_getColumnValuesTable */

```
CREATE TABLE #TempTable (ColumnID INT, ColumnValue NVARCHAR(255) )
INSERT INTO #TempTable
SELECT ColumnID,ColVal FROM dbo.fn_getColumnValuesTable(@ColumnValues)
```

```
SELECT @Column1Val = ColumnValue FROM #TempTable WHERE ColumnID = 1
SELECT @Column2Val = ColumnValue FROM #TempTable WHERE ColumnID = 2
SELECT @Column3Val = ColumnValue FROM #TempTable WHERE ColumnID = 3
SELECT @Column4Val = ColumnValue FROM #TempTable WHERE ColumnID = 4
SELECT @Column5Val = ColumnValue FROM #TempTable WHERE ColumnID = 5
SELECT @Column6Val = ColumnValue FROM #TempTable WHERE ColumnID = 6
SELECT @Column7Val = ColumnValue FROM #TempTable WHERE ColumnID = 7
SELECT @Column8Val = ColumnValue FROM #TempTable WHERE ColumnID = 8
SELECT @Column9Val = ColumnValue FROM #TempTable WHERE ColumnID = 9
SELECT @Column10Val = ColumnValue FROM #TempTable WHERE ColumnID = 10
```

/*Check whether the row contains column values or column specifications
A row can only contain column values or column specifications(headers).*/

```
IF @RowContent LIKE 'ColVals'
BEGIN
```

```
    /* Check for the existing rows with same IDTableVerDef to keep track of the number
    of rows the IDTableVerDef has if the @RowContent = 'ColumnValues' */
    SELECT @IDRow = MAX(IDRow) FROM TableDefinition10 WHERE IDTableVerDef = @IDTableVerDef
    IF @IDRow IS NULL
        SELECT @IDRow = 1
    ELSE
        SELECT @IDRow = @IDRow + 1
        INSERT INTO TableDefinition10(IDTableVerDef, IDRow, Column1Val,
        Column2Val,Column3Val,Column4Val,
        Column5Val, Column6Val,Column7Val, Column8Val, Column9Val,Column10Val, RowContent)
        VALUES(@IDTableVerDef, @IDRow,
        LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),
        LTRIM(RTRIM(@Column3Val)),
        LTRIM(RTRIM(@Column4Val)),LTRIM(RTRIM(@Column5Val)),
        LTRIM(RTRIM(@Column6Val)),
        LTRIM(RTRIM(@Column7Val)),LTRIM(RTRIM(@Column8Val)),
        LTRIM(RTRIM(@Column9Val)),
        LTRIM(RTRIM(@Column10Val)),LTRIM(RTRIM(@RowContent)))
        SELECT @Error = @@ERROR
        IF @Error != 0
        BEGIN
            SELECT @IDProcState = 2
            SELECT @Message = 'An error occured while inserting the table row.'
            SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '
            '
            SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
            NVARCHAR) + ' '
            SELECT @Message = @Message + 'Error occured in Procedure Proc_InsertRow10.'
            SELECT @Message = @Message + 'Procedure is terminated abnormally.'
            RAISERROR(@Message,1,1) WITH SETERROR
            RETURN @@ERROR
        END
        END
        ELSE
        BEGIN
            DROP TABLE #TempTable
            RETURN
        END
        END
        END --End of IF @RowContent LIKE 'TableRow'
    ELSE
        BEGIN
            SELECT @IDRow = 0
            INSERT INTO TableDefinition10(IDTableVerDef, IDRow, Column1Val, Column2Val, Column3Val,
            Column4Val,
            Column5Val, Column6Val,Column7Val, Column8Val,
            Column9Val, Column10Val,RowContent)
```

Appendix 5: Product Class Database (PCD) System Code

```
VALUES(@IDTableVerDef, @IDRow, LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),
LTRIM(RTRIM(@Column3Val)),LTRIM(RTRIM(@Column4Val)),LTRIM(RTRIM(@Column5Val)),LTRIM(R
TRIM(@Column6Val)),LTRIM(RTRIM(@Column7Val)),LTRIM(RTRIM(@Column8Val)),
LTRIM(RTRIM(@Column9Val)),LTRIM(RTRIM(@Column10Val)),
LTRIM(RTRIM(@RowContent)) )
SELECT @Error = @@ERROR
IF @Error != 0
BEGIN
    SELECT @IDProcState = 3
    SELECT @Message = 'An error occured while inserting the table row. '
    SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
    SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
    NVARCHAR) + ' '
    SELECT @Message = @Message + 'Error occured in Procedure Proc_InsertRow10. '
    SELECT @Message = @Message + 'Procedure is terminated abnormally. '
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END
ELSE
BEGIN
    DROP TABLE #TempTable
    RETURN
END
END
END -- End of Proc_InsertRow10 Procedure.
GO
```

/* 1.40 Procedure Name: dbo.proc_MeasurementUnit_sel

Database: PCD

Description:

This procedure enables selection of measurement unit.

*/

```
CREATE PROCEDURE proc_MeasurementUnit_sel
@IDMeasUnit          BIGINT = NULL,
@OrderBy            NVARCHAR(255) = NULL,
@UpDown            NVARCHAR(255) = 'ASC',
@RecordCount       INT OUTPUT
AS
DECLARE @SQL        NVARCHAR(4000)
SET NoCount ON

/*
if @IDMeasUnit is null then select all the measurement units.
*/

IF (@IDMeasUnit IS NULL)
BEGIN
    SET @SQL = ' SELECT IDMeasUnit, MeasUnitName, MeasUnitDesc
    FROM dbo.MeasurementUnit'

END
ELSE
BEGIN
    SET @SQL = ' SELECT IDMeasUnit, MeasUnitName, MeasUnitDesc
    FROM dbo.MeasurementUnit WHERE IDMeasUnit = @IDMeasUnit '

END

IF (@OrderBy IS NOT NULL)
BEGIN
    SET @SQL = @SQL + ' ORDER BY ' + @OrderBy + ' ' + @UpDown
END
EXEC sp_executesql @SQL, N'@IDMeasUnit BIGINT',@IDMeasUnit
SELECT @RecordCount = @@rowcount
GO
```

Appendix 5: Product Class Database (PCD) System Code

/*
1.41 Procedure Name: dbo.proc_ProductClass_SubProductClass_sel

Database: PCD

Description:

The following procedure selects all the product classes assigned to a product class.

*/

```
CREATE PROCEDURE proc_ProductClass_SubProductClass_sel
@IDProdClassDef          BIGINT = -1,
@OrderBy                 NVARCHAR(255) = NULL,
@UpDown                 NVARCHAR(255) = 'ASC',
@RecordCount            INT OUTPUT
AS
DECLARE @SQL             NVARCHAR(4000)
SET NoCount ON

/*
IF @IDProdClassDef = -1, all product classes and sub product classes assigned to them are selected.
*/

IF (@IDProdClassDef = -1)
BEGIN
    SET @SQL = 'SELECT pc.IDProdClass, pc.ProdClassName, pcv.IDProdClassDef,
pc1.IDProdClass AS IDSubProdClass, pc1.ProdClassName AS SubProdClassName, pcv1.IDProdClassDef AS
IDSubProdClassDef, pcv1.IDProdClassVer AS SubProdClassVer
FROM ProductClass pc, ProductClassVersion pcv, ProductClass pc1, productClassVersion pcv1, ProductClassDefinition pcd
WHERE pcv.IDProdClass = pc.IDProdClass AND
pcv1.IDProdClass = pc1.IDProdClass AND
pcv.IDProdClassDef = pcd.IDProdClassDef AND
pcv1.IDProdClassDef = pcd.IDSubClassDef'
END
ELSE
BEGIN
    SET @SQL = 'SELECT pc.IDProdClass, pc.ProdClassName, pcv.IDProdClassDef,
pc1.IDProdClass AS IDSubProdClass, pc1.ProdClassName AS SubProdClassName, pcv1.IDProdClassDef AS
IDSubProdClassDef, pcv1.IDProdClassVer AS SubProdClassVer
FROM ProductClass pc, ProductClassVersion pcv, ProductClass pc1, productClassVersion pcv1, ProductClassDefinition pcd
WHERE pcv.IDProdClass = pc.IDProdClass AND
pcv1.IDProdClass = pc1.IDProdClass AND
pcv.IDProdClassDef = pcd.IDProdClassDef AND
pcv1.IDProdClassDef = pcd.IDSubClassDef AND
pcv.IDProdClassDef = @IDProdClassDef'
END

IF (@OrderBy IS NOT NULL)
BEGIN
    SET @SQL = @SQL + ' ORDER BY ' + @OrderBy + ' ' + @UpDown
END
EXEC sp_executesql @SQL, N'@IDProdClassDef BIGINT',@IDProdClassDef
SELECT @RecordCount = @@rowcount
GO
```

/*
1.42 Procedure Name: dbo.proc_SpecGroup_ProductClass_sel

Database: PCD

Description:

The following procedure selects all the product classes assigned to a Specification Group.

*/

```
CREATE PROCEDURE proc_SpecGroup_ProductClass_sel
@IDSpecGroupDef          BIGINT,
@OrderBy                 NVARCHAR(255) = NULL,
@UpDown                 NVARCHAR(255) = 'ASC',
@RecordCount            INT OUTPUT
AS
DECLARE @SQL             NVARCHAR(4000)
SET NoCount ON
```

Appendix 5: Product Class Database (PCD) System Code

```
/*
IF @IDSpecGroupDef = -1, all the top level product classes assigned to all the specification groups are selected.
*/
IF (@IDSpecGroupDef = -1)
BEGIN
    SET @SQL = ' SELECT sgv.IDSpecGroupDef,
sgd.IDProdClassDef,
pcv.IDProdClass,
pcv.IDProdClassVer,
pcv.ProdClassVerDesc,
pc.ProdClassName,
pc.ProdClassDesc
FROM dbo.SpecificationGroupVersion sgv, dbo.SpecificationGroupDefinition sgd, dbo.ProductClassVersion pcv,
dbo.ProductClass pc
WHERE sgv.IDSpecGroupDef = sgd.IDSpecGroupDef AND
sgd.IDProdClassDef = pcv.IDProdClassDef AND
pcv.IDProdClass = pc.IDProdClass '

END
ELSE
/*
The following sql statement selects all the product classes assigned to a specification group.
*/
BEGIN
    SET @SQL = 'SELECT sgv.IDSpecGroupDef,
sgd.IDProdClassDef,
pcv.IDProdClass,
pcv.IDProdClassVer,
pcv.ProdClassVerDesc,
pc.ProdClassName,
pc.ProdClassDesc
FROM dbo.SpecificationGroupVersion sgv, dbo.SpecificationGroupDefinition sgd, dbo.ProductClassVersion pcv,
dbo.ProductClass pc
WHERE sgv.IDSpecGroupDef = sgd.IDSpecGroupDef AND
sgd.IDProdClassDef = pcv.IDProdClassDef AND
pcv.IDProdClass = pc.IDProdClass AND
sgv.IDSpecGroupDef = @IDSpecGroupDef '

END
IF (@OrderBy IS NOT NULL)
BEGIN
    SET @SQL = @SQL + ' ORDER BY ' + @OrderBy + ' ' + @UpDown
END
EXEC sp_executesql @SQL, N'@IDSpecGroupDef BIGINT', @IDSpecGroupDef
SELECT @RecordCount = @@rowcount
GO
```

1.43 Procedure Name: `dbo.proc_SpecGroup_SubSpecGroup_sel`

Database: PCD

Description:

This procedure selects sub specification groups for a given specification group.

```
CREATE PROCEDURE proc_SpecGroup_SubSpecGroup_sel
@IDSpecGroupDef BIGINT = NULL,
@OrderBy NVARCHAR(255) = NULL,
@UpDown NVARCHAR(255) = 'ASC',
@RecordCount INT OUTPUT
AS
DECLARE @SQL NVARCHAR(4000)
declare @IDSubSpecGroup bigint
declare @SubSpecGroupName nvarchar(255)
declare @IDSubSpecGroupDef bigint
declare @IDSubSpecGroupVer money

select @IDSubSpecGroup = null
select @SubSpecGroupName = null
```

Appendix 5: Product Class Database (PCD) System Code

```
select @IDSubSpecGroupDef = null
select @IDSubSpecGroupVer = null
SET NoCount ON
--The following if block selects top level spec groups having no parent spec groups.
--Top level spec group def are those whose IDs are not listed in IDSubSpecGroupDef column of SpecificationGroupDefinition Table.
IF (@IDSpecGroupDef IS NULL)
BEGIN
    SET @SQL = ' SELECT
    sg.IDSpecGroup,
    sg.SpecGroupName,
    --sg.SpecGroupDesc,
    sg.IDSpecGroupDef,
    sgv.IDSpecGroupVer
    --sgv.SpecGroupVerDesc
    FROM SpecificationGroup sg, SpecificationGroupVersion sgv
    WHERE sg.IDSpecGroup = sgv.IDSpecGroup
    AND sgv.IDSpecGroupDef NOT IN
    (SELECT IDSubSpecGroupDef FROM SpecificationGroupDefinition WHERE IDSubSpecGroupDef IS NOT NULL)'
    IF (@OrderBy IS NOT NULL)
    BEGIN
        SET @SQL = @SQL + ' ORDER BY ' + @OrderBy + ' + @UpDown
    END
    EXEC sp_executesql @SQL, N"
    SELECT @RecordCount = @@rowcount
    RETURN
END

END

--The Following if block selects all the top level specification groups and its sub-spec groups. In the rows selected
-- the top level specification groups are those having NULL value in the last four columns. This is a technique employed
-- to distinguish top level spec groups from rest others.

IF(@IDSpecGroupDef = -1)
BEGIN
    SET @SQL = 'SELECT  sg.IDSpecGroup, sg.SpecGroupName, sgv.IDSpecGroupDef, sgv.IDSpecGroupVer,
    sg1.IDSpecGroup AS IDSubSpecGroup, sg1.SpecGroupName AS SubSpecGroupName, sgv1.IDSpecGroupDef AS
    IDSubSpecGroupDef, sgv1.IDSpecGroupVer AS IDSubSpecGroupVer
    FROM SpecificationGroup sg, SpecificationGroupVersion sgv, SpecificationGroup sg1, SpecificationGroupVersion sgv1,
    SpecificationGroupDefinition sgd
    WHERE sg.IDSpecGroup = sgv.IDSpecGroup AND
    sg1.IDSpecGroup = sgv1.IDSpecGroup AND
    sgv.IDSpecGroupDef = sgd.IDSpecGroupDef AND
    sgv1.IDSpecGroupDef = sgd.IDSubSpecGroupDef AND sgv.IDSpecGroupDef IN
    (SELECT IDSpecGroupDef FROM SpecificationGroupDefinition)'
END
ELSE
BEGIN
    /*
    The following else block selects the sub specification groups for a given specification group.
    */
    SET @SQL = ' SELECT  sg.IDSpecGroup, sg.SpecGroupName, sgv.IDSpecGroupDef, sgv.IDSpecGroupVer,
    sg1.IDSpecGroup AS IDSubSpecGroup, sg1.SpecGroupName AS SubSpecGroupName, sgv1.IDSpecGroupDef AS
    IDSubSpecGroupDef, sgv1.IDSpecGroupVer AS IDSubSpecGroupVer
    FROM SpecificationGroup sg, SpecificationGroupVersion sgv, SpecificationGroup sg1, SpecificationGroupVersion sgv1,
    SpecificationGroupDefinition sgd
    WHERE sg.IDSpecGroup = sgv.IDSpecGroup AND
    sg1.IDSpecGroup = sgv1.IDSpecGroup AND
    sgv.IDSpecGroupDef = sgd.IDSpecGroupDef AND
    sgv1.IDSpecGroupDef = sgd.IDSubSpecGroupDef AND
    sgv.IDSpecGroupDef = @IDSpecGroupDef AND
    sgv.IDSpecGroupDef IN
    (SELECT IDSpecGroupDef FROM SpecificationGroupDefinition)'
END
IF (@OrderBy IS NOT NULL)
BEGIN
    SET @SQL = @SQL + ' ORDER BY ' + @OrderBy + ' + @UpDown
END
EXEC sp_executesql @SQL, N'@IDSpecGroupDef BIGINT, @IDSubSpecGroup BIGINT, @SubSpecGroupName nvarchar(255),
@IDSubSpecGroupDef bigint, @IDSubSpecGroupVer money',
```

Appendix 5: Product Class Database (PCD) System Code

```
@IDSpecGroupDef, @IDSubSpecGroup, @SubSpecGroupName, @IDSubSpecGroupDef, @IDSubSpecGroupVer
SELECT @RecordCount = @@rowcount
GO
```

```
/*
```

1.44 Procedure Name: dbo.proc_SpecName_check

Database: PCD

Description:

The following procedure checks whether the supplier SpecName exists in the SpecificationTable.

```
*/
```

```
CREATE PROCEDURE proc_SpecName_check
@SpecName          NVARCHAR(255),
@RecordCount       INT OUTPUT
AS
DECLARE @SQL        NVARCHAR(4000)
SET NoCount ON
BEGIN
    SET @SQL = ' SELECT LOWER(SpecName) FROM dbo.Specification
                WHERE SpecName = LOWER(@SpecName)'
END
EXEC sp_executesql @SQL, N'@SpecName NVARCHAR(255)',@SpecName
SELECT @RecordCount = @@rowcount
GO
```

```
/*
```

1.45 Procedure Name: dbo.proc_VersionRequest

Database: PCD

Description:

This procedure computes the latest version of a product class.

```
*/
```

```
CREATE Procedure proc_VersionRequest
/* Param List */
@IDProductClass      BIGINT,
@IDExistingLatestVersion  MONEY OUTPUT,
@IDPossibleNewVersion  MONEY OUTPUT,
@IDProcState         TINYINT OUTPUT,
@Message             NVARCHAR(500) OUTPUT
AS
BEGIN
    SELECT @IDProcState = 0
    SELECT @Message = 'Procedure Successfully Executed.'
    --BEGIN TRAN
    /*Check whether the supplied product class exists. If yes, then get the
    existing latest version of that product class in the database
    and compute the possible new version from it. */
    IF EXISTS(SELECT IDProductClass FROM ProductClass WHERE IDProductClass = @IDProductClass)
    BEGIN
        SELECT @IDExistingLatestVersion = MAX(IDProductClassVersion) FROM ProductClass
        WHERE IDProductClass = @IDProductClass
        SELECT @IDPossibleNewVersion = FLOOR(@IDExistingLatestVersion) + 1
    END
    ELSE
    BEGIN
        SELECT @IDProcState = 1
        SELECT @Message = 'Supplied IDProductClass: ' + CAST(@IDProductClass AS NVARCHAR) + ' does not
        exists. '
        SELECT @Message = @Message + 'Error occured in procedure Proc_CreateNewProductClassVersion. '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Procedure is abnormally terminated.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
END -- End of proc_VersionRequest.
GO
```

Appendix 5: Product Class Database (PCD) System Code

```
/*
1.46 Procedure Name: dbo.ProductClass_SubProductClass_sel
```

Database: PCD

Description:

The following procedure selects all the product classes assigned to a product class.

```
*/
```

```
CREATE PROCEDURE ProductClass_SubProductClass_sel
  @IDProdClassDef          BIGINT = -1,
  @OrderBy                 NVARCHAR(255) = NULL,
  @UpDown                 NVARCHAR(255) = 'ASC',
  @RecordCount            INT OUTPUT
AS
DECLARE @SQL              NVARCHAR(4000)
SET NoCount ON

/*
IF @IDProdClassDef = -1, all product classes and sub product classes assigned to them are selected.
*/

IF (@IDProdClassDef = -1)
BEGIN
  SET @SQL = 'SELECT pc.IDProdClass, pc.ProdClassName, pcv.IDProdClassDef,
pc1.IDProdClass AS IDSubProdClass, pc1.ProdClassName AS IDSubProdClassName, pcv1.IDProdClassDef AS
IDSubProdClassDef
FROM ProductClass pc, ProductClassVersion pcv, ProductClass pc1, productClassVersion pcv1, ProductClassDefinition pcd
WHERE pcv.IDProdClass = pc.IDProdClass AND
pcv1.IDProdClass = pc1.IDProdClass AND
pcv.IDProdClassDef = pcd.IDProdClassDef AND
pcv1.IDProdClassDef = pcd.IDSubClassDef'
END
ELSE
BEGIN
  SET @SQL = 'SELECT pc.IDProdClass, pc.ProdClassName, pcv.IDProdClassDef,
pc1.IDProdClass AS IDSubProdClass, pc1.ProdClassName AS IDSubProdClassName, pcv1.IDProdClassDef AS
IDSubProdClassDef
FROM ProductClass pc, ProductClassVersion pcv, ProductClass pc1, productClassVersion pcv1, ProductClassDefinition pcd
WHERE pcv.IDProdClass = pc.IDProdClass AND
pcv1.IDProdClass = pc1.IDProdClass AND
pcv.IDProdClassDef = pcd.IDProdClassDef AND
pcv1.IDProdClassDef = pcd.IDSubClassDef AND
pcv.IDProdClassDef = @IDProdClassDef'
END

IF (@OrderBy IS NOT NULL)
BEGIN
  SET @SQL = @SQL + ' ORDER BY ' + @OrderBy + ' ' + @UpDown
END
EXEC sp_executesql @SQL, N'@IDProdClassDef BIGINT',@IDProdClassDef
SELECT @RecordCount = @@rowcount
GO
```

```
/*
1.47 Procedure Name: dbo.proc_CreateColumnSpecification
```

Database: PCD

Description:

This procedure enables creation of column specification.

```
*/
```

```
CREATE Procedure proc_CreateColumnSpecification
  @IDTableDefinition BIGINT,
  @ColumnSpecs NVARCHAR(4000)
  /* Param List */
AS
BEGIN
  DECLARE
    @ColValue NVARCHAR(500),
```

Appendix 5: Product Class Database (PCD) System Code

```
@ColNumber INT
SELECT @ColNumber = NumberOfColumns FROM TableSpecification
WHERE IDTableDefinition = @IDTableDefinition
--SELECT @ColNumber = @ColNumber + 1
Declare TableValuesCursor Cursor FOR
select colval from fn_getColumnValuesTable(@ColumnSpecs)
open TableValuesCursor
FETCH NEXT FROM TableValuesCursor INTO @ColValue
WHILE @@FETCH_STATUS = 0
    BEGIN
        --PRINT @ColValue
        SELECT @ColNumber = @ColNumber + 1
        INSERT INTO TableDefinition (IDTableDefinition,TableRow,TableColumn,TableValue)
        VALUES (@IDTableDefinition, 0, @ColNumber, @ColValue)
        FETCH NEXT FROM TableValuesCursor INTO @ColValue
    END
close TableValuesCursor
Deallocate TableValuesCursor
UPDATE TableSpecification SET NumberOfColumns = @ColNumber
WHERE IDTableDefinition = @IDTableDefinition

END
GO

/*
1.48 Procedure Name: dbo.proc_CreateTableRow
Database: PCD
Description:
This procedure enables the creation of table row for table specification.
*/
CREATE Procedure proc_CreateTableRow
@IDTableDefinition BIGINT,
@ColumnValues NVARCHAR(4000)
/* Param List */
AS
BEGIN
    DECLARE
    @NumberOfColumns INT,
    @NumberOfRows INT,
    @ThisRow INT,
    @NumberOfColValsCreated INT,
    @ErrorMessage NVARCHAR(255),
    @ColValue NVARCHAR(500)
    SELECT @NumberOfColumns = NumberOfColumns FROM TableSpecification
    WHERE IDTableDefinition = @IDTableDefinition
    IF @NumberOfColumns = 0
    BEGIN
        SELECT @ErrorMessage = 'No Column Specifications found. Procedure Terminated!'
        RAISERROR(@ErrorMessage,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    SELECT @NumberOfColValsCreated = 0
    SELECT @NumberOfRows = NumberOfRows FROM TableSpecification
    WHERE IDTableDefinition = @IDTableDefinition
    SELECT @ThisRow = @NumberOfRows + 1
    Declare TableValuesCursor Cursor FOR
    select colval from fn_getColumnValuesTable(@ColumnValues)
    open TableValuesCursor
    FETCH NEXT FROM TableValuesCursor INTO @ColValue
    WHILE @@FETCH_STATUS = 0
    BEGIN
        Print '@NumOfColValsCreate Val: ' + CAST(@NumberOfColValsCreated AS NVARCHAR)
        Print '@NumberOfColumns Val: ' + CAST(@NumberOfColumns AS NVARCHAR)
        IF @NumberOfColValsCreated < @NumberOfColumns
        BEGIN
            INSERT INTO TableDefinition (IDTableDefinition,TableRow,TableColumn,TableValue)
            VALUES (@IDTableDefinition, @ThisRow, @NumberOfColValsCreated + 1, @ColValue)
            SELECT @NumberOfColValsCreated = @NumberOfColValsCreated + 1
        END
    END
```

Appendix 5: Product Class Database (PCD) System Code

```
    FETCH NEXT FROM TableValuesCursor INTO @ColValue
END
close TableValuesCursor
Deallocate TableValuesCursor
UPDATE TableSpecification SET NumberOfRows = NumberOfRows + 1
WHERE IDTableDefinition = @IDTableDefinition
END
GO
```

```
/*
1.49 Procedure Name: dbo.sp_DisplayTable
```

Database: PCD

Description:

This procedure displays table specification data.

```
*/
```

```
CREATE Procedure sp_DisplayTable
@IDTableDefinition BIGINT
/* Param List */
AS
BEGIN
    DECLARE
        @TableRowNumber INT,
        @TableColumnNumber INT,
        @TableColumnValues NVARCHAR(500)
    SELECT @TableColumnValues = "
    DECLARE TableRowsCursor CURSOR FOR
    SELECT DISTINCT (TableRow) FROM TableDefinition WHERE IDTableDefinition = @IDTableDefinition
    open TableRowsCursor
    FETCH NEXT FROM TableRowsCursor INTO @TableRowNumber
    WHILE @@FETCH_STATUS = 0
    BEGIN
        DECLARE TableColumnsCursor CURSOR FOR
        SELECT TableColumn FROM TableDefinition
        WHERE IDTableDefinition = @IDTableDefinition AND TableRow = @TableRowNumber
        OPEN TableColumnsCursor
        FETCH NEXT FROM TableColumnsCursor INTO @TableColumnNumber
        WHILE @@FETCH_STATUS = 0
        BEGIN
            SELECT @TableColumnValues = @TableColumnValues + ' ' +
            TableValue FROM TableDefinition
            WHERE IDTableDefinition = @IDTableDefinition
            AND TableRow = @TableRowNumber
            AND TableColumn = @TableColumnNumber
            FETCH NEXT FROM TableColumnsCursor INTO @TableColumnNumber
        END
        CLOSE TableColumnsCursor
        DEALLOCATE TableColumnsCursor
        PRINT @TableColumnValues
        SELECT @TableColumnValues = "
        FETCH NEXT FROM TableRowsCursor INTO @TableRowNumber
    END
    END
    CLOSE TableRowsCursor
    DEALLOCATE TableRowsCursor
END
GO
```

```
/*
1.50 Function Name: dbo.fn_getColumnValuesTable
```

Database: PCD

Description:

This function extracts column values from a string Column values in a string are delimited by '*****' */

After extracting the column values the function inserts each individual column value into the table and returns the table to the called procedure */

Appendix 5: Product Class Database (PCD) System Code

```
CREATE FUNCTION dbo.fn_getColumnValuesTable
(@ColumnValues NVARCHAR(4000))
RETURNS @ColumnValuesTable TABLE
(
    ColumnID INT,
    ColVal NVARCHAR(500)
)
AS
BEGIN
    DECLARE
    @Index INT, /* Keeps the index of position from where the delimiter starts. ie the starting position of '*****' in
a string */
    @DONE TINYINT, /* Acts as a boolean variable. */
    @ColumnVal NVARCHAR(500), /* Holds each individual column value */
    @Counter INT

    /*The following is executed when the procedure is called with empty string as input parameter */
    SELECT @DONE = 0
    SELECT @ColumnValues = LTRIM(RTRIM(@ColumnValues))
    IF LEN(@ColumnValues) = 0
    BEGIN
        SELECT @DONE = 1
        RETURN
    END

    /* The following is executed when the string contains only one column values */
    SELECT @Index = CHARINDEX('###', @ColumnValues)
    IF @Index = 0
    BEGIN
        SELECT @ColumnVal = @ColumnValues
        SELECT @ColumnVal = LTRIM(RTRIM(@ColumnVal))
        SELECT @Counter = 1
        INSERT INTO @ColumnValuesTable (ColumnID,ColVal) VALUES (@Counter, @ColumnValues)
        SELECT @DONE = 1
        RETURN
    END

    /* The following loop is executed when there are more than one column values in the string */
    SELECT @Counter = 1
    WHILE @DONE = 0
    BEGIN
        SELECT @Index = CHARINDEX('###', @ColumnValues)
        SELECT @ColumnVal = LEFT(@ColumnValues, @Index - 1)
        SELECT @ColumnVal = LTRIM(RTRIM(@ColumnVal))
        IF LEN(@ColumnVal) > 0
            INSERT INTO @ColumnValuesTable (ColumnID,ColVal) VALUES (@Counter, @ColumnVal)
        ELSE
            INSERT INTO @ColumnValuesTable (ColumnID,ColVal) VALUES (@Counter, NULL)
        SELECT @Counter = @Counter + 1
        SELECT @ColumnValues = SUBSTRING(@ColumnValues, @Index + 3, LEN(@ColumnValues) - @Index + 2)
        SELECT @ColumnValues = LTRIM(RTRIM(@ColumnValues))
        SELECT @Index = CHARINDEX('###', @ColumnValues)
        IF @Index = 0
        BEGIN
            IF LEN(@ColumnValues) = 0
            BEGIN
                SELECT @DONE = 1
            END
        ELSE
        BEGIN
            SELECT @ColumnVal = @ColumnValues
            SELECT @ColumnVal = LTRIM(RTRIM(@ColumnVal))
            IF LEN(@ColumnVal) > 0
                INSERT INTO @ColumnValuesTable (ColumnID,ColVal) VALUES (@Counter, @ColumnVal)
            ELSE
                INSERT INTO @ColumnValuesTable (ColumnID,ColVal) VALUES (@Counter, NULL)
            SELECT @DONE = 1
        END
    END
END
```

Appendix 5: Product Class Database (PCD) System Code

```
END
END
RETURN
END
```

/*

1.51 Function Name: dbo.fn_GetIDEntityPart

Database: PCD

Description:

This function returns entity part from a complete ID.

*/

```
CREATE FUNCTION dbo.fn_GetIDEntityPart
(@IDComplete BIGINT)
RETURNS INT
AS
BEGIN
    RETURN CAST(SUBSTRING(CAST(@IDComplete AS NVARCHAR), 1, 3) AS INT)
END
```

/*

1.52 Function Name: dbo.fn_GetIDPart

Database: PCD

Description:

This function returns ID part from a complete ID.

*/

```
CREATE FUNCTION dbo.fn_GetIDPart
(@IDComplete BIGINT)
RETURNS BIGINT
AS
BEGIN
    DECLARE
        @IDCompleteLength TINYINT,
        @IDPart NVARCHAR(20)
    SELECT @IDCompleteLength = LEN(CAST(@IDComplete AS NVARCHAR))
    SELECT @IDPart = SUBSTRING(CAST(@IDComplete AS NVARCHAR), 4, @IDCompleteLength - 3)
    RETURN CAST(@IDPart AS BIGINT)
END
```

/*

1.53 Function Name: dbo.fn_GetNewID

Database: PCD

Description:

This function generates a new ID for given entity such as product class, specification, etc.

*/

```
CREATE FUNCTION dbo.fn_GetNewID
(@IDEntity INT)
RETURNS BIGINT
AS
BEGIN
    DECLARE
        @IDAvailable BIGINT,
        @IDNext BIGINT
    SELECT @IDAvailable = IDAvailable FROM Entity WHERE [IDEntity] = @IDEntity
    SELECT @IDNext = dbo.fn_IncrementID(@IDAvailable, DEFAULT)
    --EXECUTE sp_SetNewID(@IDEntity)
    RETURN @IDAvailable
END
```

/*

1.54 Function Name: dbo.fn_IncrementID

Database: PCD

Description:

Appendix 5: Product Class Database (PCD) System Code

This function increments ID.

```
*/
CREATE FUNCTION dbo.fn_IncrementID
(@IDComplete BIGINT,
 @IncrementBy INT = 1)
RETURNS BIGINT
AS
BEGIN
    DECLARE
        @IDPart        BIGINT,
        @EntityPart    BIGINT
    SELECT @IDPart = dbo.fn_GetIDPart(@IDComplete)
    SELECT @EntityPart = dbo.fn_GetIDEntityPart(@IDComplete)
    SELECT @IDPart = @IDPart + @IncrementBy
    RETURN CAST(CAST(@EntityPart AS NVARCHAR) + CAST(@IDPart AS NVARCHAR) AS BIGINT)
END
```

/*
1.55 Function Name: dbo.fn_GetNextAvailableID

Database: PCD

Description:

This function generates next available ID.

```
*/
CREATE FUNCTION dbo.fn_GetNextAvailableID
(@IDComplete BIGINT)
RETURNS BIGINT
AS
BEGIN
    DECLARE
        @IDPart        BIGINT,
        @EntityPart    BIGINT
    SELECT @IDPart = dbo.GetIDPart(@IDComplete)
    SELECT @EntityPart = dbo.GetIDEntityPart(@IDComplete)
    SELECT @IDPart = @IDPart + 1
    RETURN CAST(CAST(@EntityPart AS NVARCHAR) + CAST(@IDPart AS NVARCHAR) AS BIGINT)
END
```

Supplier's Product Class Database (SPCD) System Code

/*
2.1 Procedure Name: dbo.proc_AssignCategory

Database: SupplierPCD

Description:

This procedure enables assigning a category to super category and sub category.

*/

```

CREATE Procedure proc_AssignCategory
/* Param List */
@IDCategory          BIGINT,
@IDSuperCategory     BIGINT,
@IDProcState         TINYINT    OUTPUT,
@Message             NVARCHAR(500) OUTPUT
AS
BEGIN
    DECLARE
    @Error            INT
    SELECT @IDProcState = 0
    SELECT @ERROR = 0

    IF NOT EXISTS (SELECT IDCategory, IDSuperCategory FROM Category_SuperCategory
        WHERE IDCategory = @IDCategory AND IDSuperCategory = @IDSuperCategory)
    BEGIN
        INSERT INTO Category_SuperCategory (IDCategory, IDSuperCategory)
        VALUES (@IDCategory, @IDSuperCategory)
        SELECT @ERROR = @@ERROR
        IF @Error != 0
        BEGIN
            ROLLBACK TRAN
            SELECT @IDProcState = 1
            SELECT @Message = 'An error occured while creating the new Category. '
            SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
            SELECT @Message = @Message + 'Error occured in Procedure proc_CreateCategory. '
            SELECT @Message = @Message + 'Procedure is terminated abnormally. '
            RAISERROR(@Message,1,1) WITH SETERROR
            RETURN @@ERROR
        END
    END -- End of IF NOT EXISTS...

    IF NOT EXISTS (SELECT IDCategory, IDSubCategory FROM Category_SubCategory
        WHERE IDCategory = @IDSuperCategory AND IDSubCategory = @IDCategory)
    BEGIN
        INSERT INTO Category_SubCategory (IDCategory, IDSubCategory)
        VALUES (@IDSuperCategory, @IDCategory)
        SELECT @ERROR = @@ERROR
        IF @Error != 0
        BEGIN
            ROLLBACK TRAN
            SELECT @IDProcState = 2
            SELECT @Message = 'An error occured while creating the new category. '
            SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
            SELECT @Message = @Message + 'Error occured in Procedure proc_CreateCategory. '
            SELECT @Message = @Message + 'Procedure is terminated abnormally. '
            RAISERROR(@Message,1,1) WITH SETERROR
            RETURN @@ERROR
        END
    END
    ELSE
    BEGIN
        SELECT @Message = 'Category successfully assigned.'
        RETURN
    END
END -- End of IF NOT EXISTS.

END -- End of Proc_AssignCategory Procedure.
GO

```


Appendix 5: Supplier's Product Class Database (SPCD) System Code

2.2 Procedure Name: `dbo.proc_AssignListSpecification`

Database: `SupplierPCD`

Description:

This procedure enables assigning a list specification to product class or specification group.

```
*/
CREATE Procedure proc_AssignListSpecification
/* Param List */
@IDListDef          BIGINT,
@IDAssignToSpecTypeDef  BIGINT,
@IDProcState        TINYINT      OUTPUT,
@Message            NVARCHAR(500) OUTPUT
AS
BEGIN
    DECLARE
    @Error          INT,
    @IDEntityPart2  INT /*Holds Entity part of @IDAssignToSpecTypeDef*/

    SELECT @IDProcState = 0
    SELECT @ERROR = 0
    SELECT @IDEntityPart2 = dbo.fn_getIDEntityPart(@IDAssignToSpecTypeDef)

    /* A List Specification can be assigned to a product class or a specification group */

    IF @IDEntityPart2 NOT IN (105,106)
    BEGIN
        SELECT @IDProcState = 1
        SELECT @Message = 'A List Specification can only be assigned to a Product Class or a Specification Group.'

        SELECT @Message = @Message + 'The supplied specification type to which specification group needs to be
assigned to is invalid.'
        SELECT @Message = @Message + 'Error occured in Procedure proc_AssignListSpecification.'
        SELECT @Message = @Message + 'Procedure is terminated abnormally.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END

    /* Assigning List specification group to a product class*/
    /* Assign a list specification to a product class only if it is not previously assigned.
The following IF NOT EXISTS block checks this. */

    IF @IDEntityPart2 = 105
    BEGIN
        IF NOT EXISTS (SELECT IDProdClassDef, IDListDef FROM ProductClassDefinition
WHERE IDProdClassDef = @IDAssignToSpecTypeDef AND IDListDef = @IDListDef)
        BEGIN
            INSERT INTO ProductClassDefinition(IDProdClassDef, IDListDef)
VALUES (@IDAssignToSpecTypeDef, @IDListDef)
            SELECT @Error = @@ERROR
            IF @Error != 0
            BEGIN
                ROLLBACK TRAN
                SELECT @IDProcState = 2
                SELECT @Message = 'An error occured while assigning the List Specification to the product
class.'
                SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '.'
                SELECT @Message = @Message + 'Error occured in Procedure proc_AssignListSpecification.'
                SELECT @Message = @Message + 'Procedure is terminated abnormally.'
                RAISERROR(@Message,1,1) WITH SETERROR
                RETURN @@ERROR
            END
        ELSE
        BEGIN
            SELECT @Message = 'List Specification successfully assigned to the Product Class.'
            RETURN
        END
    END
    END -- End of IF NOT EXISTS...

    END -- End of IF @IDEntityPart2 = 105

    /* Assigning list specification to a specification Group */
    /* Assign a list specification to a specification group only if it is not previously assigned.
The following IF NOT EXISTS block checks this. */
```

Appendix 5: Supplier's Product Class Database (SPCD) System Code

```
IF @IDEntityPart2 = 106
BEGIN
    IF NOT EXISTS (SELECT IDSpecGroupDef, IDListDef FROM SpecificationGroupDefinition
        WHERE IDSpecGroupDef = @IDAssignToSpecTypeDef AND IDListDef = @IDListDef)
    BEGIN
        INSERT INTO SpecificationGroupDefinition(IDSpecGroupDef, IDListDef)
        VALUES (@IDAssignToSpecTypeDef, @IDListDef)
        SELECT @Error = @@ERROR
        IF @Error != 0
        BEGIN
            ROLLBACK TRAN
            SELECT @IDProcState = 3
            SELECT @Message = 'An error occurred while assigning the List Specification to the
                specification group.'
            SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS
                NVARCHAR) + ' '
            SELECT @Message = @Message + 'Error occurred in Procedure
                proc_AssignListSpecification. '
            SELECT @Message = @Message + 'Procedure is terminated abnormally. '
            RAISERROR(@Message,1,1) WITH SETERROR
            RETURN @@ERROR
        END
    END
    ELSE
    BEGIN
        SELECT @Message = 'List Specification successfully assigned to the Specification
            Group.'
        RETURN
    END
    END -- End of IF NOT EXISTS...
END --End of IF @IDEntityPart2 = 106 */
END -- End of proc_AssignListSpecification Procedure.
GO
```

2.3 Procedure Name: **dbo.proc_AssignProductClass**

Database: **SupplierPCD**

Description:

This procedure enables assigning a (sub) product class to category, product class or specification group.

*/

```
CREATE Procedure proc_AssignProductClass
/* Param List */
@IDProdClassDef          BIGINT,
@IDAssignToSpecTypeDef   BIGINT,
@IDProcState             TINYINT OUTPUT,
@Message                 NVARCHAR(500) OUTPUT
AS
BEGIN
    DECLARE
    @Error                INT,
    @IDEntityPart2       INT /*Holds Entity part of @IDAssignToSpecTypeDef*/

    SELECT @IDProcState = 0
    SELECT @ERROR = 0
    SELECT @IDEntityPart2 = dbo.fn_getIDEntityPart(@IDAssignToSpecTypeDef)
    /* A sub product class can be assigned to a category,product class or a specification group */

    IF @IDEntityPart2 NOT IN (102,105,106)
    BEGIN
        SELECT @IDProcState = 1
        SELECT @Message = 'A product class can only be assigned to a Category, Product Class or a Specification
            Group. '
        SELECT @Message = @Message + 'The supplied specification type to which Product Class needs to be
            assigned is invalid. '
        SELECT @Message = @Message + 'Error occurred in Procedure proc_AssignProductClass. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END

    /* Assigning product class to a product class*/
    /* Assign a sub product class to a product class only if it is not previously assigned.
```

Appendix 5: Supplier's Product Class Database (SPCD) System Code

The following IF NOT EXISTS block checks this. */

IF @IDEntityPart2 = 105

BEGIN

```
IF NOT EXISTS (SELECT IDProdClassDef, IDSubClassDef FROM ProductClassDefinition
WHERE IDProdClassDef = @IDAssignToSpecTypeDef AND IDSubClassDef = @IDProdClassDef)
BEGIN
```

```
    INSERT INTO ProductClassDefinition(IDProdClassDef, IDSubclassDef)
```

```
    VALUES (@IDAssignToSpecTypeDef, @IDProdClassDef)
```

```
    SELECT @Error = @@ERROR
```

```
    IF @Error != 0
```

```
    BEGIN
```

```
        ROLLBACK TRAN
```

```
        SELECT @IDProcState = 2
```

```
        SELECT @Message = 'An error ocured while assigning the Subproduct class to the
product class.'
```

```
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS
NVARCHAR) + '.'
```

```
        SELECT @Message = @Message + 'Error ocured in Procedure
proc_AssignProductClass.'
```

```
        SELECT @Message = @Message + 'Procedure is terminated abnormally.'
```

```
        RAISERROR(@Message,1,1) WITH SETERROR
```

```
        RETURN @@ERROR
```

```
    END
```

```
    ELSE
```

```
    BEGIN
```

```
        SELECT @Message = 'Subproduct class successfully assigned to the product class.'
```

```
        RETURN
```

```
    END
```

```
END -- End of IF NOT EXISTS...
```

```
END -- End of IF @IDEntityPart2 = 105
```

/* Assigning product class to a specification Group */

/* Assign a product class to a specification Group only if it is not previously assigned.

The following IF NOT EXISTS block checks this. */

IF @IDEntityPart2 = 106

BEGIN

```
IF NOT EXISTS (SELECT IDSpecGroupDef, IDProdClassDef FROM SpecificationGroupDefinition
WHERE IDSpecGroupDef = @IDAssignToSpecTypeDef AND IDProdClassDef = @IDProdClassDef)
BEGIN
```

```
    INSERT INTO SpecificationGroupDefinition(IDSpecGroupDef, IDProdClassDef)
```

```
    VALUES (@IDAssignToSpecTypeDef, @IDProdClassDef)
```

```
    SELECT @Error = @@ERROR
```

```
    IF @Error != 0
```

```
    BEGIN
```

```
        ROLLBACK TRAN
```

```
        SELECT @IDProcState = 3
```

```
        SELECT @Message = 'An error ocured while assigning the product class to the
specification group.'
```

```
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS
NVARCHAR) + '.'
```

```
        SELECT @Message = @Message + 'Error ocured in Procedure
proc_AssignProductClass.'
```

```
        SELECT @Message = @Message + 'Procedure is terminated abnormally.'
```

```
        RAISERROR(@Message,1,1) WITH SETERROR
```

```
        RETURN @@ERROR
```

```
    END
```

```
    ELSE
```

```
    BEGIN
```

```
        SELECT @Message = 'Product class successfully assigned to the specification group.'
```

```
        RETURN
```

```
    END
```

```
END -- End of IF NOT EXISTS
```

```
END --End of IF @IDEntityPart2 = 106 */
```

/* Assigning product class to a category */

/* Assigning product class to a specification Group */

/* Assign a product class to a category only if it is not previously assigned.

The following IF NOT EXISTS block checks this. */

IF @IDEntityPart2 = 102

BEGIN

```
IF NOT EXISTS (SELECT IDCcategory, IDProdClassDef FROM Category_ProductClass
WHERE IDCcategory = @IDAssignToSpecTypeDef AND IDProdClassDef = @IDProdClassDef)
```

```
BEGIN
```

```
    INSERT INTO Category_ProductClass(IDCcategory, IDProdClassDef)
```

Appendix 5: Supplier's Product Class Database (SPCD) System Code

```
VALUES (@IDAssignToSpecTypeDef, @IDProdClassDef)
SELECT @Error = @@ERROR
IF @Error != 0
BEGIN
    ROLLBACK TRAN
    SELECT @IDProcState = 4
    SELECT @Message = 'An error occurred while assigning the product class to the
category.'
    SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS
NVARCHAR) + '.'
    SELECT @Message = @Message + 'Error occurred in Procedure
proc_AssignProductClass.'
    SELECT @Message = @Message + 'Procedure is terminated abnormally.'
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END
ELSE
BEGIN
    SELECT @Message = 'Product class successfully assigned to the category.'
    RETURN
END
END --End of IF @IDEntityPart2 = 102 */
END -- End of IF NOT EXISTS.
-- End of Proc_AssignProductClass Procedure.
END
GO
```

/*

2.4 Procedure Name: **dbo.proc_AssignSpecification**

Database: **SupplierPCD**

Description:

This procedure enables assigning a specification to product class or specification group.

*/

```
CREATE Procedure proc_AssignSpecification
/* Param List */
@IDSpec                BIGINT,
@IDAssignToSpecTypeDef BIGINT,
@SpecValue             NVARCHAR(4000) = NULL,
@IDMeasUnit            BIGINT = NULL,
@IDProcState           TINYINT OUTPUT,
@Message               NVARCHAR(500) OUTPUT

AS
BEGIN
    DECLARE
    @Error                INT,
    @IDEntityPart2       INT /*Holds Entity part of @IDAssignToSpecTypeDef*/
    SELECT @IDProcState = 0
    SELECT @ERROR = 0
    SELECT @IDEntityPart2 = dbo.fn_getIDEntityPart(@IDAssignToSpecTypeDef)

    /*A specification can only be assigned to a product class or a specification group */
    IF @IDEntityPart2 NOT IN (105, 106)
    BEGIN
        SELECT @Message = 'A specification can only be assigned to a Product Class or a Specification Group.'
        SELECT @Message = @Message + 'The supplied specification type to which specification needs to be
assigned is invalid.'
        SELECT @Message = @Message + 'Error occurred in Procedure proc_AssignSpecification.'
        SELECT @Message = @Message + 'Procedure is terminated abnormally.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    IF @IDMeasUnit IS NOT NULL
    BEGIN
        IF @SpecValue IS NULL
        BEGIN
            ROLLBACK TRAN
            SELECT @IDProcState = 4
            SELECT @Message = 'An error occurred while creating the new specification.'
            SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
NVARCHAR) + '.'
            SELECT @Message = @Message + 'Error occurred in Procedure proc_CreateNewSpecification.'
            SELECT @Message = @Message + 'Procedure is terminated abnormally.'
            RAISERROR(@Message,1,1) WITH SETERROR
```

Appendix 5: Supplier's Product Class Database (SPCD) System Code

```
        RETURN @@ERROR
    END
END
/* Assign Specification to a product class */
/* Assign Specification to a product class only if it is not previously assigned.
   The following IF NOT EXISTS block checks this. */
IF @IDEntityPart2 = 105
BEGIN
    IF NOT EXISTS (SELECT IDProdClassDef, IDSpec FROM PCDSpecificationValue WHERE
        IDProdClassDef=@IDAssignToSpecTypeDef AND IDSpec = @IDSpec)
    BEGIN
        INSERT INTO PCDSpecificationValue(IDProdClassDef, IDSpec,SpecValue, IDMeasUnit)
        VALUES(@IDAssignToSpecTypeDef, @IDSpec, @SpecValue, @IDMeasUnit)
        SELECT @Error = @@ERROR
        IF @Error != 0
        BEGIN
            ROLLBACK TRAN
            SELECT @IDProcState = 1
            SELECT @Message = 'An error occured while assigning the specification to the product
                class.'
            SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState
                AS NVARCHAR) + ' '
            SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS
                NVARCHAR) + ' '
            SELECT @Message = @Message + 'Error occured in Procedure
                proc_AssignSpecification. '
            SELECT @Message = @Message + 'Procedure is terminated abnormally. '
            RAISERROR(@Message,1,1) WITH SETERROR
            RETURN @@ERROR
        END
        ELSE
        BEGIN
            SELECT @Message = 'Specification successfully assigned to the product class.'
            RETURN
        END
    END
    END -- End of IF NOT EXISTS...
END -- End of IF @IDEntityPart2 = 105

/* Assign Specification to a specification Group */
/* Assign Specification to Specification only if it is not previously assigned.
   The following IF NOT EXISTS block checks this. */
IF @IDEntityPart2 = 106
BEGIN
    IF NOT EXISTS (SELECT IDSpecGroupDef, IDSpec FROM SGDSpecificationValue WHERE
        IDSpecGroupDef=@IDAssignToSpecTypeDef AND IDSpec = @IDSpec)
    BEGIN
        INSERT INTO SGDSpecificationValue (IDSpecGroupDef, IDSpec,SpecValue, IDMeasUnit)
        VALUES(@IDAssignToSpecTypeDef, @IDSpec, @SpecValue, @IDMeasUnit)
        SELECT @Error = @@ERROR
        IF @Error != 0
        BEGIN
            ROLLBACK TRAN
            SELECT @IDProcState = 2
            SELECT @Message = 'An error occured while assigning the specification to the
                specification group.'
            SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS
                NVARCHAR) + ' '
            SELECT @Message = @Message + 'Error occured in Procedure
                proc_AssignSpecification. '
            SELECT @Message = @Message + 'Procedure is terminated abnormally. '
            RAISERROR(@Message,1,1) WITH SETERROR
            RETURN @@ERROR
        END
        ELSE
        BEGIN
            SELECT @Message = 'Specification successfully assigned to the specification group.'
            RETURN
        END
    END
    END -- End of IF NOT EXISTS...
END -- End of IF @IDEntityPart2 = 106
-- End of Proc_AssignSpecification Procedure.
END
GO
/*
```

Appendix 5: Supplier's Product Class Database (SPCD) System Code

2.5 Procedure Name: dbo.proc_AssignSpecificationGroup

Database: SupplierPCD

Description:

This procedure enables assigning a (sub) specification group to specification group or product class.

*/

```
CREATE Procedure proc_AssignSpecificationGroup
/* Param List */
@IDSpecGroupDef          BIGINT,
@IDAssignToSpecTypeDef   BIGINT,
@IDProcState             TINYINT   OUTPUT,
@Message                 NVARCHAR(500) OUTPUT

AS
BEGIN
    DECLARE
        @Error          INT,
        @IDEntityPart2 INT /*Holds Entity part of @IDAssignToSpecTypeDef*/

    SELECT @IDProcState = 0
    SELECT @ERROR = 0
    SELECT @IDEntityPart2 = dbo.fn_getIDEntityPart(@IDAssignToSpecTypeDef)
    /* A specification group can be assigned to a product class or a specification group */

    IF @IDEntityPart2 NOT IN (105,106)
    BEGIN
        SELECT @IDProcState = 1
        SELECT @Message = 'A specification group can only be assigned to a Product Class or a Specification
        Group.'
        SELECT @Message = @Message + 'The supplied specification type to which specification group needs to be
        assigned to is invalid.'
        SELECT @Message = @Message + 'Error occured in Procedure proc_AssignSpecificationGroup.'
        SELECT @Message = @Message + 'Procedure is terminated abnormally.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END

    /* Assigning specification group to a product class*/
    /* Assign specification group to a product class only if it is not previously assigned.
    The following IF NOT EXISTS block checks this. */
    IF @IDEntityPart2 = 105
    BEGIN
        IF NOT EXISTS (SELECT IDProdClassDef, IDSpecGroupDef FROM ProductClassDefinition
            WHERE IDProdClassDef = @IDAssignToSpecTypeDef AND IDSpecGroupDef =
            @IDSpecGroupDef)
        BEGIN
            INSERT INTO ProductClassDefinition(IDProdClassDef, IDSpecGroupDef)
            VALUES (@IDAssignToSpecTypeDef, @IDSpecGroupDef)
            SELECT @Error = @@ERROR
            IF @Error != 0
            BEGIN
                ROLLBACK TRAN
                SELECT @IDProcState = 2
                SELECT @Message = 'An error occured while assigning the specification group to the
                product class.'
                SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS
                NVARCHAR) + '.'
                SELECT @Message = @Message + 'Error occured in Procedure
                proc_AssignSpecificationGroup.'
                SELECT @Message = @Message + 'Procedure is terminated abnormally.'
                RAISERROR(@Message,1,1) WITH SETERROR
                RETURN @@ERROR
            END
        END
        ELSE
        BEGIN
            SELECT @Message = 'Specification group successfully assigned to the Product Class.'
            RETURN
        END
    END -- End of IF NOT EXISTS...
END -- End of IF @IDEntityPart2 = 105

/* Assigning sub specification group to a specification Group */
/* Assign sub specification group to a specification group only if it is not previously assigned.
The following IF NOT EXISTS block checks this. */
```

Appendix 5: Supplier's Product Class Database (SPCD) System Code

```
IF @IDEntityPart2 = 106
BEGIN
IF NOT EXISTS (SELECT IDSpecGroupDef, IDSubSpecGroupDef FROM SpecificationGroupDefinition
WHERE IDSpecGroupDef = @IDAssignToSpecTypeDef AND IDSubSpecGroupDef = @IDSpecGroupDef)
BEGIN
INSERT INTO SpecificationGroupDefinition(IDSpecGroupDef, IDSubSpecGroupDef)
VALUES (@IDAssignToSpecTypeDef, @IDSpecGroupDef)
SELECT @Error = @@ERROR
IF @Error != 0
BEGIN
ROLLBACK TRAN
SELECT @IDProcState = 3
SELECT @Message = 'An error occurred while assigning the specification group to the
specification group.'
SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
SELECT @Message = @Message + 'Error occurred in Procedure proc_AssignSpecificationGroup.

SELECT @Message = @Message + 'Procedure is terminated abnormally. '
RAISERROR(@Message,1,1) WITH SETERROR
RETURN @@ERROR
END
ELSE
BEGIN
SELECT @Message = 'Specification group successfully assigned to the Specification
Group.'
RETURN
END
END -- End of IF NOT EXISTS...
END --End of IF @IDEntityPart2 = 106 */
END -- End of Proc_AssignSpecificationGroup Procedure.
GO
```

2.6 Procedure Name: **dbo.proc_AssignTableSpecification**

Database: **SupplierPCD**

Description:

This procedure enables assigning a table specification to product class or specification group.

*/

```
CREATE Procedure proc_AssignTableSpecification
/* Param List */
@IDTableVerDef          BIGINT,
@IDAssignToSpecTypeDef  BIGINT,
@IDProcState            TINYINT      OUTPUT,
@Message                NVARCHAR(500) OUTPUT

AS
BEGIN

DECLARE
@Error          INT,
@IDEntityPart2 INT /*Holds Entity part of @IDAssignToSpecTypeDef*/

SELECT @IDProcState = 0
SELECT @ERROR = 0
SELECT @IDEntityPart2 = dbo.fn_getIDEntityPart(@IDAssignToSpecTypeDef)

/* A Table specification can be assigned to a product class or a specification group */

IF @IDEntityPart2 NOT IN (105,106)
BEGIN
SELECT @IDProcState = 1
SELECT @Message = 'A Table Specification can only be assigned to a Product Class or a Specification
Group. '
SELECT @Message = @Message + 'The supplied specification type to which specification group needs to be
assigned to is invalid. '
SELECT @Message = @Message + 'Error occurred in Procedure proc_AssignTableSpecification. '
SELECT @Message = @Message + 'Procedure is terminated abnormally. '
RAISERROR(@Message,1,1) WITH SETERROR
RETURN @@ERROR
END

/* Assigning Table specification to a product class*/
```

Appendix 5: Supplier's Product Class Database (SPCD) System Code

```
/* Assign a table specification to a product class only if it is not previously assigned.
The following IF NOT EXISTS block checks this. */
IF @IDEntityPart2 = 105
BEGIN
    IF NOT EXISTS (SELECT IDProdClassDef, IDTableVerDef FROM ProductClassDefinition
    WHERE IDProdClassDef = @IDAssignToSpecTypeDef AND IDTableVerDef = @IDTableVerDef)
    BEGIN
        INSERT INTO ProductClassDefinition(IDProdClassDef, IDTableVerDef)
        VALUES (@IDAssignToSpecTypeDef, @IDTableVerDef)
        SELECT @Error = @@ERROR
        IF @Error != 0
        BEGIN
            ROLLBACK TRAN
            SELECT @IDProcState = 2
            SELECT @Message = 'An error occurred while assigning the table specification to the
            product class.'
            SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS
            NVARCHAR) + ', '
            SELECT @Message = @Message + 'Error occurred in Procedure
            proc_AssignTableSpecification. '
            SELECT @Message = @Message + 'Procedure is terminated abnormally. '
            RAISERROR(@Message,1,1) WITH SETERROR
            RETURN @@ERROR
        END
        ELSE
        BEGIN
            SELECT @Message = 'Table Specification successfully assigned to the Product Class.'
            RETURN
        END
    END -- End of IF NOT EXISTS...
END -- End of IF @IDEntityPart2 = 105

/* Assigning Table Specification to a specification Group */
/* Assign a table specification to a product class only if it is not previously assigned.
The following IF NOT EXISTS block checks this. */

IF @IDEntityPart2 = 106
BEGIN
    IF NOT EXISTS (SELECT IDSpecGroupDef, IDTableVerDef FROM SpecificationGroupDefinition
    WHERE IDSpecGroupDef = @IDAssignToSpecTypeDef AND IDTableVerDef = @IDTableVerDef)
    BEGIN
        INSERT INTO SpecificationGroupDefinition(IDSpecGroupDef, IDTableVerDef)
        VALUES (@IDAssignToSpecTypeDef, @IDTableVerDef)
        SELECT @Error = @@ERROR
        IF @Error != 0
        BEGIN
            ROLLBACK TRAN
            SELECT @IDProcState = 3
            SELECT @Message = 'An error occurred while assigning the table specification to the
            specification group.'
            SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS
            NVARCHAR) + ', '
            SELECT @Message = @Message + 'Error occurred in Procedure
            proc_AssignTableSpecification. '
            SELECT @Message = @Message + 'Procedure is terminated abnormally. '
            RAISERROR(@Message,1,1) WITH SETERROR
            RETURN @@ERROR
        END
        ELSE
        BEGIN
            SELECT @Message = 'Table Specification successfully assigned to the Specification
            Group.'
            RETURN
        END
    END -- End of IF NOT EXISTS...
END --End of IF @IDEntityPart2 = 106 */
-- End of proc_AssignTableSpecification Procedure.
END
GO

/*
```


Appendix 5: Supplier's Product Class Database (SPCD) System Code

2.7 Procedure Name: dbo.proc_CreateCategory

Database: SupplierPCD

Description:

This procedure enables creation of a category.

*/

```
CREATE Procedure proc_CreateCategory
/* Param List */
@IDCategory          BIGINT,
@CategoryName        NVARCHAR(255),
@IDSuperCategory     BIGINT = NULL,
@CategoryDesc        NVARCHAR(4000) = NULL,
@IDProcState         TINYINT OUTPUT,
@Message             NVARCHAR(500) OUTPUT

AS
BEGIN
    DECLARE
    @Error            INT
    SELECT @IDProcState = 0
    SELECT @ERROR = 0
    BEGIN TRAN

    /*Create the Category by inserting values into the Category table */
    /*
    If the Category already exists in the Supplier PCD Database then we do not need
    to perform the following insert operation.
    */
    IF NOT EXISTS (SELECT IDCategory FROM Category WHERE IDCategory = @IDCategory)
    BEGIN
        INSERT INTO Category(IDCategory, CategoryName, CategoryDesc)
        VALUES (@IDCategory, LTRIM(RTRIM(@CategoryName)), LTRIM(RTRIM(@CategoryDesc)))
        SELECT @Error = @@ERROR
        IF @Error != 0
        BEGIN
            ROLLBACK TRAN
            SELECT @IDProcState = 2
            SELECT @Message = 'An error occured while creating the new category.'
            SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
            SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
            NVARCHAR) + ' '
            SELECT @Message = @Message + 'Error occured in Procedure proc_CreateCategory. '
            SELECT @Message = @Message + 'Procedure is terminated abnormally. '
            RAISERROR(@Message,1,1) WITH SETERROR
            RETURN @@ERROR
        END
    END -- END of IF NOT EXISTS...
    /* Now insert values into the CategoryHierarchy table to maintain category hierarchy. */

    /*IF @SuperCategory is null then a category is top level category and the IDSuperCategory
    value for the category is 0. It also has no sub category. */
    IF @IDSuperCategory IS NULL
    BEGIN
        SELECT @Message = 'Category successfully Created. '
        SELECT @Message = @Message + 'Category ID is: ' + CAST(@IDCategory AS NVARCHAR) + ' '
        COMMIT TRAN
        RETURN
    END
    ELSE
    BEGIN
        /* If a category has a super category then the category is also a sub category
        of that super category. In this case two inserts are required. First to create a
        category and its super category and second to create a supercategory and its sub category
        For this we call proc_AssignCategory. */

        EXEC proc_AssignCategory
        @IDCategory          = @IDCategory,
        @IDSuperCategory     = @IDSuperCategory,
        @IDProcState         = @IDProcState OUTPUT,
        @Message             = @Message OUTPUT
        IF @IDProcState != 0
        BEGIN
            ROLLBACK TRAN
            SELECT @Message = @Message + ' This Procedure was called from Proc_AssignCategory.'
```

Appendix 5: Supplier's Product Class Database (SPCD) System Code

```
RAISERROR(@Message,1,1) WITH SETERROR
RETURN @@ERROR
```

```
END
ELSE
BEGIN
```

```
SELECT @Message = 'Category successfully Downloaded. '
SELECT @Message = @Message + 'Category ID is: ' + CAST(@IDCategory AS NVARCHAR)
+ '!'
COMMIT TRAN
RETURN
```

```
END
```

```
END
```

```
END -- End of Proc_CreateCategory Procedure.
GO
```

```
/*
```

2.8 Procedure Name: dbo.proc_CreateNewListSpecification

Database: SupplierPCD

Description:

This procedure enables creation of a new list specification.

```
*/
```

```
CREATE Procedure proc_CreateNewListSpecification
```

```
/* Param List */
```

```
@IDList          BIGINT,
@IDListVer       MONEY,
@IDListDef       BIGINT,
@ListName        NVARCHAR(255),
@ListDesc        NVARCHAR(4000) = NULL,
@ListVerDesc     NVARCHAR(4000) = NULL,
@ListValues      NVARCHAR(4000),
@ListIDMeasUnits NVARCHAR(4000) = NULL,
@IDAssignToSpecTypeDef BIGINT = NULL,
@IDProcState     TINYINT OUTPUT,
@Message         NVARCHAR(500) OUTPUT
```

```
AS
BEGIN
```

```
DECLARE
```

```
@Error          INT,
@NewListVersion NVARCHAR(5)
SELECT @IDProcState = 0
SELECT @ERROR = 0
SELECT @NewListVersion = 'No'
BEGIN TRAN
```

```
/*Create a List specification in the Supplier PCD database
by inserting values into ListSpecification, List version
tables. The order in which the values are inserted into the
table should be maintained. First values should be inserted into the ListSpecification table
then into ListVersion table because IDList in ListVersion table
references IDList in ListSpecification.*/
```

```
/*
```

```
If the list specification already exists in the Supplier PCD Database then we do not need
to perform the following insert operation.
```

```
*/
```

```
IF NOT EXISTS(SELECT IDList FROM ListSpecification WHERE IDList = @IDList)
BEGIN
```

```
INSERT INTO ListSpecification(IDList, ListName,ListDesc)
VALUES (@IDList, LTRIM(RTRIM(@ListName)), LTRIM(RTRIM(@ListDesc)))
SELECT @Error = @@ERROR
IF @Error != 0
BEGIN
```

```
ROLLBACK TRAN
SELECT @IDProcState = 1
SELECT @Message = 'An error occured while creating the new list specification.'
SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '!'
SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
NVARCHAR) + '!'
SELECT @Message = @Message + 'Error occured in Procedure
proc_CreateNewListSpecification.'
SELECT @Message = @Message + 'Procedure is terminated abnormally.'
```

Appendix 5: Supplier's Product Class Database (SPCD) System Code

```
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
END -- End of IF NOT EXISTS...

IF NOT EXISTS( SELECT IDListDef FROM ListVersion WHERE IDListDef = @IDListDef)
BEGIN
    SELECT @NewListVersion = 'Yes'
    INSERT INTO ListVersion(IDList, IDListVer, IDListDef, ListVerDesc)
    VALUES(@IDList,@IDListVer,@IDListDef, LTRIM(RTRIM(@ListVerDesc)) )
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @IDProcState = 2
        SELECT @Message = 'An error ocured while creating the new list specification.'
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
        NVARCHAR) + ' '
        SELECT @Message = @Message + 'Error ocured in Procedure
        proc_CreateNewListSpecification. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
END -- End of IF NOT EXISTS...

/* Now assign the list specification. The following procedure is called for
assigning the list specification. A list specification can be assigned
to a product class or specification group. */

IF @IDAssignToSpecTypeDef IS NOT NULL
BEGIN
    EXEC proc_AssignListSpecification
    @IDListDef      = @IDListDef,
    @IDAssignToSpecTypeDef = @IDAssignToSpecTypeDef,
    @IDProcState    = @IDProcState    OUTPUT,
    @Message        = @Message        OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from
        Proc_CreateNewListSpecification.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
END

/* Now insert values in the ListDefiniton Table. For this we call proc_InsertListValues
procedure */
/*
If the list specification definition already exists in the Supplier PCD Database then we do not need
to perform the following insert list values operation. We check this with the value of
@NewListVersion variable. If @NewListVersion is 'Yes' then it indicates that the list
version is being downloaded for the first time in the Supplier PCD therefore the
proc_InsertValues needs to be called.
*/

--IF NOT EXISTS(SELECT IDListDef FROM ListVersion WHERE IDListDef = @IDListDef)
IF @NewListVersion = 'Yes'
BEGIN
    EXEC dbo.proc_InsertListValues
    @IDListDef      = @IDListDef,
    @ListValues     = @ListValues,
    @ListDMeasUnits = @ListDMeasUnits,
    @Message        = @Message        OUTPUT,
    @IDProcState    = @IDProcState    OUTPUT

    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from
        proc_CreateNewListSpecification.'
        RAISERROR(@Message,1,1) WITH SETERROR
    END
END
```

Appendix 5: Supplier's Product Class Database (SPCD) System Code

```
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        /*If everything goes well until this point it means that all the values are inserted
        properly into the above tables. Now we need to commit the transaction. */
        COMMIT TRAN
        SELECT @Message = 'List Specification Created Successfully. List Specification ID is: ' +
        CAST(@IDList AS NVARCHAR) + ', '
        SELECT @Message = @Message + 'List Specification Version is: ' + CAST(@IDListVer AS
        NVARCHAR) + ', '
        RETURN
    END
END -- End of IF NOT EXISTS...
ELSE
BEGIN
    /*If everything goes well until this point it means that all the values are inserted
    properly into the above tables. Now we need to commit the transaction. */
    COMMIT TRAN
    SELECT @Message = 'List Specification downloaded Successfully. List Specification ID is: ' +
    CAST(@IDList AS NVARCHAR) + ', '
    SELECT @Message = @Message + 'List Specification Version is: ' + CAST(@IDListVer AS
    NVARCHAR) + ', '
    RETURN
END --End of ELSE.
END -- End of proc_CreateNewListSpecification Procedure.
GO
```

/*

2.9 Procedure Name: dbo.proc_CreateNewProductClass

Database: SupplierPCD

Description:

This procedure enables creation of a new product class.

*/

```
CREATE Procedure proc_CreateNewProductClass
```

```
/* Param List */
```

```
@IDProdClass      BIGINT,
@IDProdClassVer    MONEY ,
@IDProdClassDef    BIGINT,
@ProdClassName     NVARCHAR(255),
@ProdClassDesc     NVARCHAR(4000) = NULL,
@ProdClassVerDesc  NVARCHAR(4000) = NULL,
@IDAssignToSpecTypeDef BIGINT = NULL,
@IDProcState       TINYINT OUTPUT,
@Message           NVARCHAR(500) OUTPUT
```

```
AS
```

```
BEGIN
```

```
    DECLARE
```

```
    @Error          INT
```

```
    SELECT @IDProcState = 0
```

```
    SELECT @ERROR = 0
```

```
    BEGIN TRAN
```

```
    /*Create a product class in the Supplier PCD Database by inserting values into ProductClass, product class version
    and Category_ProductClass tables. The order in which the values are inserted into the
    table should be maintained. First values should be inserted into the ProductClass table
    then into ProductClassVersion table because IDProdClass in ProductClassVersion table
    references IDProdClass in ProductClass. Finally the product class should be assigned.
    A product class can be assigned to a product class, category or specification group.
    For assigning the product class we call proc_AssignProductclass procedure.*/
```

```
    /*
```

```
    If the product class already exists in the Supplier PCD Database then we do not need
    to perform the following insert operation.
```

```
    */
```

```
    IF NOT EXISTS(SELECT IDProdClass FROM ProductClass WHERE IDProdClass = @IDProdClass)
```

```
    BEGIN
```

```
        INSERT INTO ProductClass(IDProdClass, ProdClassName,ProdClassDesc)
```

```
        VALUES (@IDProdClass, LTRIM(RTRIM(@ProdClassName)), LTRIM(RTRIM(@ProdClassDesc)) )
```

```
        SELECT @Error = @@ERROR
```

```
        IF @Error != 0
```

Appendix 5: Supplier's Product Class Database (SPCD) System Code

```
BEGIN
    ROLLBACK TRAN
    SELECT @IDProcState = 1
    SELECT @Message = 'An error occured while creating the new product class.'
    SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
    SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
    NVARCHAR) + ' '
    SELECT @Message = @Message + 'Error occured in Procedure proc_CreateNewProductClass.'
    SELECT @Message = @Message + 'Procedure is terminated abnormally.'
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END
END -- End IF NOT EXISTS...

/*
If the product class version already exists in the Supplier PCD Database then we do not need
to perform the following insert operation.
*/
IF NOT EXISTS( SELECT IDProdClassDef FROM ProductClassVersion WHERE IDProdClassDef =
@IDProdClassDef)
BEGIN
    INSERT INTO ProductClassVersion(IDProdClass, IDProdClassVer, IDProdClassDef, ProdClassVerDesc)
    VALUES(@IDProdClass,@IDProdClassVer,@IDProdClassDef, LTRIM(RTRIM(@ProdClassVerDesc)) )
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @IDProcState = 2
        SELECT @Message = 'An error occured while creating the new product class.'
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
        NVARCHAR) + ' '
        SELECT @Message = @Message + 'Error occured in Procedure proc_CreateNewProductClass.'
        SELECT @Message = @Message + 'Procedure is terminated abnormally.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
END -- End IF NOT EXISTS...
/* Assigning product class */

EXEC proc_AssignProductClass
@IDProdClassDef      = @IDProdClassDef,
@IDAssignToSpecTypeDef = @IDAssignToSpecTypeDef,

@IDProcState        = @IDProcState OUTPUT,
@Message            = @Message OUTPUT
IF @IDProcState != 0
BEGIN
    ROLLBACK TRAN
    SELECT @Message = @Message + ' This Procedure was called from proc_CreateNewProductClass.'
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END
ELSE
BEGIN
    /*If everything goes well until this point it means that all the values are inserted
    properly into the above tables. Now we need to commit the transaction. */

    COMMIT TRAN
    SELECT @Message = 'Product Class Downloaded Successfully. Product Class ID is: ' +
    CAST(@IDProdClass AS NVARCHAR) + ' '
    SELECT @Message = @Message + 'Product Class Version is: ' + CAST(@IDProdClassVer AS
    NVARCHAR) + ' '
    RETURN
END
END -- End of Proc_CreateNewProductClass Procedure.
GO
```

/*

Appendix 5: Supplier's Product Class Database (SPCD) System Code

2.10 Procedure Name: dbo.proc_CreateNewSpecification

Database: SupplierPCD

Description:

This procedure enables creation of a new specification.

*/

```
CREATE Procedure proc_CreateNewSpecification
/* Param List */
@IDSpec          BIGINT,
@SpecName        NVARCHAR(255),
@SpecDesc        NVARCHAR(4000) = NULL,
@SpecValue       NVARCHAR(4000) = NULL,
@IDMeasUnit      BIGINT = NULL,
@IDAssignToSpecTypeDef BIGINT = NULL,
@IDProcState     TINYINT OUTPUT,
@Message         NVARCHAR(500) OUTPUT

AS
BEGIN

    DECLARE
    @Error          INT,
    @IDEntityPart  BIGINT
    SELECT @IDProcState = 0
    SELECT @ERROR = 0

    BEGIN TRAN

    /*Create the Specification by inserting values into the Specification table */
    /*
    If the Specification already exists in the Supplier PCD Database then we do not need
    to perform the following insert operation.
    */
    IF NOT EXISTS(SELECT IDSpec FROM Specification WHERE IDSpec = @IDSpec)
    BEGIN
        INSERT INTO Specification(IDSpec, SpecName, SpecDesc)
        VALUES
        (@IDSpec, LTRIM(RTRIM(@SpecName)), LTRIM(RTRIM(@SpecDesc)))
        SELECT @Error = @@ERROR
        IF @Error != 0
        BEGIN
            ROLLBACK TRAN
            SELECT @IDProcState = 1
            SELECT @Message = 'An error occurred while creating the new specification.'
            SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '.'
            SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
            NVARCHAR) + '.'
            SELECT @Message = @Message + 'Error occurred in Procedure proc_CreateNewSpecification.'
            SELECT @Message = @Message + 'Procedure is terminated abnormally.'
            RAISERROR(@Message,1,1) WITH SETERROR
            RETURN @@ERROR
        END
    END -- End of IF NOT EXISTS...

    /* Now assign the specification to Specification Definition under which it was created.
    We call the following procedure for assigning specification.
    */

    IF @IDAssignToSpecTypeDef IS NULL
    BEGIN
        COMMIT TRAN
        SELECT @Message = 'Specification Created Successfully. Specification ID is: ' +
        CAST(@IDSpec AS NVARCHAR) + '.'

    RETURN
    END
    ELSE
    BEGIN
        EXEC proc_AssignSpecification
        @IDSpec          = @IDSpec,
        @IDAssignToSpecTypeDef = @IDAssignToSpecTypeDef,
        @SpecValue       = @SpecValue,
        @IDMeasUnit      = @IDMeasUnit,

        @IDProcState     = @IDProcState OUTPUT,

```

Appendix 5: Supplier's Product Class Database (SPCD) System Code

```
@Message = @Message OUTPUT
IF @IDProcState != 0
BEGIN
    ROLLBACK TRAN
    SELECT @Message = @Message + ' This Procedure was called from
proc_CallAssignSpecification.'
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END
ELSE
BEGIN
    COMMIT TRAN
    SELECT @Message = 'Specification Downloaded Successfully. Specification ID is: ' +
CAST(@IDSpec AS NVARCHAR) + ' .'
    RETURN
END
END -- End of IF @IDEntityPart = 106
END -- End of Proc_CreateNewSpecification Procedure.
GO
```

2.11 Procedure Name: dbo.proc_CreateNewSpecificationGroup

Database: SupplierPCD

Description:

This procedure enables creation of a new specification group.

```
/*
CREATE Procedure proc_CreateNewSpecificationGroup
/* Param List */
@IDSpecGroup          BIGINT,
@IDSpecGroupVer       MONEY,
@IDSpecGroupDef       BIGINT,
@SpecGroupName        NVARCHAR(255),
@SpecGroupDesc        NVARCHAR(4000) = NULL,
@SpecGroupVerDesc     NVARCHAR(4000) = NULL,
@IDAssignToSpecTypeDef BIGINT = NULL,
@IDProcState          TINYINT OUTPUT,
@Message              NVARCHAR(500) OUTPUT

AS
BEGIN
    BEGIN TRAN
    DECLARE
    --@IDSpecGroupDef          BIGINT,
    @Error                    INT,
    @IDEntityPart            INT
    SELECT @IDProcState = 0
    SELECT @ERROR = 0

    /*Create a specification group in the Supplier PCD by inserting values into SpecificationGroup and
    SpecificationGroupVersion tables. The order in which the values are inserted into the
    table should be maintained. First values should be inserted into the SpecificationGroup table
    then into SpecificationGroupVersion table because IDSpecGroup in SpecificationGroupVersion table
    references IDSpecGroup in SpecificationGroup table.*/

    /*
    If the specification group already exists in the Supplier PCD Database then we do not need
    to perform the following insert operation.
    */
    IF NOT EXISTS(SELECT IDSpecGroup FROM SpecificationGroup WHERE IDSpecGroup = @IDSpecGroup)
    BEGIN
        INSERT INTO SpecificationGroup (IDSpecGroup, SpecGroupName, SpecGroupDesc)
        VALUES (@IDSpecGroup, LTRIM(RTRIM(@SpecGroupName)), LTRIM(RTRIM(@SpecGroupDesc)))
        SELECT @Error = @@ERROR
        IF @Error != 0
        BEGIN
            ROLLBACK TRAN
            SELECT @IDProcState = 1
            SELECT @Message = 'An error occurred while creating the new specification group. '
            SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' .'
            SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
            NVARCHAR) + ' .'
            SELECT @Message = @Message + 'Error occurred in Procedure
            proc_CreateNewSpecificationGroup. '
            SELECT @Message = @Message + 'Procedure is terminated abnormally. '

```

Appendix 5: Supplier's Product Class Database (SPCD) System Code

```
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
END -- End of IF NOT EXISTS...

/*
If the specification group version already exists in the Supplier PCD Database then we do not need
to perform the following insert operation.
*/
IF NOT EXISTS( SELECT IDSpecGroupDef FROM SpecificationGroupVersion WHERE IDSpecGroupDef =
@IDSpecGroupDef)
BEGIN
    INSERT INTO SpecificationGroupVersion (IDSpecGroup, IDSpecGroupVer, IDSpecGroupDef,
SpecGroupVerDesc)
VALUES (@IDSpecGroup, @IDSpecGroupVer, @IDSpecGroupDef,
LTRIM(RTRIM(@SpecGroupVerDesc)) )
SELECT @Error = @@ERROR
IF @Error != 0
BEGIN
    ROLLBACK TRAN
    SELECT @IDProcState = 1
    SELECT @Message = 'An error occurred while creating the new specification group. '
    SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
    SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
NVARCHAR) + ' '
    SELECT @Message = @Message + 'Error occurred in Procedure
proc_CreateNewSpecificationGroup. '
    SELECT @Message = @Message + 'Procedure is terminated abnormally. '
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END
END --End of IF NOT EXISTS...
/* Now assign the specification group. The following procedure is called for
assigning the specification group */

IF @IDAssignToSpecTypeDef IS NOT NULL
BEGIN
    EXEC proc_AssignSpecificationGroup
    @IDSpecGroupDef = @IDSpecGroupDef,
    @IDAssignToSpecTypeDef = @IDAssignToSpecTypeDef,
    @IDProcState = @IDProcState OUTPUT,
    @Message = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from
proc_CreateNewSpecificationGroup.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
END
COMMIT TRAN
SELECT @Message = 'Specification group downloaded Successfully. Specification Group ID is: ' +
CAST(@IDSpecGroup AS NVARCHAR) + ' '
SELECT @Message = @Message + 'Specification Group Version is: ' + CAST(@IDSpecGroupVer AS NVARCHAR) +
' '
RETURN
END -- End of Proc_CreateNewSpecificationGroup Procedure.
GO

/*
```

2.12 Procedure Name: dbo.proc_CreateNewTableSpecification

Database: SupplierPCD

Description:

This procedure enables creation of a new table specification.

*/

```
CREATE Procedure proc_CreateNewTableSpecification
/* Param List */
@IDTableSpec          BIGINT,
@IDTableSpecVer      MONEY,
@IDTableVerDef       BIGINT,
@TableName           NVARCHAR(255),
@NumOfRows           INT = NULL,
```


Appendix 5: Supplier's Product Class Database (SPCD) System Code

```

@NumOfColumns          INT,
@ColumnValues           NVARCHAR(4000),
@ColIDMeasUnits        NVARCHAR(4000) = NULL,
@RowContent            NVARCHAR(100),
@TableSpecDesc         NVARCHAR(4000) = NULL,
@TableSpecVerDesc     NVARCHAR(4000) = NULL,
@IDAssignToSpecTypeDef BIGINT = NULL,
@IDProcState          TINYINT OUTPUT,
@Message              NVARCHAR(500) OUTPUT

AS
BEGIN

DECLARE
@Error                INT,
@NewTableVersion      NVARCHAR(5)
SELECT @IDProcState = 0
SELECT @ERROR = 0
SELECT @NewTableVersion = 'No'
BEGIN TRAN

/*Create the table specification in the Supplier PCD by inserting
values into TableSpecification and TableVersion
tables. The order in which the values are inserted into the table should be maintained.
First values should be inserted into the TableSpecification table
then into the TableVersion table because IDTableSpec in TableVersion table
references IDTableSpec in TableSpecification. */

/*
If the Table Specification already exists in the Supplier PCD Database then we do not need
to perform the following insert operation.
*/
IF NOT EXISTS(SELECT IDTableSpec FROM TableSpecification WHERE IDTableSpec = @IDTableSpec)
BEGIN
    INSERT INTO TableSpecification(IDTableSpec, TableSpecName, TableSpecDesc)
    VALUES (@IDTableSpec, LTRIM(RTRIM(@TableSpecName)), LTRIM(RTRIM(@TableSpecDesc)))

    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @IDProcState = 1
        SELECT @Message = 'An error occurred while creating the new table specification. '
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
        NVARCHAR) + ' '
        SELECT @Message = @Message + 'Error occurred in Procedure
        proc_CreateNewTableSpecification. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
END -- End of IF NOT EXISTS...

/*
If the Table Specification version already exists in the Supplier PCD Database then
we do not need to perform the following insert operation.
*/
IF NOT EXISTS(SELECT IDTableVerDef FROM TableVersion WHERE IDTableVerDef = @IDTableVerDef)
BEGIN
    SELECT @NewTableVersion = 'Yes'
    INSERT INTO TableVersion(IDTableSpec, IDTableSpecVer, IDTableVerDef,
    NumOfRows, NumOfColumns, TableVerDesc)
    VALUES(@IDTableSpec,@IDTableSpecVer,@IDTableVerDef, @NumOfRows, @NumOfColumns,
    LTRIM(RTRIM(@TableSpecVerDesc)) )
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @IDProcState = 2
        SELECT @Message = 'An error occurred while creating the new table specification. '
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
        NVARCHAR) + ' '
        SELECT @Message = @Message + 'Error occurred in Procedure
        proc_CreateNewTableSpecification. '

```

Appendix 5: Supplier's Product Class Database (SPCD) System Code

```
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
END -- End of IF NOT EXISTS...

/* Assign the table specification by calling the following procedure */
/* A Table specification can be assigned to a product class or a specification group */
IF @IDAssignToSpecTypeDef IS NOT NULL
BEGIN
    EXEC proc_AssignTableSpecification
        @IDTableVerDef = @IDTableVerDef,
        @IDAssignToSpecTypeDef = @IDAssignToSpecTypeDef,
        @IDProcState = @IDProcState OUTPUT,
        @Message = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from
        proc_CreateNewTableSpecification.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
END

/* Now check how many columns are supplied by the user. Accordingly invoke the procedure
that handle that many columns */
/*
If the table version definition already exists in the Supplier PCD Database then we do not need
to invoke any of the Proc_InsertRow operations.
The following if block checks this. If the downloaded tableVersion is not a new one
then any of the following Proc_InsertRow procedures are not invoked.
*/

IF @NewTableVersion = 'No'
BEGIN
    COMMIT TRAN
    SELECT @Message = 'Table Specification Downloaded Successfully. Table Specification ID is: ' +
    CAST(@IDTableSpec AS NVARCHAR) + ' '
    SELECT @Message = @Message + 'Table Specification Version is: ' + CAST(@IDTableSpecVer AS
    NVARCHAR) + ' '
    RETURN
END

IF @NumOfColumns = 2
BEGIN
    EXEC proc_InsertRow2
        @IDTableVerDef = @IDTableVerDef,
        @ColumnValues = @ColumnValues,
        @ColIDMeasUnits = @ColIDMeasUnits,
        @RowContent = @RowContent,
        @IDProcState = @IDProcState OUTPUT,
        @Message = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from
        proc_CreateNewTableSpecification.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        COMMIT TRAN
        SELECT @Message = 'Table Specification Created Successfully. Table Specification ID is: ' +
        CAST(@IDTableSpec AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Table Specification Version is: ' + CAST(@IDTableSpecVer
        AS NVARCHAR) + ' '
        RETURN
    END
END

IF @NumOfColumns = 3
BEGIN
    EXEC proc_InsertRow3
```

Appendix 5: Supplier's Product Class Database (SPCD) System Code

```
@IDTableVerDef = @IDTableVerDef,
@ColumnValues = @ColumnValues,
@ColIDMeasUnits = @ColIDMeasUnits,
@RowContent = @RowContent,
@IDProcState = @IDProcState OUTPUT,
@Message = @Message OUTPUT
IF @IDProcState != 0
BEGIN
    ROLLBACK TRAN
    SELECT @Message = @Message + ' This Procedure was called from
proc_CreateNewTableSpecification.'
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END
ELSE
BEGIN
    COMMIT TRAN
    SELECT @Message = 'Table Specification Created Successfully. Table Specification ID is: ' +
CAST(@IDTableSpec AS NVARCHAR) + '.'
    SELECT @Message = @Message + 'Table Specification Version is: ' + CAST(@IDTableSpecVer
AS NVARCHAR) + '.'
    RETURN
END
END

IF @NumOfColumns = 4
BEGIN
    EXEC proc_InsertRow4
    @IDTableVerDef = @IDTableVerDef,
    @ColumnValues = @ColumnValues,
    @ColIDMeasUnits = @ColIDMeasUnits,
    @RowContent = @RowContent,
    @IDProcState = @IDProcState OUTPUT,
    @Message = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from
proc_CreateNewTableSpecification.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        COMMIT TRAN
        SELECT @Message = 'Table Specification Created Successfully. Table Specification ID is: ' +
CAST(@IDTableSpec AS NVARCHAR) + '.'
        SELECT @Message = @Message + 'Table Specification Version is: ' + CAST(@IDTableSpecVer
AS NVARCHAR) + '.'
        RETURN
    END
END

IF @NumOfColumns = 5
BEGIN
    EXEC proc_InsertRow5
    @IDTableVerDef = @IDTableVerDef,
    @ColumnValues = @ColumnValues,
    @ColIDMeasUnits = @ColIDMeasUnits,
    @RowContent = @RowContent,
    @IDProcState = @IDProcState OUTPUT,
    @Message = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from
proc_CreateNewTableSpecification.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        COMMIT TRAN
        SELECT @Message = 'Table Specification Created Successfully. Table Specification ID is: ' +
CAST(@IDTableSpec AS NVARCHAR) + '.'
```

Appendix 5: Supplier's Product Class Database (SPCD) System Code

```
        SELECT @Message = @Message + 'Table Specification Version is: ' + CAST(@IDTableSpecVer
        AS NVARCHAR) + ' '
        RETURN
    END
END

IF @NumOfColumns = 6
BEGIN
    EXEC proc_InsertRow6
    @IDTableVerDef = @IDTableVerDef,
    @ColumnValues = @ColumnValues,
    @ColIDMeasUnits = @ColIDMeasUnits,
    @RowContent = @RowContent,
    @IDProcState = @IDProcState OUTPUT,
    @Message = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from
        proc_CreateNewTableSpecification.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        COMMIT TRAN
        SELECT @Message = 'Table Specification Created Successfully. Table Specification ID is: ' +
        CAST(@IDTableSpec AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Table Specification Version is: ' + CAST(@IDTableSpecVer
        AS NVARCHAR) + ' '
        RETURN
    END
END

IF @NumOfColumns = 7
BEGIN
    EXEC proc_InsertRow7
    @IDTableVerDef = @IDTableVerDef,
    @ColumnValues = @ColumnValues,
    @ColIDMeasUnits = @ColIDMeasUnits,
    @RowContent = @RowContent,
    @IDProcState = @IDProcState OUTPUT,
    @Message = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from
        proc_CreateNewTableSpecification.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        COMMIT TRAN
        SELECT @Message = 'Table Specification Created Successfully. Table Specification ID is: ' +
        CAST(@IDTableSpec AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Table Specification Version is: ' + CAST(@IDTableSpecVer
        AS NVARCHAR) + ' '
        RETURN
    END
END

IF @NumOfColumns = 8
BEGIN
    EXEC proc_InsertRow8
    @IDTableVerDef = @IDTableVerDef,
    @ColumnValues = @ColumnValues,
    @ColIDMeasUnits = @ColIDMeasUnits,
    @RowContent = @RowContent,
    @IDProcState = @IDProcState OUTPUT,
    @Message = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
```

Appendix 5: Supplier's Product Class Database (SPCD) System Code

```
        SELECT @Message = @Message + ' This Procedure was called from
        proc_CreateNewTableSpecification.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        COMMIT TRAN
        SELECT @Message = 'Table Specification Created Successfully. Table Specification ID is: ' +
        CAST(@IDTableSpec AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Table Specification Version is: ' + CAST(@IDTableSpecVer
        AS NVARCHAR) + ' '
        RETURN
    END
END

IF @NumOFColumns = 9
BEGIN
    EXEC proc_InsertRow9
    @IDTableVerDef = @IDTableVerDef,
    @ColumnValues = @ColumnValues,
    @ColIDMeasUnits = @ColIDMeasUnits,
    @RowContent = @RowContent,
    @IDProcState = @IDProcState OUTPUT,
    @Message = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from
        proc_CreateNewTableSpecification.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        COMMIT TRAN
        SELECT @Message = 'Table Specification Created Successfully. Table Specification ID is: ' +
        CAST(@IDTableSpec AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Table Specification Version is: ' + CAST(@IDTableSpecVer
        AS NVARCHAR) + ' '
        RETURN
    END
END

IF @NumOFColumns = 10
BEGIN
    EXEC proc_InsertRow10
    @IDTableVerDef = @IDTableVerDef,
    @ColumnValues = @ColumnValues,
    @ColIDMeasUnits = @ColIDMeasUnits,
    @RowContent = @RowContent,
    @IDProcState = @IDProcState OUTPUT,
    @Message = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from
        proc_CreateNewTableSpecification.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        COMMIT TRAN
        SELECT @Message = 'Table Specification Created Successfully. Table Specification ID is: ' +
        CAST(@IDTableSpec AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Table Specification Version is: ' + CAST(@IDTableSpecVer
        AS NVARCHAR) + ' '
        RETURN
    END
END

END -- End of Proc_CreateNewTableSpecification Procedure.
GO

/*
```

2.13 Procedure Name: dbo.proc_InsertListValues

Database: SupplierPCD

Description:

This procedure enables creation of list values for list specification.

*/

```

CREATE Procedure proc_InsertListValues
/* Param List */
@IDListDef          BIGINT,
@ListValues         NVARCHAR(4000),
@ListIDMeasUnits   NVARCHAR(4000) = NULL,
@IDProcState       TINYINT OUTPUT,
@Message           NVARCHAR(500) OUTPUT

AS
BEGIN
    DECLARE
    @Error          INT
    SELECT @IDProcState = 0
    SELECT @ERROR = 0
    SELECT @Message = ''

    /*Create a temporary table and insert the values from the table returned
    by the function fn_getListValuesTable */

    CREATE TABLE #TempTableListValues (ListID INT, ListValue NVARCHAR(255) )
    INSERT INTO #TempTableListValues
    SELECT ColumnID,ColVal FROM dbo.fn_getColumnValuesTable(@ListValues)

    /*Create another temporary table and insert IDMeas units from the table returned
    by the function fn_getListValuesTable */

    CREATE TABLE #TempTableIDMeasUnit (MeasID INT, IDMeasUnit BIGINT )
    INSERT INTO #TempTableIDMeasUnit
    SELECT ColumnID, CAST(ColVal AS BIGINT) FROM dbo.fn_getColumnValuesTable(@ListIDMeasUnits)
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        SELECT @IDProcState = 1
        SELECT @Message = 'An error occured while inserting the list specification. '
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Error occured in Procedure proc_InsertListValues. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END

    /* Now insert the values into the List Definition table */
    INSERT INTO ListDefinition (IDListDef, ListValue, IDMeasUnit)
    SELECT @IDListDef, a.ListValue, b.IDMeasUnit
    FROM #TempTableListValues a, #TempTableIDMeasUnit b
    WHERE a.ListID = b.MeasID

    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        SELECT @IDProcState = 2
        SELECT @Message = 'An error occured while inserting the list specification. '
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Error occured in Procedure proc_InsertListValues. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        DROP TABLE #TempTableListValues
        DROP TABLE #TempTableIDMeasUnit
        RETURN
    END
END
END -- End of proc_InsertListValues Procedure.
GO

```

Appendix 5: Supplier's Product Class Database (SPCD) System Code

/* 2.14 Procedure Name: dbo.proc_InsertRow2

Database: SupplierPCD

Description:

This procedure enables creation of table specification values having 2 columns.

*/

```
CREATE Procedure proc_InsertRow2
```

```
/* Param List */
```

```
@IDTableVerDef          BIGINT,  
@ColumnValues           NVARCHAR(4000),  
@ColIDMeasUnits        NVARCHAR(4000) = NULL,  
@RowContent            NVARCHAR(100),  
@IDProcState           TINYINT OUTPUT,  
@Message               NVARCHAR(500) OUTPUT
```

```
AS  
BEGIN
```

```
DECLARE
```

```
@Error                 INT,  
@IDRow                INT,  
@NumOfRows            INT,  
@Column1Val           NVARCHAR(255),  
@Column2Val           NVARCHAR(255)
```

```
SELECT @IDProcState = 0  
SELECT @ERROR = 0  
SELECT @Message = ''
```

```
SELECT @NumOfRows = COUNT(*) FROM dbo.fn_getColumnValuesTable(@ColumnValues)  
PRINT '@NumOfRows: ' + CAST(@NumOfRows AS NVARCHAR)  
/* The function fn_getColumnValuesTable(@ColumnValues) should return two rows only */
```

```
IF @NumOfRows != 2  
BEGIN
```

```
    SELECT @IDProcState = 1  
    SELECT @Message = 'An error occurred while inserting the table row.'  
    SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + '!'  
    SELECT @Message = @Message + 'Error occurred in Procedure proc_InsertRow2.'  
    SELECT @Message = @Message + 'Procedure is terminated abnormally.'  
    RAISERROR(@Message,1,1) WITH SETERROR  
    RETURN
```

```
END
```

```
/*Create a temporary table and insert the values from the table returned  
by the function fn_getColumnValuesTable */
```

```
CREATE TABLE #TempTable (ColumnID INT, ColumnValue NVARCHAR(255) )  
INSERT INTO #TempTable  
SELECT ColumnID,ColVal FROM dbo.fn_getColumnValuesTable(@ColumnValues)
```

```
SELECT @Column1Val = ColumnValue FROM #TempTable WHERE ColumnID = 1  
SELECT @Column2Val = ColumnValue FROM #TempTable WHERE ColumnID = 2
```

```
/*Check whether the row is contains column values or column specifications  
A row can contain column values or column specifications(headers).*/
```

```
IF @RowContent LIKE 'ColVals'  
BEGIN
```

```
    /* Check for the existing rows with same IDTableVerDef to keep track of the number  
    of rows the IDTableVerDef has if the @RowContent = 'ColumnValues' */  
    SELECT @IDRow = MAX(IDRow) FROM TableDefinition2 WHERE IDTableVerDef = @IDTableVerDef  
    IF @IDRow IS NULL  
        SELECT @IDRow = 1  
    ELSE  
        SELECT @IDRow = @IDRow + 1
```

```
    INSERT INTO TableDefinition2(IDTableVerDef, IDRow, Column1Val, Column2Val, RowContent)  
    VALUES(@IDTableVerDef, @IDRow,  
    LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)), LTRIM(RTRIM(@RowContent)))  
    SELECT @Error = @@ERROR
```

Appendix 5: Supplier's Product Class Database (SPCD) System Code

```
IF @Error != 0
BEGIN
    SELECT @IDProcState = 2
    SELECT @Message = 'An error occured while inserting the table row. '
    SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '. '
    SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
    NVARCHAR) + '. '
    SELECT @Message = @Message + 'Error occured in Procedure proc_InsertRow2. '
    SELECT @Message = @Message + 'Procedure is terminated abnormally. '
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END
ELSE
BEGIN
    DROP TABLE #TempTable
    RETURN
END
END --End of IF @RowContent LIKE 'TableRow'
ELSE
BEGIN
    SELECT @IDRow = 0
    INSERT INTO TableDefinition2(IDTableVerDef, IDRow, Column1Val, Column2Val, RowContent)
    VALUES(@IDTableVerDef, @IDRow,
    LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)), LTRIM(RTRIM(@RowContent)))
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        SELECT @IDProcState = 3
        SELECT @Message = 'An error occured while inserting the table row. '
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '. '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
        NVARCHAR) + '. '
        SELECT @Message = @Message + 'Error occured in Procedure proc_InsertRow2. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        DROP TABLE #TempTable
        RETURN
    END
END
END -- End of Proc_InsertRow2 Procedure.
GO
```

2.15 Procedure Name: dbo.proc_InsertRow3

Database: SupplierPCD

Description:

This procedure enables creation of table specification values having 3 columns.

*/

```
CREATE Procedure proc_InsertRow3
/* Param List */
@IDTableVerDef          BIGINT,
@ColumnValues           NVARCHAR(4000),
@ColIDMeasUnits         NVARCHAR(4000) = NULL,
@RowContent             NVARCHAR(100),
@IDProcState            TINYINT OUTPUT,
@Message                NVARCHAR(500) OUTPUT
```

```
AS
BEGIN
```

```
    DECLARE
    @Error                INT,
    @IDRow                INT,
    @NumOfRows           INT,
    @Column1Val           NVARCHAR(255),
    @Column2Val           NVARCHAR(255),
    @Column3Val           NVARCHAR(255)
```

```
    SELECT @IDProcState = 0
    SELECT @ERROR = 0
```


Appendix 5: Supplier's Product Class Database (SPCD) System Code

```
SELECT @Message = ''

SELECT @NumOfRows = COUNT(*) FROM dbo.fn_getColumnValuesTable(@ColumnValues)
PRINT '@NumOfRows: ' + CAST(@NumOfRows AS NVARCHAR)

/* The function fn_getColumnValuesTable(@ColumnValues) should return three rows only */

IF @NumOfRows != 3
BEGIN
    SELECT @IDProcState = 1
    SELECT @Message = 'An error occurred while inserting the table row. '
    SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + '. '
    SELECT @Message = @Message + 'Error occurred in Procedure proc_InsertRow3. '
    SELECT @Message = @Message + 'Procedure is terminated abnormally. '
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN
END

/*Create a temporary table and insert the values from the table returned
by the function fn_getColumnValuesTable */

CREATE TABLE #TempTable (ColumnID INT, ColumnValue NVARCHAR(255) )
INSERT INTO #TempTable
SELECT ColumnID,ColVal FROM dbo.fn_getColumnValuesTable(@ColumnValues)

SELECT @Column1Val = ColumnValue FROM #TempTable WHERE ColumnID = 1
SELECT @Column2Val = ColumnValue FROM #TempTable WHERE ColumnID = 2
SELECT @Column3Val = ColumnValue FROM #TempTable WHERE ColumnID = 3

/*Check whether the row is contains column values or column specifications
A row can contain column values or column specifications(headers).*/

IF @RowContent LIKE 'ColVals'
BEGIN

    /* Check for the existing rows with same IDTableVerDef to keep track of the number
of rows the IDTableVerDef has if the @RowContent = 'ColumnValues' */

    SELECT @IDRow = MAX(IDRow) FROM TableDefinition3 WHERE IDTableVerDef = @IDTableVerDef
    IF @IDRow IS NULL
        SELECT @IDRow = 1
    ELSE
        SELECT @IDRow = @IDRow + 1

    INSERT INTO TableDefinition3(IDTableVerDef, IDRow, Column1Val, Column2Val, Column3Val,
RowContent)
VALUES(@IDTableVerDef, @IDRow,
LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),
LTRIM(RTRIM(@Column3Val)), LTRIM(RTRIM(@RowContent)))
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        SELECT @IDProcState = 2
        SELECT @Message = 'An error occurred while inserting the table row. '
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '. '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
NVARCHAR) + '. '
        SELECT @Message = @Message + 'Error occurred in Procedure proc_InsertRow3. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        DROP TABLE #TempTable
        RETURN
    END
END
--End of IF @RowContent LIKE 'TableRow'
ELSE
BEGIN
    SELECT @IDRow = 0
    INSERT INTO TableDefinition3(IDTableVerDef, IDRow, Column1Val, Column2Val,Column3Val,
RowContent)
```

Appendix 5: Supplier's Product Class Database (SPCD) System Code

```
VALUES(@IDTableVerDef, @IDRow,
LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),
LTRIM(RTRIM(@Column3Val)),LTRIM(RTRIM(@RowContent))
SELECT @Error = @@ERROR
IF @Error != 0
BEGIN
    SELECT @IDProcState = 3
    SELECT @Message = 'An error occured while inserting the table row. '
    SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '.'
    SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
    NVARCHAR) + '.'
    SELECT @Message = @Message + 'Error occured in Procedure proc_InsertRow3. '
    SELECT @Message = @Message + 'Procedure is terminated abnormally. '
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END
ELSE
BEGIN
    DROP TABLE #TempTable
    RETURN
END
END
END -- End of Proc_InsertRow3 Procedure.
GO
```

2.16 Procedure Name: dbo.proc_InsertRow4

Database: SupplierPCD

Description:

This procedure enables creation of table specification values having 4 columns.

*/

```
CREATE Procedure proc_InsertRow4
/* Param List */
@IDTableVerDef          BIGINT,
@ColumnValues            NVARCHAR(4000),
@ColIDMeasUnits         NVARCHAR(4000) = NULL,
@RowContent             NVARCHAR(100),
@IDProcState            TINYINT OUTPUT,
@Message                NVARCHAR(500) OUTPUT

AS
BEGIN
    DECLARE
    @Error                INT,
    @IDRow               INT,
    @NumOfRows           INT,
    @Column1Val          NVARCHAR(255),
    @Column2Val          NVARCHAR(255),
    @Column3Val          NVARCHAR(255),
    @Column4Val          NVARCHAR(255)

    SELECT @IDProcState = 0
    SELECT @ERROR = 0
    SELECT @Message = ''

    SELECT @NumOfRows = COUNT(*) FROM dbo.fn_getColumnValuesTable(@ColumnValues)
    --PRINT '@NumOfRows: ' + CAST(@NumOfRows AS NVARCHAR)
    /* The function fn_getColumnValuesTable(@ColumnValues) should return two rows only */

    IF @NumOfRows != 4
    BEGIN
        SELECT @IDProcState = 1
        SELECT @Message = 'An error occured while inserting the table row. '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + '.'
        SELECT @Message = @Message + 'Error occured in Procedure proc_InsertRow4. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN
    END

    /*Create a temporary table and insert the values from the table returned
    by the function fn_getColumnValuesTable */
```

Appendix 5: Supplier's Product Class Database (SPCD) System Code

```
CREATE TABLE #TempTable (ColumnID INT, ColumnValue NVARCHAR(255) )
INSERT INTO #TempTable
SELECT ColumnID,ColVal FROM dbo.fn_getColumnValuesTable(@ColumnValues)

SELECT @Column1Val = ColumnValue FROM #TempTable WHERE ColumnID = 1
SELECT @Column2Val = ColumnValue FROM #TempTable WHERE ColumnID = 2
SELECT @Column3Val = ColumnValue FROM #TempTable WHERE ColumnID = 3
SELECT @Column4Val = ColumnValue FROM #TempTable WHERE ColumnID = 4

/*Check whether the row contains column values or column specifications
A row can only contain column values or column specifications(headers).*/

IF @RowContent LIKE 'ColVals'
BEGIN
    /* Check for the existing rows with same IDTableVerDef to keep track of the number
of rows the IDTableVerDef has if the @RowContent = 'ColumnValues' */
    SELECT @IDRow = MAX(IDRow) FROM TableDefinition4 WHERE IDTableVerDef = @IDTableVerDef
    IF @IDRow IS NULL
        SELECT @IDRow = 1
    ELSE
        SELECT @IDRow = @IDRow + 1

    INSERT INTO TableDefinition4(IDTableVerDef, IDRow, Column1Val,
Column2Val,Column3Val,Column4Val, RowContent)
VALUES(@IDTableVerDef, @IDRow,
LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),LTRIM(RTRIM(@Column3Val)),
LTRIM(RTRIM(@Column4Val)), LTRIM(RTRIM(@RowContent)) )
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        SELECT @IDProcState = 2
        SELECT @Message = 'An error occurred while inserting the table row. '
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '. '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
NVARCHAR) + '. '
        SELECT @Message = @Message + 'Error occurred in Procedure proc_InsertRow4. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        DROP TABLE #TempTable
        RETURN
    END
END
--End of IF @RowContent LIKE 'TableRow'
ELSE
BEGIN
    SELECT @IDRow = 0
    INSERT INTO TableDefinition4(IDTableVerDef, IDRow, Column1Val, Column2Val, Column3Val,
Column4Val, RowContent)
VALUES(@IDTableVerDef, @IDRow,
LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),
LTRIM(RTRIM(@Column3Val)),LTRIM(RTRIM(@Column4Val)),LTRIM(RTRIM(@RowContent)) )
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        SELECT @IDProcState = 3
        SELECT @Message = 'An error occurred while inserting the table row. '
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '. '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
NVARCHAR) + '. '
        SELECT @Message = @Message + 'Error occurred in Procedure proc_InsertRow4. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        DROP TABLE #TempTable
        RETURN
    END
END
END
-- End of Proc_InsertRow4 Procedure.
GO
```

2.17 Procedure Name: dbo.proc_InsertRow5

Database: SupplierPCD

Description:

This procedure enables creation of table specification values having 5 columns.

```

*/
CREATE Procedure proc_InsertRow5
/* Param List */
@IDTableVerDef          BIGINT,
@ColumnValues            NVARCHAR(4000),
@ColIDMeasUnits         NVARCHAR(4000) = NULL,
@RowContent             NVARCHAR(100),
@IDProcState           TINYINT OUTPUT,
@Message                NVARCHAR(500) OUTPUT
AS
BEGIN
    DECLARE
        @Error           INT,
        @IDRow           INT,
        @NumOfRows       INT,
        @Column1Val      NVARCHAR(255),
        @Column2Val      NVARCHAR(255),
        @Column3Val      NVARCHAR(255),
        @Column4Val      NVARCHAR(255),
        @Column5Val      NVARCHAR(255)

    SELECT @IDProcState = 0
    SELECT @ERROR = 0
    SELECT @Message = ''

    SELECT @NumOfRows = COUNT(*) FROM dbo.fn_getColumnValuesTable(@ColumnValues)
    --PRINT @NumOfRows: ' + CAST(@NumOfRows AS NVARCHAR)
    /* The function fn_getColumnValuesTable(@ColumnValues) should return two rows only */

    IF @NumOfRows != 5
    BEGIN
        SELECT @IDProcState = 1
        SELECT @Message = 'An error occured while inserting the table row. '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Error occured in Procedure proc_InsertRow5. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN
    END

    /*Create a temporary table and insert the values from the table returned
    by the function fn_getColumnValuesTable */

    CREATE TABLE #TempTable (ColumnID INT, ColumnValue NVARCHAR(255) )
    INSERT INTO #TempTable
    SELECT ColumnID,ColVal FROM dbo.fn_getColumnValuesTable(@ColumnValues)

    SELECT @Column1Val = ColumnValue FROM #TempTable WHERE ColumnID = 1
    SELECT @Column2Val = ColumnValue FROM #TempTable WHERE ColumnID = 2
    SELECT @Column3Val = ColumnValue FROM #TempTable WHERE ColumnID = 3
    SELECT @Column4Val = ColumnValue FROM #TempTable WHERE ColumnID = 4
    SELECT @Column5Val = ColumnValue FROM #TempTable WHERE ColumnID = 5

    /*Check whether the row contains column values or column specifications
    A row can only contain column values or column specifications(headers).*/

    IF @RowContent LIKE 'ColVals'
    BEGIN
        /* Check for the existing rows with same IDTableVerDef to keep track of the number
        of rows the IDTableVerDef has if the @RowContent = 'ColumnValues' */
        SELECT @IDRow = MAX(IDRow) FROM TableDefinition2 WHERE IDTableVerDef = @IDTableVerDef
        IF @IDRow IS NULL
            SELECT @IDRow = 1
        ELSE
            SELECT @IDRow = @IDRow + 1
    END

```

Appendix 5: Supplier's Product Class Database (SPCD) System Code

```
INSERT INTO TableDefinition5(IDTableVerDef, IDRow, Column1Val,
Column2Val,Column3Val,Column4Val, Column5Val, RowContent)
VALUES(@IDTableVerDef, @IDRow,
LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),LTRIM(RTRIM(@Column3Val)),
LTRIM(RTRIM(@Column4Val)),LTRIM(RTRIM(@Column5Val)), LTRIM(RTRIM(@RowContent)))
SELECT @Error = @@ERROR
IF @Error != 0
BEGIN
    SELECT @IDProcState = 2
    SELECT @Message = 'An error occured while inserting the table row. '
    SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
    SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
    NVARCHAR) + ' '
    SELECT @Message = @Message + 'Error occured in Procedure proc_InsertRow5. '
    SELECT @Message = @Message + 'Procedure is terminated abnormally. '
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END
ELSE
BEGIN
    DROP TABLE #TempTable
    RETURN
END
END --End of IF @RowContent LIKE 'TableRow'
ELSE
BEGIN
    SELECT @IDRow = 0
    INSERT INTO TableDefinition5(IDTableVerDef, IDRow, Column1Val, Column2Val, Column3Val,
Column4Val, Column5Val, RowContent)
VALUES(@IDTableVerDef, @IDRow, LTRIM(RTRIM(@Column1Val)),
LTRIM(RTRIM(@Column2Val)),
LTRIM(RTRIM(@Column3Val)),LTRIM(RTRIM(@Column4Val)),LTRIM(RTRIM(@Column5Val)),LTRI
M(RTRIM(@RowContent)))
SELECT @Error = @@ERROR
IF @Error != 0
BEGIN
    SELECT @IDProcState = 3
    SELECT @Message = 'An error occured while inserting the table row. '
    SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
    SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
    NVARCHAR) + ' '
    SELECT @Message = @Message + 'Error occured in Procedure proc_InsertRow5. '
    SELECT @Message = @Message + 'Procedure is terminated abnormally. '
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END
ELSE
BEGIN
    DROP TABLE #TempTable
    RETURN
END
END
END -- End of Proc_InsertRow2 Procedure.
GO
```

2.18 Procedure Name: dbo. proc_InsertRow6

Database: SupplierPCD

Description:

This procedure enables creation of table specification values having 6 columns.

*/

```
CREATE Procedure proc_InsertRow6
/* Param List */
@IDTableVerDef          BIGINT,
@Column Values          NVARCHAR(4000),
@ColIDMeasUnits        NVARCHAR(4000) = NULL,
@RowContent            NVARCHAR(100),
@IDProcState          TINYINT OUTPUT,
@Message              NVARCHAR(500) OUTPUT
```

```
AS
BEGIN
```

Appendix 5: Supplier's Product Class Database (SPCD) System Code

```
DECLARE
@Error          INT,
@IDRow          INT,
@NumOfRows     INT,
@Column1Val     NVARCHAR(255),
@Column2Val     NVARCHAR(255),
@Column3Val     NVARCHAR(255),
@Column4Val     NVARCHAR(255),
@Column5Val     NVARCHAR(255),
@Column6Val     NVARCHAR(255)

SELECT @IDProcState = 0
SELECT @ERROR = 0
SELECT @Message = ''

SELECT @NumOfRows = COUNT(*) FROM dbo.fn_getColumnValuesTable(@ColumnValues)
--PRINT '@NumOfRows: ' + CAST(@NumOfRows AS NVARCHAR)
/* The function fn_getColumnValuesTable(@ColumnValues) should return two rows only */

IF @NumOfRows != 6
BEGIN
    SELECT @IDProcState = 1
    SELECT @Message = 'An error occurred while inserting the table row. '
    SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + '. '
    SELECT @Message = @Message + 'Error occurred in Procedure Proc_InsertRow6. '
    SELECT @Message = @Message + 'Procedure is terminated abnormally. '
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN
END

/*Create a temporary table and insert the values from the table returned
by the function fn_getColumnValuesTable */

CREATE TABLE #TempTable (ColumnID INT, ColumnValue NVARCHAR(255) )
INSERT INTO #TempTable
SELECT ColumnID,ColVal FROM dbo.fn_getColumnValuesTable(@ColumnValues)

SELECT @Column1Val = ColumnValue FROM #TempTable WHERE ColumnID = 1
SELECT @Column2Val = ColumnValue FROM #TempTable WHERE ColumnID = 2
SELECT @Column3Val = ColumnValue FROM #TempTable WHERE ColumnID = 3
SELECT @Column4Val = ColumnValue FROM #TempTable WHERE ColumnID = 4
SELECT @Column5Val = ColumnValue FROM #TempTable WHERE ColumnID = 5
SELECT @Column6Val = ColumnValue FROM #TempTable WHERE ColumnID = 6

/*Check whether the row contains column values or column specifications
A row can only contain column values or column specifications(headers).*/

IF @RowContent LIKE 'ColVals'
BEGIN
    /* Check for the existing rows with same IDTableVerDef to keep track of the number
of rows the IDTableVerDef has if the @RowContent = 'ColumnValues' */
    SELECT @IDRow = MAX(IDRow) FROM TableDefinition6 WHERE IDTableVerDef = @IDTableVerDef
    IF @IDRow IS NULL
        SELECT @IDRow = 1
    ELSE
        SELECT @IDRow = @IDRow + 1

    INSERT INTO TableDefinition6(IDTableVerDef, IDRow, Column1Val,
Column2Val,Column3Val,Column4Val,
Column5Val, Column6Val, RowContent)
VALUES(@IDTableVerDef, @IDRow,
LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),LTRIM(RTRIM(@Column3Val)),
LTRIM(RTRIM(@Column4Val)),LTRIM(RTRIM(@Column5Val)),
LTRIM(RTRIM(@Column6Val)),LTRIM(RTRIM(@RowContent)) )
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        SELECT @IDProcState = 2
        SELECT @Message = 'An error occurred while inserting the table row. '
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '. '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
NVARCHAR) + '. '
        SELECT @Message = @Message + 'Error occurred in Procedure Proc_InsertRow6. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
    END
END
```

Appendix 5: Supplier's Product Class Database (SPCD) System Code

```
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        DROP TABLE #TempTable
        RETURN
    END
END --End of IF @RowContent LIKE 'TableRow'
ELSE
BEGIN
    SELECT @IDRow = 0
    INSERT INTO TableDefinition6(IDTableVerDef, IDRow, Column1Val, Column2Val, Column3Val,
    Column4Val, Column5Val,
    Column6Val,RowContent)
    VALUES(@IDTableVerDef, @IDRow,
LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),
    LTRIM(RTRIM(@Column3Val)),LTRIM(RTRIM(@Column4Val)),LTRIM(RTRIM(@Column5Val)),
    LTRIM(RTRIM(@Column6Val)),LTRIM(RTRIM(@RowContent)))
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        SELECT @IDProcState = 3
        SELECT @Message = 'An error occured while inserting the table row. '
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
        NVARCHAR) + ' '
        SELECT @Message = @Message + 'Error occured in Procedure Proc_InsertRow6. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        DROP TABLE #TempTable
        RETURN
    END
END
END -- End of Proc_InsertRow6 Procedure.
GO
```

2.19 Procedure Name: dbo.proc_InsertRow7

Database: SupplierPCD

Description:

This procedure enables creation of table specification values having 7 columns.

```
*/
CREATE Procedure proc_InsertRow7
/* Param List */
@IDTableVerDef          BIGINT,
@ColumnValues           NVARCHAR(4000),
@ColIDMeasUnits         NVARCHAR(4000) = NULL,
@RowContent             NVARCHAR(100),
@IDProcState           TINYINT OUTPUT,
@Message               NVARCHAR(500) OUTPUT
```

```
AS
BEGIN
```

```
    DECLARE
    @Error                INT,
    @IDRow               INT,
    @NumOfRows           INT,
    @Column1Val          NVARCHAR(255),
    @Column2Val          NVARCHAR(255),
    @Column3Val          NVARCHAR(255),
    @Column4Val          NVARCHAR(255),
    @Column5Val          NVARCHAR(255),
    @Column6Val          NVARCHAR(255),
    @Column7Val          NVARCHAR(255)
```

```
    SELECT @IDProcState = 0
    SELECT @ERROR = 0
    SELECT @Message = ''
```

Appendix 5: Supplier's Product Class Database (SPCD) System Code

```
SELECT @NumOfRows = COUNT(*) FROM dbo.fn_getColumnValuesTable(@ColumnValues)
--PRINT @NumOfRows: ' + CAST(@NumOfRows AS NVARCHAR)
/* The function fn_getColumnValuesTable(@ColumnValues) should return two rows only */

IF @NumOfRows != 7
BEGIN
    SELECT @IDProcState = 1
    SELECT @Message = 'An error occurred while inserting the table row. '
    SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + ' '
    SELECT @Message = @Message + 'Error occurred in Procedure Proc_InsertRow7. '
    SELECT @Message = @Message + 'Procedure is terminated abnormally. '
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN
END

/*Create a temporary table and insert the values from the table returned
by the function fn_getColumnValuesTable */

CREATE TABLE #TempTable (ColumnID INT, ColumnValue NVARCHAR(255) )
INSERT INTO #TempTable
SELECT ColumnID,ColVal FROM dbo.fn_getColumnValuesTable(@ColumnValues)

SELECT @Column1Val = ColumnValue FROM #TempTable WHERE ColumnID = 1
SELECT @Column2Val = ColumnValue FROM #TempTable WHERE ColumnID = 2
SELECT @Column3Val = ColumnValue FROM #TempTable WHERE ColumnID = 3
SELECT @Column4Val = ColumnValue FROM #TempTable WHERE ColumnID = 4
SELECT @Column5Val = ColumnValue FROM #TempTable WHERE ColumnID = 5
SELECT @Column6Val = ColumnValue FROM #TempTable WHERE ColumnID = 6
SELECT @Column7Val = ColumnValue FROM #TempTable WHERE ColumnID = 7

/*Check whether the row contains column values or column specifications
A row can only contain column values or column specifications(headers).*/
IF @RowContent LIKE 'ColVals'
BEGIN
    /* Check for the existing rows with same IDTableVerDef to keep track of the number
of rows the IDTableVerDef has if the @RowContent = 'ColumnValues' */
    SELECT @IDRow = MAX(IDRow) FROM TableDefinition7 WHERE IDTableVerDef = @IDTableVerDef
    IF @IDRow IS NULL
        SELECT @IDRow = 1
    ELSE
        SELECT @IDRow = @IDRow + 1

    INSERT INTO TableDefinition7(IDTableVerDef, IDRow, Column1Val,
Column2Val,Column3Val,Column4Val,
Column5Val, Column6Val,Column7Val, RowContent)
VALUES(@IDTableVerDef, @IDRow,
LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),LTRIM(RTRIM(@Column3Val)),
LTRIM(RTRIM(@Column4Val)),LTRIM(RTRIM(@Column5Val)), LTRIM(RTRIM(@Column6Val)),
LTRIM(RTRIM(@Column7Val)),LTRIM(RTRIM(@RowContent)) )
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        SELECT @IDProcState = 2
        SELECT @Message = 'An error occurred while inserting the table row. '
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
NVARCHAR) + ' '
        SELECT @Message = @Message + 'Error occurred in Procedure Proc_InsertRow7. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        DROP TABLE #TempTable
        RETURN
    END
END
END --End of IF @RowContent LIKE 'TableRow'
ELSE
BEGIN
    SELECT @IDRow = 0
    INSERT INTO TableDefinition7(IDTableVerDef, IDRow, Column1Val, Column2Val, Column3Val,
Column4Val, Column5Val,
```


Appendix 5: Supplier's Product Class Database (SPCD) System Code

```
Column6Val,Column7Val, RowContent)
VALUES(@IDTableVerDef, @IDRow,
LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),
LTRIM(RTRIM(@Column3Val)),LTRIM(RTRIM(@Column4Val)),LTRIM(RTRIM(@Column5Val)),
LTRIM(RTRIM(@Column6Val)),LTRIM(RTRIM(@Column7Val)),LTRIM(RTRIM(@RowContent)))
SELECT @Error = @@ERROR
IF @Error != 0
BEGIN
    SELECT @IDProcState = 3
    SELECT @Message = 'An error occured while inserting the table row. '
    SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '. '
    SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
    NVARCHAR) + '. '
    SELECT @Message = @Message + 'Error occured in Procedure Proc_InsertRow7. '
    SELECT @Message = @Message + 'Procedure is terminated abnormally. '
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END
ELSE
BEGIN
    DROP TABLE #TempTable
    RETURN
END
END
END -- End of Proc_InsertRow7 Procedure.
GO
```

2.20 Procedure Name: dbo.proc_InsertRow8

Database: SupplierPCD

Description:

This procedure enables creation of table specification values having 8 columns.

*/

```
CREATE Procedure proc_InsertRow8
/* Param List */
@IDTableVerDef          BIGINT,
@ColumnValues           NVARCHAR(4000),
@CollIDMeasUnits       NVARCHAR(4000) = NULL,
@RowContent            NVARCHAR(100),
@IDProcState           TINYINT OUTPUT,
@Message               NVARCHAR(500) OUTPUT

AS
BEGIN

    DECLARE
    @Error              INT,
    @IDRow             INT,
    @NumOfRows         INT,
    @Column1Val        NVARCHAR(255),
    @Column2Val        NVARCHAR(255),
    @Column3Val        NVARCHAR(255),
    @Column4Val        NVARCHAR(255),
    @Column5Val        NVARCHAR(255),
    @Column6Val        NVARCHAR(255),
    @Column7Val        NVARCHAR(255),
    @Column8Val        NVARCHAR(255)

    SELECT @IDProcState = 0
    SELECT @ERROR = 0
    SELECT @Message = ''

    SELECT @NumOfRows = COUNT(*) FROM dbo.fn_getColumnValuesTable(@ColumnValues)
    --PRINT '@NumOfRows: ' + CAST(@NumOfRows AS NVARCHAR)
    /* The function fn_getColumnValuesTable(@ColumnValues) should return two rows only */

    IF @NumOfRows != 8
    BEGIN
        SELECT @IDProcState = 1
        SELECT @Message = 'An error occured while inserting the table row. '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + '. '
        SELECT @Message = @Message + 'Error occured in Procedure Proc_InsertRow8. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
    END
```

Appendix 5: Supplier's Product Class Database (SPCD) System Code

```
RAISERROR(@Message,1,1) WITH SETERROR
RETURN
END

/*Create a temporary table and insert the values from the table returned
by the function fn_getColumnValuesTable */

CREATE TABLE #TempTable (ColumnID INT, ColumnValue NVARCHAR(255) )
INSERT INTO #TempTable
SELECT ColumnID,ColVal FROM dbo.fn_getColumnValuesTable(@ColumnValues)

SELECT @Column1Val = ColumnValue FROM #TempTable WHERE ColumnID = 1
SELECT @Column2Val = ColumnValue FROM #TempTable WHERE ColumnID = 2
SELECT @Column3Val = ColumnValue FROM #TempTable WHERE ColumnID = 3
SELECT @Column4Val = ColumnValue FROM #TempTable WHERE ColumnID = 4
SELECT @Column5Val = ColumnValue FROM #TempTable WHERE ColumnID = 5
SELECT @Column6Val = ColumnValue FROM #TempTable WHERE ColumnID = 6
SELECT @Column7Val = ColumnValue FROM #TempTable WHERE ColumnID = 7
SELECT @Column8Val = ColumnValue FROM #TempTable WHERE ColumnID = 8

/*Check whether the row contains column values or column specifications
A row can only contain column values or column specifications(headers).*/

IF @RowContent LIKE 'ColVals'
BEGIN
    /* Check for the existing rows with same IDTableVerDef to keep track of the number
of rows the IDTableVerDef has if the @RowContent = 'ColumnValues' */
    SELECT @IDRow = MAX(IDRow) FROM TableDefinition8 WHERE IDTableVerDef = @IDTableVerDef
    IF @IDRow IS NULL
        SELECT @IDRow = 1
    ELSE
        SELECT @IDRow = @IDRow + 1

    INSERT INTO TableDefinition8(IDTableVerDef, IDRow, Column1Val,
Column2Val,Column3Val,Column4Val,
Column5Val, Column6Val,Column7Val, Column8Val, RowContent)
VALUES(@IDTableVerDef, @IDRow,
LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),LTRIM(RTRIM(@Column3Val)),
LTRIM(RTRIM(@Column4Val)),LTRIM(RTRIM(@Column5Val)), LTRIM(RTRIM(@Column6Val)),
LTRIM(RTRIM(@Column7Val)),LTRIM(RTRIM(@Column8Val)),LTRIM(RTRIM(@RowContent)))
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        SELECT @IDProcState = 2
        SELECT @Message = 'An error occured while inserting the table row. '
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '. '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
NVARCHAR) + '. '
        SELECT @Message = @Message + 'Error occured in Procedure Proc_InsertRow8. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        DROP TABLE #TempTable
        RETURN
    END
END
END --End of IF @RowContent LIKE 'TableRow'
ELSE
BEGIN
    SELECT @IDRow = 0
    INSERT INTO TableDefinition8(IDTableVerDef, IDRow, Column1Val, Column2Val, Column3Val,
Column4Val, Column5Val,
Column6Val,Column7Val, Column8Val, RowContent)
VALUES(@IDTableVerDef, @IDRow,
LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),
LTRIM(RTRIM(@Column3Val)),LTRIM(RTRIM(@Column4Val)),LTRIM(RTRIM(@Column5Val)),
LTRIM(RTRIM(@Column6Val)),LTRIM(RTRIM(@Column7Val)),LTRIM(RTRIM(@Column8Val)),
LTRIM(RTRIM(@RowContent)))
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        SELECT @IDProcState = 3
```

Appendix 5: Supplier's Product Class Database (SPCD) System Code

```
        SELECT @Message = 'An error occurred while inserting the table row. '
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
        NVARCHAR) + ' '
        SELECT @Message = @Message + 'Error occurred in Procedure Proc_InsertRow8. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        DROP TABLE #TempTable
        RETURN
    END
END
END -- End of Proc_InsertRow8 Procedure.
GO
```

/*

2.21 Procedure Name: dbo.proc_InsertRow9

Database: SupplierPCD

Description:

This procedure enables creation of table specification values having 9 columns.

*/

```
CREATE Procedure Proc_InsertRow9
/* Param List */
@IDTableVerDef      BIGINT,
@ColumnValues        NVARCHAR(4000),
@CollIDMeasUnits    NVARCHAR(4000) = NULL,
@RowContent          NVARCHAR(100),

@IDProcState        TINYINT OUTPUT,
@Message             NVARCHAR(500) OUTPUT

AS
BEGIN
    DECLARE
    @Error            INT,
    @IDRow            INT,
    @NumOfRows        INT,
    @Column1Val       NVARCHAR(255),
    @Column2Val       NVARCHAR(255),
    @Column3Val       NVARCHAR(255),
    @Column4Val       NVARCHAR(255),
    @Column5Val       NVARCHAR(255),
    @Column6Val       NVARCHAR(255),
    @Column7Val       NVARCHAR(255),
    @Column8Val       NVARCHAR(255),
    @Column9Val       NVARCHAR(255)

    SELECT @IDProcState = 0
    SELECT @ERROR = 0
    SELECT @Message = ''

    SELECT @NumOfRows = COUNT(*) FROM dbo.fn_getColumnValuesTable(@ColumnValues)
    --PRINT '@NumOfRows: ' + CAST(@NumOfRows AS NVARCHAR)
    /* The function fn_getColumnValuesTable(@ColumnValues) should return two rows only */

    IF @NumOfRows != 9
    BEGIN
        SELECT @IDProcState = 1
        SELECT @Message = 'An error occurred while inserting the table row. '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Error occurred in Procedure Proc_InsertRow9. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN
    END

    /*Create a temporary table and insert the values from the table returned
    by the function fn_getColumnValuesTable */
```

Appendix 5: Supplier's Product Class Database (SPCD) System Code

```
CREATE TABLE #TempTable (ColumnID INT, ColumnValue NVARCHAR(255) )
INSERT INTO #TempTable
SELECT ColumnID,ColVal FROM dbo.fn_getColumnValuesTable(@ColumnValues)

SELECT @Column1Val = ColumnValue FROM #TempTable WHERE ColumnID = 1
SELECT @Column2Val = ColumnValue FROM #TempTable WHERE ColumnID = 2
SELECT @Column3Val = ColumnValue FROM #TempTable WHERE ColumnID = 3
SELECT @Column4Val = ColumnValue FROM #TempTable WHERE ColumnID = 4
SELECT @Column5Val = ColumnValue FROM #TempTable WHERE ColumnID = 5
SELECT @Column6Val = ColumnValue FROM #TempTable WHERE ColumnID = 6
SELECT @Column7Val = ColumnValue FROM #TempTable WHERE ColumnID = 7
SELECT @Column8Val = ColumnValue FROM #TempTable WHERE ColumnID = 8
SELECT @Column9Val = ColumnValue FROM #TempTable WHERE ColumnID = 9

/*Check whether the row contains column values or column specifications
A row can only contain column values or column specifications(headers).*/

IF @RowContent LIKE 'ColVals'
BEGIN
    /* Check for the existing rows with same IDTableVerDef to keep track of the number
of rows the IDTableVerDef has if the @RowContent = 'ColumnValues' */
    SELECT @IDRow = MAX(IDRow) FROM TableDefinition9 WHERE IDTableVerDef = @IDTableVerDef
    IF @IDRow IS NULL
        SELECT @IDRow = 1
    ELSE
        SELECT @IDRow = @IDRow + 1

    INSERT INTO TableDefinition9(IDTableVerDef, IDRow, Column1Val,
    Column2Val,Column3Val,Column4Val,
    Column5Val, Column6Val,Column7Val, Column8Val, Column9Val,RowContent)
    VALUES(@IDTableVerDef, @IDRow,
    LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),LTRIM(RTRIM(@Column3Val)),
    LTRIM(RTRIM(@Column4Val)),LTRIM(RTRIM(@Column5Val)), LTRIM(RTRIM(@Column6Val)),
    LTRIM(RTRIM(@Column7Val)),LTRIM(RTRIM(@Column8Val)), LTRIM(RTRIM(@Column9Val)),
    LTRIM(RTRIM(@RowContent)))
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        SELECT @IDProcState = 2
        SELECT @Message = 'An error occured while inserting the table row. '
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '. '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
        NVARCHAR) + '. '
        SELECT @Message = @Message + 'Error occured in Procedure Proc_InsertRow9. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        DROP TABLE #TempTable
        RETURN
    END
END --End of IF @RowContent LIKE 'TableRow'
ELSE
BEGIN
    SELECT @IDRow = 0
    INSERT INTO TableDefinition9(IDTableVerDef, IDRow, Column1Val, Column2Val, Column3Val,
    Column4Val,
    Column5Val, Column6Val,Column7Val, Column8Val,
    Column9Val, RowContent)
    VALUES(@IDTableVerDef, @IDRow,
    LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),
    LTRIM(RTRIM(@Column3Val)),LTRIM(RTRIM(@Column4Val)),LTRIM(RTRIM(@Column5Val)),
    LTRIM(RTRIM(@Column6Val)),LTRIM(RTRIM(@Column7Val)),LTRIM(RTRIM(@Column8Val)),
    LTRIM(RTRIM(@Column9Val)),LTRIM(RTRIM(@RowContent)))
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        SELECT @IDProcState = 3
        SELECT @Message = 'An error occured while inserting the table row. '
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '. '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
        NVARCHAR) + '. '
    END
END
```

Appendix 5: Supplier's Product Class Database (SPCD) System Code

```
SELECT @Message = @Message + 'Error occured in Procedure Proc_InsertRow9. '  
SELECT @Message = @Message + 'Procedure is terminated abnormally. '  
RAISERROR(@Message,1,1) WITH SETERROR  
RETURN @@ERROR
```

```
END  
ELSE  
BEGIN  
    DROP TABLE #TempTable  
    RETURN  
END
```

```
END  
END -- End of Proc_InsertRow9 Procedure.  
GO
```

2.22 Procedure Name: **dbo.proc_InsertRow10**

Database: **SupplierPCD**

Description:

This procedure enables creation of table specification values having 10 columns.

*/

```
CREATE Procedure proc_InsertRow10
```

```
/* Param List */
```

```
@IDTableVerDef          BIGINT,  
@ColumnValues           NVARCHAR(4000),  
@ColIDMeasUnits         NVARCHAR(4000) = NULL,  
@RowContent             NVARCHAR(100),  
@IDProcState            TINYINT OUTPUT,  
@Message                NVARCHAR(500) OUTPUT
```

```
AS  
BEGIN
```

```
DECLARE  
@Error                INT,  
@IDRow               INT,  
@NumOfRows           INT,  
@Column1Val          NVARCHAR(255),  
@Column2Val          NVARCHAR(255),  
@Column4Val          NVARCHAR(255),  
@Column5Val          NVARCHAR(255),  
@Column6Val          NVARCHAR(255),  
@Column7Val          NVARCHAR(255),  
@Column8Val          NVARCHAR(255),  
@Column9Val          NVARCHAR(255),  
@Column10Val         NVARCHAR(255)
```

```
SELECT @IDProcState = 0  
SELECT @ERROR = 0  
SELECT @Message = ''
```

```
SELECT @NumOfRows = COUNT(*) FROM dbo.fn_getColumnValuesTable(@ColumnValues)  
--PRINT 'NumOfRows: ' + CAST(@NumOfRows AS NVARCHAR)  
/* The function fn_getColumnValuesTable(@ColumnValues) should return two rows only */
```

```
IF @NumOfRows != 10  
BEGIN
```

```
    SELECT @IDProcState = 1  
    SELECT @Message = 'An error occured while inserting the table row. '  
    SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + '. '  
    SELECT @Message = @Message + 'Error occured in Procedure Proc_InsertRow10. '  
    SELECT @Message = @Message + 'Procedure is terminated abnormally. '  
    RAISERROR(@Message,1,1) WITH SETERROR  
    RETURN
```

```
END
```

```
/*Create a temporary table and insert the values from the table returned  
by the function fn_getColumnValuesTable */
```

```
CREATE TABLE #TempTable (ColumnID INT, ColumnValue NVARCHAR(255) )  
INSERT INTO #TempTable  
SELECT ColumnID, ColVal FROM dbo.fn_getColumnValuesTable(@ColumnValues)
```

```
SELECT @Column1Val = ColumnValue FROM #TempTable WHERE ColumnID = 1
```

Appendix 5: Supplier's Product Class Database (SPCD) System Code

```
SELECT @Column2Val = ColumnValue FROM #TempTable WHERE ColumnID = 2
SELECT @Column3Val = ColumnValue FROM #TempTable WHERE ColumnID = 3
SELECT @Column4Val = ColumnValue FROM #TempTable WHERE ColumnID = 4
SELECT @Column5Val = ColumnValue FROM #TempTable WHERE ColumnID = 5
SELECT @Column6Val = ColumnValue FROM #TempTable WHERE ColumnID = 6
SELECT @Column7Val = ColumnValue FROM #TempTable WHERE ColumnID = 7
SELECT @Column8Val = ColumnValue FROM #TempTable WHERE ColumnID = 8
SELECT @Column9Val = ColumnValue FROM #TempTable WHERE ColumnID = 9
SELECT @Column10Val = ColumnValue FROM #TempTable WHERE ColumnID = 10

/*Check whether the row contains column values or column specifications
A row can only contain column values or column specifications(headers).*/

IF @RowContent LIKE 'ColVals'
BEGIN
    /* Check for the existing rows with same IDTableVerDef to keep track of the number
of rows the IDTableVerDef has if the @RowContent = 'ColumnValues' */
    SELECT @IDRow = MAX(IDRow) FROM TableDefinition10 WHERE IDTableVerDef =
@IDTableVerDef
    IF @IDRow IS NULL
        SELECT @IDRow = 1
    ELSE
        SELECT @IDRow = @IDRow + 1

    INSERT INTO TableDefinition10(IDTableVerDef, IDRow, Column1Val,
Column2Val,Column3Val,Column4Val,
Column5Val, Column6Val,Column7Val, Column8Val, Column9Val,Column10Val, RowContent)
VALUES(@IDTableVerDef, @IDRow,
LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),
LTRIM(RTRIM(@Column3Val)),
LTRIM(RTRIM(@Column4Val)),LTRIM(RTRIM(@Column5Val)), LTRIM(RTRIM(@Column6Val)),
LTRIM(RTRIM(@Column7Val)),LTRIM(RTRIM(@Column8Val)), LTRIM(RTRIM(@Column9Val)),
LTRIM(RTRIM(@Column10Val)),LTRIM(RTRIM(@RowContent)) )
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        SELECT @IDProcState = 2
        SELECT @Message = 'An error occured while inserting the table row. '
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '. '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
NVARCHAR) + '. '
        SELECT @Message = @Message + 'Error occured in Procedure Proc_InsertRow10. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        DROP TABLE #TempTable
        RETURN
    END
END
END --End of IF @RowContent LIKE 'TableRow'
ELSE
BEGIN
    SELECT @IDRow = 0
    INSERT INTO TableDefinition10(IDTableVerDef, IDRow, Column1Val, Column2Val, Column3Val,
Column4Val,
Column5Val, Column6Val,Column7Val, Column8Val,
Column9Val, Column10Val,RowContent)
VALUES(@IDTableVerDef, @IDRow,
LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),
LTRIM(RTRIM(@Column3Val)),LTRIM(RTRIM(@Column4Val)),LTRIM(RTRIM(@Column5Val)),
LTRIM(RTRIM(@Column6Val)),LTRIM(RTRIM(@Column7Val)),LTRIM(RTRIM(@Column8Val)),
LTRIM(RTRIM(@Column9Val)),LTRIM(RTRIM(@Column10Val)),LTRIM(RTRIM(@RowContent)) )
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        SELECT @IDProcState = 3
        SELECT @Message = 'An error occured while inserting the table row. '
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '. '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
NVARCHAR) + '. '
        SELECT @Message = @Message + 'Error occured in Procedure Proc_InsertRow10. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
    END
END
```

Appendix 5: Supplier's Product Class Database (SPCD) System Code

```
                RETURN @@ERROR
            END
            ELSE
            BEGIN
                DROP TABLE #TempTable
                RETURN
            END
        END
    END
END -- End of Proc_InsertRow10 Procedure.
GO
```

2.23 Function Name: **dbo.fn_getColumnValuesTable**

Database: **SupplierPCD**

Description:

This function extracts column values from a string. Column values in a string are delimited by '*****'. After extracting the column values the function inserts each individual column value into the table and returns the table to the called procedure.

```
*/

CREATE FUNCTION dbo.fn_getColumnValuesTable
(@ColumnValues NVARCHAR(4000))
RETURNS @ColumnValuesTable TABLE
(
    ColumnID INT,
    ColVal NVARCHAR(500)
)
AS
BEGIN
    DECLARE
        @Index INT, /* Keeps the index of position from where the delimiter starts.
                     i.e. the starting position of '*****' in a string */
        @DONE TINYINT, /* Acts as a boolean variable. */
        @ColumnVal NVARCHAR(500), /* Holds each individual column value */
        @Counter INT

    /*The following is executed when the procedure is called with empty string as input parameter */
    SELECT @DONE = 0
    SELECT @ColumnValues = LTRIM(RTRIM(@ColumnValues))
    IF LEN(@ColumnValues) = 0
    BEGIN
        SELECT @DONE = 1
        RETURN
    END

    /* The following is executed when the string contains only one column values */
    SELECT @Index = CHARINDEX('***', @ColumnValues)
    IF @Index = 0
    BEGIN
        SELECT @ColumnVal = @ColumnValues
        SELECT @ColumnVal = LTRIM(RTRIM(@ColumnVal))
        SELECT @Counter = 1
        INSERT INTO @ColumnValuesTable (ColumnID,ColVal) VALUES (@Counter, @ColumnValues)
        SELECT @DONE = 1
        RETURN
    END

    /* The following loop is executed when there are more than one column values in the string */
    SELECT @Counter = 1
    WHILE @DONE = 0
    BEGIN
        SELECT @Index = CHARINDEX('***', @ColumnValues)
        SELECT @ColumnVal = LEFT(@ColumnValues, @Index - 1)
        SELECT @ColumnVal = LTRIM(RTRIM(@ColumnVal))
        IF LEN(@ColumnVal) > 0
            INSERT INTO @ColumnValuesTable (ColumnID,ColVal) VALUES (@Counter, @ColumnVal)
        ELSE
            INSERT INTO @ColumnValuesTable (ColumnID,ColVal) VALUES (@Counter, NULL)
        SELECT @Counter = @Counter + 1
        SELECT @ColumnValues = SUBSTRING(@ColumnValues, @Index + 3, LEN(@ColumnValues) - @Index + 2)
        SELECT @ColumnValues = LTRIM(RTRIM(@ColumnValues))
        SELECT @Index = CHARINDEX('***', @ColumnValues)
    END
END
```

Appendix 5: Supplier's Product Class Database (SPCD) System Code

```
IF @Index = 0
BEGIN
    IF LEN(@ColumnValues) = 0
    BEGIN
        SELECT @DONE = 1
    END
    ELSE
    BEGIN
        SELECT @ColumnVal = @ColumnValues
        SELECT @ColumnVal = LTRIM(RTRIM(@ColumnVal))
        IF LEN(@ColumnVal) > 0
            INSERT INTO @ColumnValuesTable (ColumnID,ColVal) VALUES (@Counter,
@ColumnVal)
        ELSE
            INSERT INTO @ColumnValuesTable (ColumnID,ColVal) VALUES (@Counter, NULL)
        SELECT @DONE = 1
    END
END
END
RETURN
END
```

/* 2.24 Function Name: dbo. dbo.fn_GetIdentityPart

Database: SupplierPCD

Description:

This function returns entity part from a complete ID.

```
*/
CREATE FUNCTION dbo.fn_GetIdentityPart
(@IDComplete BIGINT)
RETURNS INT
AS
BEGIN
    RETURN CAST(SUBSTRING(CAST(@IDComplete AS NVARCHAR), 1, 3) AS INT)
END
```

/*

2.25 Function Name: **dbo.fn_GetIDPart**

Database: **SupplierPCD**

Description:

This function returns ID part from a complete ID.

*/

```
CREATE FUNCTION dbo.fn_GetIDPart
(@IDComplete BIGINT)
RETURNS BIGINT
AS
BEGIN
    DECLARE
        @IDCompleteLength TINYINT,
        @IDPart            NVARCHAR(20)
    SELECT @IDCompleteLength = LEN(CAST(@IDComplete AS NVARCHAR))
    SELECT @IDPart = SUBSTRING(CAST(@IDComplete AS NVARCHAR), 4, @IDCompleteLength - 3)
    RETURN CAST(@IDPart AS BIGINT)
END
```

/*

2.26 Function Name: **dbo.fn_GetNewID**

Database: **SupplierPCD**

Description:

This function generates a new ID for given entity such as product class, specification, etc.

*/

```
CREATE FUNCTION dbo.fn_GetNewID
(@IDEntity INT)
RETURNS BIGINT
AS
BEGIN
    DECLARE
        @IDAvailable BIGINT,
        @IDNext      BIGINT
    SELECT @IDAvailable = IDAvailable FROM Entity WHERE [IDEntity] = @IDEntity
    SELECT @IDNext = dbo.fn_IncrementID(@IDAvailable, DEFAULT)
    RETURN @IDAvailable
END
```

/*

2.27 Function Name: **dbo.fn_IncrementID**

Database: **SupplierPCD**

Description:

This function increments ID.

*/

```
CREATE FUNCTION dbo.fn_IncrementID
(@IDComplete BIGINT,
@IncrementBy INT = 1)
RETURNS BIGINT
AS
BEGIN
    DECLARE
        @IDPart      BIGINT,
        @EntityPart  BIGINT
    SELECT @IDPart = dbo.fn_GetIDPart(@IDComplete)
    SELECT @EntityPart = dbo.fn_GetIDEntityPart(@IDComplete)
    SELECT @IDPart = @IDPart + @IncrementBy
    RETURN CAST(CAST(@EntityPart AS NVARCHAR) + CAST(@IDPart AS NVARCHAR) AS BIGINT)
END
```

/*
2.28 Function Name: dbo.fn_GetNextAvailableID

Database: SupplierPCD

Description:

This function generates next available ID.

*/

```
CREATE FUNCTION dbo.fn_GetNextAvailableID
(@IDComplete BIGINT)
RETURNS BIGINT
AS
BEGIN
DECLARE
@IDPart          BIGINT,
@EntityPart      BIGINT
SELECT @IDPart = dbo.GetIDPart(@IDComplete)
SELECT @EntityPart = dbo.GetIDEntityPart(@IDComplete)
SELECT @IDPart = @IDPart + 1
RETURN CAST(CAST(@EntityPart AS NVARCHAR) + CAST(@IDPart AS NVARCHAR) AS BIGINT)
END
```

Supplier Database (SD) System Code

/*

3.1 Procedure Name: dbo.proc_AssignCategory

Database: SupplierDB

Description:

This procedure enables assigning a category to super category and sub category.

*/

```

CREATE Procedure proc_AssignCategory
/* Param List */
@IDCategory          UNIQUEIDENTIFIER,
@IDSuperCategory     UNIQUEIDENTIFIER,
@IDProcState         TINYINT          OUTPUT,
@Message             NVARCHAR(500)   OUTPUT
AS
BEGIN
    DECLARE
    @Error            INT
    SELECT @IDProcState = 0
    SELECT @ERROR = 0
    INSERT INTO Category_SuperCategory (IDCategory, IDSuperCategory)
    VALUES (@IDCategory, @IDSuperCategory)
    SELECT @ERROR = @@ERROR
    IF @Error != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @IDProcState = 1
        SELECT @Message = 'An error ocured while creating the new Category. '
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Error occurred in Procedure proc_CreateCategory. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END

    INSERT INTO Category_SubCategory (IDCategory, IDSubCategory)
    VALUES (@IDSuperCategory, @IDCategory)
    SELECT @ERROR = @@ERROR
    IF @Error != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @IDProcState = 2
        SELECT @Message = 'An error ocured while creating the new category. '
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Error occurred in Procedure proc_CreateCategory. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        SELECT @Message = 'Category successfully assigned.'
        RETURN
    END
END
END -- End of Proc_AssignCategory Procedure.
GO

```

/*

3.2 Procedure Name: dbo.proc_AssignList2Product

Database: SupplierDB

Description:

This procedure enables assigning list specification to product version.

*/

```

CREATE Procedure proc_AssignList2Product
/* Param List */
@IDProd             UNIQUEIDENTIFIER,
@IDList            UNIQUEIDENTIFIER,
@IDProcState         TINYINT          OUTPUT,
@Message           NVARCHAR(500)   OUTPUT

```

```

AS
BEGIN
    DECLARE
        @Error          INT
    SELECT @IDProcState = 0
    SELECT @@ERROR = 0
    /* A List can be assigned to a product or a specification group */
    /* This procedure assigns a list to a product */
    INSERT INTO ProductDefinition(IDProd, IDList)
    VALUES (@IDProd, @IDList)
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @IDProcState = 1
        SELECT @Message = 'An error occurred while assigning the List to the product.'
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '.'
        SELECT @Message = @Message + 'Error occurred in Procedure proc_AssignList2Product.'
        SELECT @Message = @Message + 'Procedure is terminated abnormally.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        SELECT @Message = 'List successfully assigned to the Product.'
        RETURN
    END
END
-- End of proc_AssignList2Product Procedure.
GO

```

/*

3.3 Procedure Name: **dbo.proc_AssignList2SpecificationGroup**

Database: **SupplierDB**

Description:

This procedure enables assigning list specification to specification group.

*/

```

CREATE Procedure proc_AssignList2SpecificationGroup
/* Param List */
@IDSpecGroup          UNIQUEIDENTIFIER,
@IDList               UNIQUEIDENTIFIER,
@IDProcState          TINYINT          OUTPUT,
@Message              NVARCHAR(500)   OUTPUT
AS
BEGIN
    DECLARE
        @Error          INT
    /* A List can be assigned to a product or a specification group */
    /* This procedure assigns a list to a Specification Group */
    INSERT INTO SpecificationGroupDefinition(IDSpecGroup, IDList)
    VALUES (@IDSpecGroup, @IDList)
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @IDProcState = 1
        SELECT @Message = 'An error occurred while assigning the List to the specification group.'
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '.'
        SELECT @Message = @Message + 'Error occurred in Procedure proc_AssignList2SpecificationGroup.'
        SELECT @Message = @Message + 'Procedure is terminated abnormally.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        SELECT @Message = 'List successfully assigned to the Specification Group.'
        RETURN
    END
END
GO

```

/*

3.4 Procedure Name: **dbo.proc_AssignProduct2Category**

Database: **SupplierDB**

Description:

This procedure enables assigning a product to a category.

*/

```

CREATE Procedure proc_AssignProduct2Category
/* Param List */
@IDCategory      UNIQUEIDENTIFIER,
@IDProd          UNIQUEIDENTIFIER,
@IDProcState     TINYINT      OUTPUT,
@Message        NVARCHAR(500) OUTPUT
AS
BEGIN
    DECLARE
    @Error INT
    SELECT @IDProcState = 0
    SELECT @ERROR = 0
    /* A sub product can be assigned to a category,product or a specification group */
    /* This procedure assigns a product to a category. Important thing to note is that
    a product is being assigned not IDProdDef */

    INSERT INTO Category_ProductClass(IDCategory, IDProd)
    VALUES (@IDCategory, @IDProd)
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @IDProcState = 1
        SELECT @Message = 'An error occured while assigning product to the category.'
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' .'
        SELECT @Message = @Message + 'Error occured in Procedure proc_AssignProduct2Category.'
        SELECT @Message = @Message + 'Procedure is terminated abnormally.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        SELECT @Message = 'Product successfully assigned to the category.'
        RETURN
    END
END
-- End of Pproc_AssignProduct2Category.
GO

```

/*

3.5 Procedure Name: **dbo.proc_AssignProduct2SpecificationGroup**

Database: **SupplierDB**

Description:

This procedure enables assigning product to specification group.

*/

```

CREATE Procedure proc_AssignProduct2SpecificationGroup
/* Param List */
@IDSpecGroup    UNIQUEIDENTIFIER,
@IDProd         UNIQUEIDENTIFIER,
@IDProcState    TINYINT      OUTPUT,
@Message        NVARCHAR(500) OUTPUT
AS
BEGIN
    DECLARE
    @Error INT
    SELECT @IDProcState = 0
    SELECT @ERROR = 0
    /* A sub product can be assigned to a category,product class or a specification group */
    /* This stored procedure assigns a product to a specification group. */
    INSERT INTO SpecificationGroupDefinition(IDSpecGroup, IDProd )
    VALUES (@IDSpecGroup, @IDProd)

```

Appendix 5: Supplier Database (SD) System Code

```
SELECT @Error = @@ERROR
IF @Error != 0
BEGIN
    ROLLBACK TRAN
    SELECT @IDProcState = 1
    SELECT @Message = 'An error occurred while assigning the product to the specification group.'
    SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
    SELECT @Message = @Message + 'Error occurred in Procedure proc_AssignProduct2SpecificationGroup.'
    SELECT @Message = @Message + 'Procedure is terminated abnormally.'
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END
ELSE
BEGIN
    SELECT @Message = 'Product successfully assigned to the specification group.'
    RETURN
END
END
-- End of proc_AssignProduct2SpecificationGroup Procedure.
GO
```

/*

3.6 Procedure Name: dbo.proc_AssignSpecificationGroup2Product

Database: SupplierDB

Description:

This procedure enables assigning specification group to product.

*/

```
CREATE Procedure proc_AssignSpecificationGroup2Product
/* Param List */
@IDProd          UNIQUEIDENTIFIER,
@IDSpecGroup     UNIQUEIDENTIFIER,
@IDProcState     TINYINT          OUTPUT,
@Message         NVARCHAR(500)   OUTPUT
AS
BEGIN
    DECLARE
    @Error         INT
    /* A specification group can be assigned to a product or another specification group. */
    /* This procedure assigns a specification group to a product */

    INSERT INTO ProductDefinition(IDProd, IDSpecGroup)
    VALUES (@IDProd, @IDSpecGroup)
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @IDProcState = 1
        SELECT @Message = 'An error occurred while assigning the specification group to the product.'
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Error occurred in Procedure proc_AssignSpecificationGroup.'
        SELECT @Message = @Message + 'Procedure is terminated abnormally.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        SELECT @Message = 'Specification group successfully assigned to the Product.'
        RETURN
    END
END
-- End of proc_AssignSpecificationGroup2Product.
GO
```

/*

3.7 Procedure Name: **dbo.proc_AssignSpecificationGroup2SpecificationGroup**

Database: **SupplierDB**

Description:

This product enables assigning a (sub) specification group to another specification group.

*/

```

CREATE Procedure proc_AssignSpecificationGroup2SpecificationGroup
/* Param List */
@IDSpecGroup      UNIQUEIDENTIFIER,
@IDSubSpecGroup   UNIQUEIDENTIFIER,
@IDProcState      TINYINT      OUTPUT,
@Message          NVARCHAR(500) OUTPUT
AS
BEGIN
    DECLARE
    @Error          INT
    /* A specification group can be assigned to a product or another specification group. */
    /* This procedure assigns a specification group to another specification group. */

    INSERT INTO SpecificationGroupDefinition(IDSpecGroup, IDSubSpecGroup)
    VALUES (@IDSpecGroup, @IDSubSpecGroup)
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @IDProcState = 1
        SELECT @Message = 'An error ocured while assigning the specification group to specification group.'
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' .'
        SELECT @Message = @Message + 'Error occurred in Procedure
        proc_AssignSpecificationGroup2SpecificationGroup.'
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        SELECT @Message = 'Specification group successfully assigned to the Specification Group.'
        RETURN
    END
END
END -- End of proc_AssignSpecificationGroup2SpecificationGroup.
GO

```

/*

3.8 Procedure Name: **dbo.proc_AssignSubProduct2Product**

Database: **SupplierDB**

Description:

This procedure enables assigning a sub product to another product.

*/

```

CREATE Procedure proc_AssignSubProduct2Product
/* Param List */
@IDProd           UNIQUEIDENTIFIER,
@IDSubProd        UNIQUEIDENTIFIER,
@IDProcState      TINYINT      OUTPUT,
@Message          NVARCHAR(500) OUTPUT
AS
BEGIN
    DECLARE
    @Error          INT
    SELECT @IDProcState = 0
    SELECT @ERROR = 0

    /* A sub product can be assigned to a category, product or a specification group */
    /* This procedure assigns a sub product to another product */

    INSERT INTO ProductDefinition(IDProd, IDSubProd)

```

```

VALUES (@IDProd, @IDSubProd)
SELECT @Error = @@ERROR
IF @Error != 0
BEGIN
    ROLLBACK TRAN
    SELECT @IDProcState = 1
    SELECT @Message = 'An error occurred while assigning the Subproduct to the product.'
    SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ', '
    SELECT @Message = @Message + 'Error occurred in Procedure proc_AssignSubProduct2Product.'
    SELECT @Message = @Message + 'Procedure is terminated abnormally.'
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END
ELSE
BEGIN
    SELECT @Message = 'Subproduct successfully assigned to the product.'
    RETURN
END
END
-- End of proc_AssignSubProduct2Product Procedure.
GO

```

/*

3.9 Procedure Name: dbo.proc_AssignTableObject2Product

Database: SupplierDB

Description:

This procedure enables assigning a table specification object to product.

*/

```

CREATE Procedure proc_AssignTableObject2Product
/* Param List */
@IDProd          UNIQUEIDENTIFIER,
@IDTableObj      UNIQUEIDENTIFIER,
@IDProcState     TINYINT          OUTPUT,
@Message         NVARCHAR(500)    OUTPUT
AS
BEGIN
    DECLARE
    @Error INT
    SELECT @IDProcState = 0
    SELECT @ERROR = 0

    /* A Table object can be assigned to a product class or a specification group */
    /*
    This procedure assigns a table object to a product.
    */
    INSERT INTO ProductDefinition(IDProd, IDTableObj)
    VALUES (@IDProd, @IDTableObj)
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @IDProcState = 1
        SELECT @Message = 'An error occurred while assigning the table object to the product.'
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ', '
        SELECT @Message = @Message + 'Error occurred in Procedure proc_AssignTableObject2Product.'
        SELECT @Message = @Message + 'Procedure is terminated abnormally.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        SELECT @Message = 'Table object successfully assigned to the Product.'
        RETURN
    END
END
-- End of proc_AssignTableObject2Product Procedure.
GO

```

/*

3.10 Procedure Name: dbo.proc_AssignTableObject2SpecificationGroup

Database: SupplierDB

Description:

This procedure enables assigning table specification object to specification group.

*/

```

CREATE Procedure proc_AssignTableObject2SpecificationGroup
/* Param List */
@IDSpecGroup      UNIQUEIDENTIFIER,
@IDTableObj       UNIQUEIDENTIFIER,
@IDProcState      TINYINT      OUTPUT,
@Message          NVARCHAR(500) OUTPUT
AS
BEGIN
    DECLARE
    @Error INT

    SELECT @IDProcState = 0
    SELECT @ERROR = 0

    /* A Table object can be assigned to a product class or a specification group */
    /*
    This procedure assigns a table object to specification group.
    */
    INSERT INTO SpecificationGroupDefinition(IDSpecGroup, IDTableObj)
    VALUES (@IDSpecGroup, @IDTableObj)
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @IDProcState = 3
        SELECT @Message = 'An error occured while assigning the table object to the specification group.'
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '.'
        SELECT @Message = @Message + 'Error occured in Procedure
        proc_AssignTableObject2SpecificationGroup.'
        SELECT @Message = @Message + 'Procedure is terminated abnormally.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        SELECT @Message = 'Table object successfully assigned to the Specification Group.'
        RETURN
    END
END
-- End of proc_AssignTableObject2SpecificationGroup Procedure.
GO

```

/*

3.11 Procedure Name: dbo.proc_CreateCategory

Database: SupplierDB

Description:

This procedure enables creation of a category.

*/

```

CREATE Procedure proc_CreateCategory
/* Param List */
@IDCategoryClass  BIGINT,
@CategoryName     NVARCHAR(255),
@IDSuperCategory  BIGINT = NULL,
@CategoryDesc     NVARCHAR(4000) = NULL,
@IDProcState      TINYINT      OUTPUT,
@Message          NVARCHAR(500) OUTPUT
AS
BEGIN
    DECLARE
    @Error INT,
    @IDCategory UNIQUEIDENTIFIER
    SELECT @IDProcState = 0
    SELECT @ERROR = 0

    BEGIN TRAN

```

Appendix 5: Supplier Database (SD) System Code

```
/*The following block checks whether the category already exists */
IF EXISTS (SELECT IDCategoryClass FROM Category WHERE IDCategoryClass = @IDCategoryClass)
BEGIN
    ROLLBACK TRAN
    SELECT @IDProcState = 1
    SELECT @Message = 'Category already exists.'
    SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '!'
    SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + '!'
    SELECT @Message = @Message + 'Error occurred in Procedure proc_CreateCategory.'
    SELECT @Message = @Message + 'Procedure is terminated abnormally.'
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END

/* Create Guid for IDCategory. */
SELECT @IDCategory = newID()

/*Create the Category by inserting values into the Category table */

INSERT INTO Category(IDCategory, IDCategoryClass, CategoryName, CategoryDesc)
VALUES (@IDCategory, @IDCategoryClass, LTRIM(RTRIM(@CategoryName)), LTRIM(RTRIM(@CategoryDesc)))
SELECT @Error = @@ERROR
IF @Error != 0
BEGIN
    ROLLBACK TRAN
    SELECT @IDProcState = 2
    SELECT @Message = 'An error occured while creating the new category.'
    SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '!'
    SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + '!'
    SELECT @Message = @Message + 'Error occurred in Procedure proc_CreateCategory.'
    SELECT @Message = @Message + 'Procedure is terminated abnormally.'
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END

/* Now insert values into the CategoryHierarchy table to maintain category hierarchy. */

/*IF @SuperCategory is null then a category is top level category and the IDSuperCategory
value for the category is 0. It also has no sub category. */
IF @IDSuperCategory IS NULL
BEGIN
    SELECT @Message = 'Category successfully Created.'
    SELECT @Message = @Message + 'Category ID is: ' + CAST(@IDCategory AS NVARCHAR) + '!'
    COMMIT TRAN
    RETURN
END
ELSE
BEGIN
    /* If a category has a super category then the category is also a sub category
of that super category. In this case two inserts are required. First to create a
category and its super category and second to create a supercategory and its sub category
For this we call proc_AssignCategory. */

    EXEC proc_AssignCategory
    @IDCategory      = @IDCategory,
    @IDSuperCategory = @IDSuperCategory,
    @IDProcState     = @IDProcState OUTPUT,
    @Message         = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from Proc_AssignCategory.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        SELECT @Message = 'Category successfully Created.'
        SELECT @Message = @Message + 'Category ID is: ' + CAST(@IDCategory AS NVARCHAR)
        + '!'

        COMMIT TRAN
        RETURN
    END
END
```

Appendix 5: Supplier Database (SD) System Code

```
END
END -- End of Proc_CreateCategory Procedure.
GO
```

3.12 Procedure Name: **dbo.proc_CreateNewList**

Database: **SupplierDB**

Description:

This procedure creates a new list specification.

```
*/
```

```
CREATE Procedure proc_CreateNewList
```

```
/* Param List */
```

```
@IDListClass          BIGINT,
@IDListVer            MONEY,
@ListName             NVARCHAR(255),
@ListDesc            NVARCHAR(4000) = NULL,
@ListValues          NVARCHAR(4000) = NULL,
@ListIDMeasUnits     NVARCHAR(4000) = NULL,
@AssignTo            NVARCHAR(60),
@IDAssignTo          UNIQUEIDENTIFIER,
@IDProcState         TINYINT OUTPUT,
@Message             NVARCHAR(500) OUTPUT
```

```
AS
BEGIN
```

```
DECLARE
@IDList          UNIQUEIDENTIFIER,
@Error          INT
SELECT @IDProcState = 0
SELECT @ERROR = 0
```

```
BEGIN TRAN
```

```
/*
```

```
Create List specification by insertinv values into ListSpecification Table. */
```

```
/*Create guids for IDList*/
```

```
SELECT @IDList= newID()
INSERT INTO ListSpecification(IDList, IDListClass, IDListVer, ListName,ListDesc)
VALUES (@IDList, @IDListClass, @IDListVer, LTRIM(RTRIM(@ListName)), LTRIM(RTRIM(@ListDesc)) )
```

```
SELECT @Error = @@ERROR
IF @Error != 0
BEGIN
```

```
ROLLBACK TRAN
SELECT @IDProcState = 1
SELECT @Message = 'An error ocured while creating the new list .'
SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' .'
SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + ' .'
SELECT @Message = @Message + 'Error occurred in Procedure proc_CreateNewList. '
SELECT @Message = @Message + 'Procedure is terminated abnormally. '
RAISERROR(@Message,1,1) WITH SETERROR
RETURN @@ERROR
```

```
END
```

```
/* Now assign the list specification. The following procedure is called for
assigning the list specification. A list specification can be assigned
to a product or specification group. */
```

```
IF @AssignTo = 'Product'
```

```
BEGIN
```

```
EXEC proc_AssignList2Product
@IDProd          = @IDAssignTo,
@IDList          = @IDList,
@IDProcState     = @IDProcState OUTPUT,
@Message         = @Message OUTPUT
IF @IDProcState != 0
BEGIN
```

```
ROLLBACK TRAN
SELECT @Message = @Message + ' This Procedure was called from Proc_CreateNewList.'
RAISERROR(@Message,1,1) WITH SETERROR
RETURN @@ERROR
```

```
END
```

```
END
```

Appendix 5: Supplier Database (SD) System Code

```
IF @AssignTo = 'SpecificationGroup'
BEGIN
    EXEC proc_AssignList2SpecificationGroup
        @IDSpecGroup = @IDAssignTo,
        @IDList = @IDList,
        @IDProcState = @IDProcState OUTPUT,
        @Message = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from Proc_CreateNewList.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
END

/* Now insert values in the ListDefiniton Table. For this we call proc_InsertListValues
procedure */
EXEC dbo.proc_InsertListValues
    @IDList = @IDList,
    @ListValues = @ListValues,
    @ListIDMeasUnits = @ListIDMeasUnits,
    @Message = @Message OUTPUT,
    @IDProcState = @IDProcState OUTPUT
IF @IDProcState != 0
BEGIN
    ROLLBACK TRAN
    SELECT @Message = @Message + ' This Procedure was called from proc_CreateNewList.'
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END
ELSE
BEGIN
    /*If everything goes well until this point it means that all the values are inserted
    properly into the above tables. Now we need to commit the transaction. */
    COMMIT TRAN
    SELECT @Message = 'List Created Successfully.'
    RETURN
END
END -- End of proc_CreateNewList Procedure.
GO
```

3.13 Procedure Name: dbo.proc_CreateNewProduct

Database: SupplierDB

Description:

This procedure creates a new product specification.

*/

```
CREATE Procedure proc_CreateNewProduct
/* Param List */
@IDProdClass BIGINT,
@IDProdClassVer MONEY,
@ProdName NVARCHAR(255),
@ProdDesc NVARCHAR(4000) = NULL,
@AssignTo NVARCHAR(60) = NULL,
@IDAssignTo UNIQUEIDENTIFIER = NULL,
@IDProcState TINYINT OUTPUT,
@Message NVARCHAR(500) OUTPUT
AS
BEGIN
    DECLARE
    @Error INT,
    @IDProdInternal BIGINT,
    @IDProd UNIQUEIDENTIFIER

    SELECT @IDProcState = 0
    SELECT @ERROR = 0

    BEGIN TRAN

    /*Get an new Internal ProductID */
    EXEC dbo.proc_GetNewID
```

Appendix 5: Supplier Database (SD) System Code

```
@IDEntity = 114,
@IDNew = @IDProdInternal OUTPUT,
@Message = @Message OUTPUT,
@IDProcState = @IDProcState OUTPUT

IF @IDProcState != 0
BEGIN
    ROLLBACK TRAN
    SELECT @Message = @Message + ' This Procedure was called from Proc_CreateNewProduct.'
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END

/*
Create product by inserting values into product table. After inserting values into the
product table, the product created should be assigned. A product can be assigned to
a another product, a specification group or a categor.
*/

/*Create guides for IDProd */
SELECT @IDProd = newID()

INSERT INTO Product(IDProd, IDProdInternal, IDProdClass, IDProdClassVer, ProdName, ProdDesc)
VALUES (@IDProd, @IDProdInternal, @IDProdClass, @IDProdClassVer, LTRIM(RTRIM(@ProdName)),
LTRIM(RTRIM(@ProdDesc)))

SELECT @Error = @@ERROR
IF @Error != 0
BEGIN
    ROLLBACK TRAN
    SELECT @IDProcState = 1
    SELECT @Message = 'An error occured while creating the new product class.'
    SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
    SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + ' '
    SELECT @Message = @Message + 'Error occured in Procedure proc_CreateNewProduct. '
    SELECT @Message = @Message + 'Procedure is terminated abnormally. '
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END

/* A product can be assigned to another product, specification group or category */
/* Assigning product to another product*/

IF @AssignTo = 'Product'
BEGIN
    EXEC proc_AssignSubProduct2Product
    @IDProd = @IDAssignTo,
    @IDSubProd = @IDProd,
    @IDProcState = @IDProcState OUTPUT,
    @Message = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from proc_CreateNewProduct.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        /*If everything goes well until this point it means that all the values are inserted
        properly into the above tables. Now we need to commit the transaction. */

        COMMIT TRAN
        SELECT @Message = 'Product Created Successfully.'
        RETURN
    END
END
END -- End of IF @AssignTo = 'product'

IF @AssignTo = 'SpecificationGroup'
BEGIN
    EXEC proc_AssignProduct2SpecificationGroup
    @IDSpecGroup = @IDAssignTo,
    @IDProd = @IDProd,
    @IDProcState = @IDProcState OUTPUT,
    @Message = @Message OUTPUT
    IF @IDProcState != 0
```

Appendix 5: Supplier Database (SD) System Code

```
BEGIN
    ROLLBACK TRAN
    SELECT @Message = @Message + ' This Procedure was called from proc_CreateNewProduct.'
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END
ELSE
BEGIN
    /*If everything goes well until this point it means that all the values are inserted
    properly into the above tables. Now we need to commit the transaction. */

    COMMIT TRAN
    SELECT @Message = 'Product Created Successfully.'
    RETURN
END
END -- IF @AssignTo = 'SpecificationGroup'

IF @AssignTo = 'Category'
BEGIN
    EXEC proc_AssignProduct2Category
    @IDCategory = @IDAssignTo,
    @IDProd = @IDProd,
    @IDProcState = @IDProcState OUTPUT,
    @Message = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from proc_CreateNewProduct.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        /*If everything goes well until this point it means that all the values are inserted
        properly into the above tables. Now we need to commit the transaction. */

        COMMIT TRAN
        SELECT @Message = 'Product Created Successfully.'
        RETURN
    END
END
END -- IF @AssignTo = 'Category'
END -- End of Proc_CreateNewProduct Procedure.
GO
```

3.14 Procedure Name: **dbo.proc_CreateNewProductSpecificationObject**

Database: **SupplierDB**

Description:

This procedure enables creation of a new product specification.

*/

```
CREATE Procedure proc_CreateNewProductSpecificationObject
```

```
/* Param List */
```

```
@ProdSpecName NVARCHAR(255),
@ProdSpecValue NVARCHAR(4000),
@ProdSpecDesc NVARCHAR(4000) = NULL,
@MeasUnitName NVARCHAR(255),
@IDProd UNIQUEIDENTIFIER,
@IDProcState TINYINT OUTPUT,
@Message NVARCHAR(500) OUTPUT
```

```
AS
```

```
BEGIN
```

```
    DECLARE
    @Error INT,
    @IDProdSpec UNIQUEIDENTIFIER
    SELECT @IDProcState = 0
    SELECT @ERROR = 0
```

```
    BEGIN TRAN
```

```
    /*Create guid for IDProdSpec */
```

```
    SELECT @IDProdSpec = newID()
```

```
    /*Create Product Specification by inserting values into the Product Specification table */
```

Appendix 5: Supplier Database (SD) System Code

```
INSERT INTO ProductSpecification(IDProdSpec, ProdSpecName, ProdSpecValue,
ProdSpecDesc, MeasUnitName, IDProd)
VALUES
(@IDProdSpec, LTRIM(RTRIM(@ProdSpecName)),LTRIM(RTRIM(@ProdSpecValue)),
LTRIM(RTRIM(@ProdSpecDesc)), LTRIM(RTRIM(@MeasUnitName)), @IDProd)

SELECT @Error = @@ERROR
IF @Error != 0
BEGIN
    ROLLBACK TRAN
    SELECT @IDProcState = 1
    SELECT @Message = 'An error occured while creating the new Product Specification object.'
    SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
    SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + ' '
    SELECT @Message = @Message + 'Error occured in Procedure
Proc_CreateNewProductSpecificationObject. '
    SELECT @Message = @Message + 'Procedure is terminated abnormally. '
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END
ELSE
BEGIN
    COMMIT TRAN
END
END -- End of Proc_CreateNewProductSpecificationObject Procedure.
GO
```

3.15 Procedure Name: dbo.proc_CreateNewSGSpecificationObject

Database: SupplierDB

Description:

This procedure enables the creation of new specification group specification.

```
*/
CREATE Procedure proc_CreateNewSGSpecificationObject
/* Param List */
@SGSpecName NVARCHAR(255),
@SGSpecValue NVARCHAR(4000),
@SGSpecDesc NVARCHAR(4000) = NULL,
@MeasUnitName NVARCHAR(255),
@IDSpecGroup UNIQUEIDENTIFIER,
@IDProcState TINYINT OUTPUT,
@Message NVARCHAR(500) OUTPUT
AS
BEGIN
    DECLARE
    @Error INT,
    @IDSGSpec UNIQUEIDENTIFIER
    SELECT @IDProcState = 0
    SELECT @ERROR = 0

    BEGIN TRAN

    /*Create guid for IDProdSpec */
    SELECT @IDSGSpec = newID()

    /*Create Product Specification by inserting values into the Product Specification table */
    INSERT INTO SGSpecification(IDSGSpec, SGSpecName, SGSpecValue,
SGSpecDesc, MeasUnitName, IDSpecGroup)
VALUES
(@IDSGSpec, LTRIM(RTRIM(@SGSpecName)),LTRIM(RTRIM(@SGSpecValue)),
LTRIM(RTRIM(@SGSpecDesc)), LTRIM(RTRIM(@MeasUnitName)), @IDSpecGroup)

    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @IDProcState = 1
        SELECT @Message = 'An error occured while creating the new Specification Group Specification object.'
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Error occured in Procedure Proc_CreateNewSGSpecificationObject. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
    END
```

Appendix 5: Supplier Database (SD) System Code

```
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        COMMIT TRAN
    END
END -- End of Proc_CreateNewSGSpecificationObject Procedure.
GO
```

3.16 Procedure Name: **dbo.proc_CreateNewSpecificationGroup**

Database: **SupplierDB**

Description:

This procedure enables creation of new specification group.

*/

```
CREATE Procedure proc_CreateNewSpecificationGroup
```

```
/* Param List */
```

```
@IDSpecGroupClass      BIGINT,
@IDSpecGroupVer        MONEY,
@SpecGroupName         NVARCHAR(255),
@SpecGroupDesc         NVARCHAR(4000),
@AssignTo              NVARCHAR(60),
@IDAssignTo            UNIQUEIDENTIFIER,
@IDProcState           TINYINT OUTPUT,
@Message               NVARCHAR(500) OUTPUT
```

```
AS
```

```
BEGIN
```

```
    BEGIN TRAN
    DECLARE
    @IDSpecGroup          UNIQUEIDENTIFIER,
    @Error                INT,
    @IDEntityPart        INT
```

```
    SELECT @IDProcState = 0
    SELECT @ERROR = 0
```

```
    /*Create guides for @IDSpecGroup */
    SELECT @IDSpecGroup = newID()
```

```
    /*Create a specification group by inserting values into SpecificationGroup table */
```

```
    INSERT INTO SpecificationGroup (IDSpecGroup, IDSpecGroupClass, IDSpecGroupVer, SpecGroupName,
SpecGroupDesc)
VALUES (@IDSpecGroup, @IDSpecGroupClass, @IDSpecGroupVer, LTRIM(RTRIM(@SpecGroupName)),
LTRIM(RTRIM(@SpecGroupDesc)) )
```

```
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
```

```
        ROLLBACK TRAN
        SELECT @IDProcState = 1
        SELECT @Message = 'An error occured while creating the new specification group.'
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Error occured in Procedure proc_CreateNewSpecificationGroup.'
        SELECT @Message = @Message + 'Procedure is terminated abnormally.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
```

```
    /* Now assign the specification group. The following procedure is called for
    assigning the specification group to a product.*/
```

```
    IF @AssignTo = 'Product'
    BEGIN
```

```
        EXEC proc_AssignSpecificationGroup2Product
        @IDProd           = @IDAssignTo,
        @IDSpecGroup     = @IDSpecGroup,
        @IDProcState     = @IDProcState OUTPUT,
        @Message         = @Message OUTPUT
        IF @IDProcState != 0
        BEGIN
```

```
            ROLLBACK TRAN
```


Appendix 5: Supplier Database (SD) System Code

```
SELECT @Message = @Message + ' This Procedure was called from
Proc_CreateNewSpecificationGroup.'
RAISERROR(@Message,1,1) WITH SETERROR
RETURN @@ERROR
END
ELSE
BEGIN
    COMMIT TRAN
    SELECT @Message = 'Specification group created Successfully.'
    RETURN
END
END -- IF @AssignTo = 'Product'
/* The following procedure is called for
assigning the specification group to another specification group.*/

IF @AssignTo = 'SpecificationGroup'
BEGIN
    EXEC proc_AssignSpecificationGroup2SpecificationGroup
    @IDSpecGroup      = @IDAssignTo,
    @IDSubSpecGroup   = @IDSpecGroup,
    @IDProcState      = @IDProcState OUTPUT,
    @Message          = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from
Proc_CreateNewSpecificationGroup.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        COMMIT TRAN
        SELECT @Message = 'Specification group created Successfully.'
        RETURN
    END
END
END -- IF @AssignTo = 'SpecificationGroup'
END -- End of Proc_CreateNewSpecificationGroup Procedure.
GO
```

3.17 Procedure Name: dbo.proc_CreateNewTableObject

Database: SupplierDB

Description:

This procedure enable creation of new table specification.

*/

```
CREATE Procedure proc_CreateNewTableObject
/* Param List */
@IDTableSpecClass    BIGINT,
@IDTableSpecVer      MONEY,
@NumOfRows           INT      = NULL,
@NumOfColumns        INT,
@ColumnValues         NVARCHAR(4000),
@CollDMeasUnits      NVARCHAR(4000) = NULL,
@RowContent          NVARCHAR(100),
@AssignTo            NVARCHAR(60),
@IDAssignTo          UNIQUEIDENTIFIER,
@TableObjName        NVARCHAR(255),
@TableObjDesc        NVARCHAR(4000),
@IDProcState         TINYINT OUTPUT,
@Message             NVARCHAR(500) OUTPUT

AS
BEGIN
    DECLARE
    @IDTableObj        UNIQUEIDENTIFIER,
    @Error            INT
    SELECT @IDProcState = 0
    SELECT @ERROR = 0
    BEGIN TRAN
    /*
    Create a table specification by inserting values into TableObj table.
```

Appendix 5: Supplier Database (SD) System Code

```
*/
/*Create guides for @IDTableObj, @IDTableObj */
SELECT @IDTableObj = newID()

INSERT INTO TableObject(IDTableObj, IDTableSpecClass, IDTableSpecVer,
                        NumOfRows, NumOfColumns, TableObjName, TableObjDesc)
VALUES (@IDTableObj, @IDTableSpecClass, @IDTableSpecVer,
        @NumOfRows, @NumOfColumns, LTRIM(RTRIM(@TableObjName)), LTRIM(RTRIM(@TableObjDesc)))

SELECT @Error = @@ERROR
IF @Error != 0
BEGIN
    ROLLBACK TRAN
    SELECT @IDProcState = 1
    SELECT @Message = 'An error occured while creating the new table Object. '
    SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
    SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + ' '
    SELECT @Message = @Message + 'Error occured in Procedure proc_CreateNewTableObject. '
    SELECT @Message = @Message + 'Procedure is terminated abnormally. '
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END

/* Assign the table specification by calling the following procedure */
/* A Table specification can be assigned to a product class or a specification group */
IF @AssignTo = 'Product'
BEGIN
    EXEC proc_AssignTableObject2Product
        @IDProd = @IDAssignTo,
        @IDTableObj = @IDTableObj,
        @IDProcState = @IDProcState OUTPUT,
        @Message = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from
proc_CreateNewTableObject.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
END

IF @AssignTo = 'SpecificationGroup'
BEGIN
    EXEC proc_AssignProduct2SpecificationGroup
        @IDSpecGroup = @IDAssignTo,
        @IDTableObj = @IDTableObj,
        @IDProcState = @IDProcState OUTPUT,
        @Message = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from
proc_CreateNewTableObject.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
END

/* Now check how many columns are supplied by the user. Accordingly invoke the procedure
that handle that many columns */

IF @NumOfColumns = 2
BEGIN
    EXEC proc_InsertRow2
        @IDTableObj = @IDTableObj,
        @ColumnValues = @ColumnValues,
        @ColIDMeasUnits = @ColIDMeasUnits,
        @RowContent = @RowContent,
        @IDProcState = @IDProcState OUTPUT,
        @Message = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from
proc_CreateNewTableObject.'
```

Appendix 5: Supplier Database (SD) System Code

```

        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        COMMIT TRAN
        SELECT @Message = 'Table Specification Created Successfully.'
        RETURN
    END
END

IF @NumOFColumns = 3
BEGIN
    EXEC proc_InsertRow3
    @IDTableObj = @IDTableObj,
    @ColumnValues = @ColumnValues,
    @ColIDMeasUnits = @ColIDMeasUnits,
    @RowContent = @RowContent,
    @IDProcState = @IDProcState OUTPUT,
    @Message = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from
proc_CreateNewTableObject.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        COMMIT TRAN
        SELECT @Message = 'Table Specification Created Successfully.'
        RETURN
    END
END

IF @NumOFColumns = 4
BEGIN
    EXEC proc_InsertRow4
    @IDTableObj = @IDTableObj,
    @ColumnValues = @ColumnValues,
    @ColIDMeasUnits = @ColIDMeasUnits,
    @RowContent = @RowContent,
    @IDProcState = @IDProcState OUTPUT,
    @Message = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from
proc_CreateNewTableObject.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        COMMIT TRAN
        SELECT @Message = 'Table Specification Created Successfully.'
        RETURN
    END
END

IF @NumOFColumns = 5
BEGIN
    EXEC proc_InsertRow5
    @IDTableObj = @IDTableObj,
    @ColumnValues = @ColumnValues,
    @ColIDMeasUnits = @ColIDMeasUnits,
    @RowContent = @RowContent,
    @IDProcState = @IDProcState OUTPUT,
    @Message = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from
proc_CreateNewTableObject.'

```

Appendix 5: Supplier Database (SD) System Code

```
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        COMMIT TRAN
        SELECT @Message = 'Table Specification Created Successfully. '
        RETURN
    END
END

IF @NumOfColumns = 6
BEGIN
    EXEC proc_InsertRow6
    @IDTableObj = @IDTableObj,
    @ColumnValues = @ColumnValues,
    @ColIDMeasUnits = @ColIDMeasUnits,
    @RowContent = @RowContent,
    @IDProcState = @IDProcState OUTPUT,
    @Message = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from
proc_CreateNewTableObject.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        COMMIT TRAN
        SELECT @Message = 'Table Specification Created Successfully. '
        RETURN
    END
END

IF @NumOfColumns = 7
BEGIN
    EXEC proc_InsertRow7
    @IDTableObj = @IDTableObj,
    @ColumnValues = @ColumnValues,
    @ColIDMeasUnits = @ColIDMeasUnits,
    @RowContent = @RowContent,
    @IDProcState = @IDProcState OUTPUT,
    @Message = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from
proc_CreateNewTableObject.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        COMMIT TRAN
        SELECT @Message = 'Table Specification Created Successfully. '
        RETURN
    END
END

IF @NumOfColumns = 8
BEGIN
    EXEC proc_InsertRow8
    @IDTableObj = @IDTableObj,
    @ColumnValues = @ColumnValues,
    @ColIDMeasUnits = @ColIDMeasUnits,
    @RowContent = @RowContent,
    @IDProcState = @IDProcState OUTPUT,
    @Message = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from
proc_CreateNewTableObject.'
```

Appendix 5: Supplier Database (SD) System Code

```
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        COMMIT TRAN
        SELECT @Message = 'Table Specification Created Successfully. '
        RETURN
    END
END
END

IF @NumOfColumns = 9
BEGIN
    EXEC proc_InsertRow9
    @IDTableObj = @IDTableObj,
    @ColumnValues = @ColumnValues,
    @ColIDMeasUnits = @ColIDMeasUnits,
    @RowContent = @RowContent,
    @IDProcState = @IDProcState OUTPUT,
    @Message = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from
proc_CreateNewTableObject.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        COMMIT TRAN
        SELECT @Message = 'Table Specification Created Successfully. '
        RETURN
    END
END
END

IF @NumOfColumns = 10
BEGIN
    EXEC proc_InsertRow10
    @IDTableObj = @IDTableObj,
    @ColumnValues = @ColumnValues,
    @ColIDMeasUnits = @ColIDMeasUnits,
    @RowContent = @RowContent,
    @IDProcState = @IDProcState OUTPUT,
    @Message = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from
proc_CreateNewTableObject.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        COMMIT TRAN
        SELECT @Message = 'Table Specification Created Successfully. '
        RETURN
    END
END
END
END -- End of proc_CreateNewTableObject Procedure.
GO
```

/*

3.18 Procedure Name: dbo.proc_GetNewID

Database: SupplierDB

Description:

This procedure enables generation of new IDs for entities.

*/

```

CREATE PROCEDURE dbo.proc_GetNewID
@IDEntity      INT,
@IDNew         BIGINT OUTPUT,
@Message       NVARCHAR(500) OUTPUT,
@IDProcState   TINYINT OUTPUT
AS
BEGIN
    DECLARE
    @RowsAffected INT

    SELECT @IDProcState = 0
    SELECT @Message = 'Procuedure Executed Successfully.'
    IF NOT EXISTS (SELECT [IDEntity] FROM Entity WHERE [IDEntity] = @IDEntity)
    BEGIN
        SELECT @IDProcState = 1
        SELECT @Message = 'An Entity with supplied IDEntity ' + CAST(@IDEntity AS NVARCHAR) + ' does
        not exist. '
        SELECT @Message = @Message + 'Error occured in Procedure proc_getNewID. '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + '.'
        SELECT @Message = @Message + 'Procedure is abnormally terminated.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        SELECT @IDNew = IDAvailable FROM ENTITY WHERE [IDEntity] = @IDENTITY
    END

    /* Call fn_IncrementID function to increment the IDNew. @IDNew parameter is the ID to be incremented
    and DEFAULT parameter is the number by which it is to be incremented. The DEFAULT increment
    number set is 1 */

    UPDATE Entity SET IDAvailable = dbo.fn_IncrementID(@IDNew, DEFAULT)
    WHERE [IDEntity] = @IDEntity
    SELECT @RowsAffected = @@RowCount

    --PRINT 'Rows affected are ' + CAST(@RowsAffected AS NVARCHAR)
    --PRINT '@Error id is: ' + CAST(@Error AS NVARCHAR)
    /* When the above update fails the following error is raised.*/
    IF @RowsAffected = 0
    BEGIN
        SELECT @IDProcState = 2
        SELECT @Message = 'An unknown error occured in Procedure Proc_GetNewID. '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + '.'
        SELECT @Message = @Message + 'Please contact your system administrator. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
END
GO

```

/*

3.19 Procedure Name: dbo.proc_InsertListValues

Database: SupplierDB

Description:

This procedure enables creation of values for list specification.

*/

```

CREATE Procedure proc_InsertListValues
/* Param List */
@IDList          UNIQUEIDENTIFIER,
@ListValues      NVARCHAR(4000),
@ListIDMeasUnits NVARCHAR(4000) = NULL,
@IDProcState     TINYINT OUTPUT,
@Message         NVARCHAR(500) OUTPUT

AS
BEGIN
    DECLARE
    @Error          INT
    SELECT @IDProcState = 0
    SELECT @ERROR = 0
    SELECT @Message = ''

    /*Create a temporary table and insert the values from the table returned
    by the function fn_getListValuesTable */

    CREATE TABLE #TempTableListValues (ListID INT, ListValue NVARCHAR(255) )
    INSERT INTO #TempTableListValues
    SELECT ColumnID,ColVal FROM dbo.fn_getColumnValuesTable(@ListValues)

    /*Create another temporary table and insert IDMeas units from the table returned
    by the function fn_getListValuesTable */
    /*
    ///The following commented because of an error on 07 Decd 06 with IDMeasUnit.
    CREATE TABLE #TempTableIDMeasUnit (MeasID INT, IDMeasUnit BIGINT )

    INSERT INTO #TempTableIDMeasUnit
    SELECT ColumnID, CAST(ColVal AS BIGINT) FROM dbo.fn_getColumnValuesTable(@ListIDMeasUnits)
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        SELECT @IDProcState = 1
        SELECT @Message = 'An error occured while inserting the list specification. '
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Error occured in Procedure proc_InsertListValues. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    */
    /* Now insert the values into the List Definition table */

    /*
    //The following commented because of an error on 07 Decd 06 with IDMeasUnit.
    INSERT INTO ListDefinition (IDList, ListValue, IDMeasUnit)
    SELECT @IDList, a.ListValue, b.IDMeasUnit
    FROM #TempTableListValues a, #TempTableIDMeasUnit b
    WHERE a.ListID = b.MeasID
    */

    INSERT INTO ListDefinition (IDList, ListValue, IDMeasUnit)
    SELECT @IDList, a.ListValue, NULL
    FROM #TempTableListValues a

    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        SELECT @IDProcState = 2
        SELECT @Message = 'An error occured while inserting the list specification. '
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Error occured in Procedure proc_InsertListValues. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
    END
    */

```

Appendix 5: Supplier Database (SD) System Code

```
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        DROP TABLE #TempTableListValues
        /*
        //The following commented because of an error on 07 Dec 06 with IDMeasUnit.
        DROP TABLE #TempTableIDMeasUnit
        */
        RETURN
    END
END -- End of proc_InsertListValues Procedure.
GO
```

```
/*
3.20 Procedure Name: dbo.proc_InsertRow2
Database: SupplierDB
Description:
This procedure enables creation of column values for table specification with 2 columns.
*/
```

```
CREATE Procedure proc_InsertRow2
/* Param List */
@IDTableObj      UNIQUEIDENTIFIER,
@ColumnValues     NVARCHAR(4000),
@ColIDMeasUnits  NVARCHAR(4000) = NULL,
@RowContent      NVARCHAR(100),
@IDProcState     TINYINT OUTPUT,
@Message         NVARCHAR(500) OUTPUT

AS
BEGIN

    DECLARE
    @Error         INT,
    @IDRow         INT,
    @NumOfRows     INT,
    @Column1Val    NVARCHAR(255),
    @Column2Val    NVARCHAR(255)

    SELECT @IDProcState = 0
    SELECT @ERROR = 0
    SELECT @Message = ''

    SELECT @NumOfRows = COUNT(*) FROM dbo.fn_getColumnValuesTable(@ColumnValues)
    PRINT '@NumOfRows: ' + CAST(@NumOfRows AS NVARCHAR)
    /* The function fn_getColumnValuesTable(@ColumnValues) should return two rows only */

    IF @NumOfRows != 2
    BEGIN
        SELECT @IDProcState = 1
        SELECT @Message = 'An error occurred while inserting the table row. '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Error occurred in Procedure proc_InsertRow2. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN
    END

    /*Create a temporary table and insert the values from the table returned
    by the function fn_getColumnValuesTable */

    CREATE TABLE #TempTable (ColumnID INT, ColumnValue NVARCHAR(255) )
    INSERT INTO #TempTable
    SELECT ColumnID,ColVal FROM dbo.fn_getColumnValuesTable(@ColumnValues)

    SELECT @Column1Val = ColumnValue FROM #TempTable WHERE ColumnID = 1
    SELECT @Column2Val = ColumnValue FROM #TempTable WHERE ColumnID = 2

    /*Check whether the row is contains column values or column specifications
    A row can contain column values or column specifications(headers).*/

    IF @RowContent LIKE 'ColVals'
    BEGIN
```

Appendix 5: Supplier Database (SD) System Code

```
/* Check for the existing rows with same IDTableObj to keep track of the number
of rows the IDTableObj has if the @RowContent = 'Column Values' */
SELECT @IDRow = MAX(IDRow) FROM TableDefinition2 WHERE IDTableObj = @IDTableObj
IF @IDRow IS NULL
    SELECT @IDRow = 1
ELSE
    SELECT @IDRow = @IDRow + 1

INSERT INTO TableDefinition2(IDTableObj, IDRow, Column1Val, Column2Val, RowContent)
VALUES(@IDTableObj, @IDRow, LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),
LTRIM(RTRIM(@RowContent)) )
SELECT @Error = @@ERROR
IF @Error != 0
BEGIN
    SELECT @IDProcState = 2
    SELECT @Message = 'An error occured while inserting the table row. '
    SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '. '
    SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
    NVARCHAR) + '. '
    SELECT @Message = @Message + 'Error occured in Procedure proc_InsertRow2. '
    SELECT @Message = @Message + 'Procedure is terminated abnormally. '
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END
ELSE
BEGIN
    DROP TABLE #TempTable
    RETURN
END
END --End of IF @RowContent LIKE 'TableRow'
ELSE -- Enter header information of the table.
BEGIN
    SELECT @IDRow = 0
    INSERT INTO TableDefinition2(IDTableObj, IDRow, Column1Val, Column2Val, RowContent)
    VALUES(@IDTableObj, @IDRow, LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),
    LTRIM(RTRIM(@RowContent)) )
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        SELECT @IDProcState = 3
        SELECT @Message = 'An error occured while inserting the table row. '
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '. '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
        NVARCHAR) + '. '
        SELECT @Message = @Message + 'Error occured in Procedure proc_InsertRow2. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        DROP TABLE #TempTable
        RETURN
    END
END
END -- End of Proc_InsertRow2 Procedure.
GO
```

/*

3.21 Procedure Name: dbo.proc_InsertRow3

Database: SupplierDB

Description:

This procedure enables creation of column values for table specification with 3 columns.

*/

```

CREATE Procedure Proc_InsertRow3
/* Param List */
@IDTableObj          UNIQUEIDENTIFIER,
@ColumnValues        NVARCHAR(4000),
@ColIDMeasUnits     NVARCHAR(4000) = NULL,
@RowContent         NVARCHAR(100),
@IDProcState        TINYINT OUTPUT,
@Message            NVARCHAR(500) OUTPUT

AS
BEGIN
    DECLARE
    @Error            INT,
    @IDRow           INT,
    @NumOfRows       INT,
    @Column1Val      NVARCHAR(255),
    @Column2Val      NVARCHAR(255),
    @Column3Val      NVARCHAR(255)

    SELECT @IDProcState = 0
    SELECT @ERROR = 0
    SELECT @Message = ''

    SELECT @NumOfRows = COUNT(*) FROM dbo.fn_getColumnValuesTable(@ColumnValues)
    PRINT '@NumOfRows: ' + CAST(@NumOfRows AS NVARCHAR)

    /* The function fn_getColumnValuesTable(@ColumnValues) should return three rows only */

    IF @NumOfRows != 3
    BEGIN
        SELECT @IDProcState = 1
        SELECT @Message = 'An error occured while inserting the table row. '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Error occured in Procedure proc_InsertRow3. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN
    END

    /*Create a temporary table and insert the values from the table returned
    by the function fn_getColumnValuesTable */

    CREATE TABLE #TempTable (ColumnID INT, ColumnValue NVARCHAR(255) )
    INSERT INTO #TempTable
    SELECT ColumnID,ColVal FROM dbo.fn_getColumnValuesTable(@ColumnValues)

    SELECT @Column1Val = ColumnValue FROM #TempTable WHERE ColumnID = 1
    SELECT @Column2Val = ColumnValue FROM #TempTable WHERE ColumnID = 2
    SELECT @Column3Val = ColumnValue FROM #TempTable WHERE ColumnID = 3

    /*Check whether the row is contains column values or column specifications
    A row can contain column values or column specifications(headers).*/

    IF @RowContent LIKE 'ColVals'
    BEGIN

        /* Check for the existing rows with same IDTableObj to keep track of the number
        of rows the IDTableObj has if the @RowContent = 'ColumnValues' */

        SELECT @IDRow = MAX(IDRow) FROM TableDefinition3 WHERE IDTableObj = @IDTableObj
        IF @IDRow IS NULL
            SELECT @IDRow = 1
        ELSE
            SELECT @IDRow = @IDRow + 1
    
```

Appendix 5: Supplier Database (SD) System Code

```
INSERT INTO TableDefinition3(IDTableObj, IDRow, Column1Val, Column2Val, Column3Val,
RowContent)
VALUES(@IDTableObj, @IDRow, LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),
LTRIM(RTRIM(@Column3Val)), LTRIM(RTRIM(@RowContent)) )
SELECT @Error = @@ERROR
IF @Error != 0
BEGIN
    SELECT @IDProcState = 2
    SELECT @Message = 'An error ocured while inserting the table row. '
    SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '!'
    SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
    NVARCHAR) + '!'
    SELECT @Message = @Message + 'Error ocured in Procedure proc_InsertRow3. '
    SELECT @Message = @Message + 'Procedure is terminated abnormally. '
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END
ELSE
BEGIN
    DROP TABLE #TempTable
    RETURN
END
END --End of IF @RowContent LIKE 'TableRow'
ELSE -- Now enter header information of the table.
BEGIN
    SELECT @IDRow = 0
    INSERT INTO TableDefinition3(IDTableObj, IDRow, Column1Val, Column2Val,Column3Val,
RowContent)
VALUES(@IDTableObj, @IDRow, LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),
LTRIM(RTRIM(@Column3Val)),LTRIM(RTRIM(@RowContent)) )
SELECT @Error = @@ERROR
IF @Error != 0
BEGIN
    SELECT @IDProcState = 3
    SELECT @Message = 'An error ocured while inserting the table row. '
    SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '!'
    SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
    NVARCHAR) + '!'
    SELECT @Message = @Message + 'Error ocured in Procedure proc_InsertRow3. '
    SELECT @Message = @Message + 'Procedure is terminated abnormally. '
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END
ELSE
BEGIN
    DROP TABLE #TempTable
    RETURN
END
END
END -- End of Proc_InsertRow3 Procedure.
GO
```

/*
3.22 Procedure Name: dbo.proc_InsertRow4

Database: SupplierDB

Description:

This procedure enables creation of column values for table specification with 4 columns.

*/

CREATE Procedure Proc_InsertRow4

/* Param List */

```
@IDTableObj          UNIQUEIDENTIFIER,
@ColumnValues         NVARCHAR(4000),
@ColIDMeasUnits      NVARCHAR(4000) = NULL,
@RowContent          NVARCHAR(100),
@IDProcState         TINYINT OUTPUT,
@Message             NVARCHAR(500) OUTPUT
```

AS
BEGIN

DECLARE

```
@Error              INT,
@IDRow              INT,
@NumOfRows          INT,
@Column1Val          NVARCHAR(255),
@Column2Val          NVARCHAR(255),
@Column3Val          NVARCHAR(255),
@Column4Val          NVARCHAR(255)
```

```
SELECT @IDProcState = 0
SELECT @ERROR = 0
SELECT @Message = ''
```

```
SELECT @NumOfRows = COUNT(*) FROM dbo.fn_getColumnValuesTable(@ColumnValues)
--PRINT '@NumOfRows: ' + CAST(@NumOfRows AS NVARCHAR)
/* The function fn_getColumnValuesTable(@ColumnValues) should return two rows only */
```

IF @NumOfRows != 4
BEGIN

```
SELECT @IDProcState = 1
SELECT @Message = 'An error occurred while inserting the table row. '
SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + ' '
SELECT @Message = @Message + 'Error occurred in Procedure proc_InsertRow4. '
SELECT @Message = @Message + 'Procedure is terminated abnormally. '
RAISERROR(@Message,1,1) WITH SETERROR
RETURN
```

END

/*Create a temporary table and insert the values from the table returned
by the function fn_getColumnValuesTable */

```
CREATE TABLE #TempTable (ColumnID INT, ColumnValue NVARCHAR(255) )
INSERT INTO #TempTable
SELECT ColumnID,ColVal FROM dbo.fn_getColumnValuesTable(@ColumnValues)
```

```
SELECT @Column1Val = ColumnValue FROM #TempTable WHERE ColumnID = 1
SELECT @Column2Val = ColumnValue FROM #TempTable WHERE ColumnID = 2
SELECT @Column3Val = ColumnValue FROM #TempTable WHERE ColumnID = 3
SELECT @Column4Val = ColumnValue FROM #TempTable WHERE ColumnID = 4
```

/*Check whether the row contains column values or column specifications
A row can only contain column values or column specifications(headers).*/

IF @RowContent LIKE 'ColVals'
BEGIN

```
/* Check for the existing rows with same IDTableObj to keep track of the number
of rows the IDTableObj has if the @RowContent = 'ColumnValues' */
SELECT @IDRow = MAX(IDRow) FROM TableDefinition4 WHERE IDTableObj = @IDTableObj
IF @IDRow IS NULL
SELECT @IDRow = 1
ELSE
SELECT @IDRow = @IDRow + 1
```

```
INSERT INTO TableDefinition4(IDTableObj, IDRow, Column1Val,
Column2Val,Column3Val,Column4Val, RowContent)
```

Appendix 5: Supplier Database (SD) System Code

```
VALUES(@IDTableObj, @IDRow,
LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),LTRIM(RTRIM(@Column3
Val)),
LTRIM(RTRIM(@Column4Val)), LTRIM(RTRIM(@RowContent)) )
SELECT @Error = @@ERROR
IF @Error != 0
BEGIN
    SELECT @IDProcState = 2
    SELECT @Message = 'An error occured while inserting the table row. '
    SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
    SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
    NVARCHAR) + ' '
    SELECT @Message = @Message + 'Error occured in Procedure proc_InsertRow4. '
    SELECT @Message = @Message + 'Procedure is terminated abnormally. '
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END
ELSE
BEGIN
    DROP TABLE #TempTable
    RETURN
END
END --End of IF @RowContent LIKE 'TableRow'
ELSE -- Now enter header information of the table.
BEGIN
    SELECT @IDRow = 0
    INSERT INTO TableDefinition4(IDTableObj, IDRow, Column1Val, Column2Val, Column3Val,
    Column4Val, RowContent)
    VALUES(@IDTableObj, @IDRow, LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),
    LTRIM(RTRIM(@Column3Val)),LTRIM(RTRIM(@Column4Val)),LTRIM(RTRIM(@RowContent)) )
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        SELECT @IDProcState = 3
        SELECT @Message = 'An error occured while inserting the table row. '
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
        NVARCHAR) + ' '
        SELECT @Message = @Message + 'Error occured in Procedure proc_InsertRow4. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        DROP TABLE #TempTable
        RETURN
    END
END
END -- End of Proc_InsertRow4 Procedure.
GO
```

3.23 Procedure Name: dbo.proc_InsertRow5

Database: **SupplierDB**

Description:

This procedure enables creation of column values for table specification with 5 columns.

*/

CREATE Procedure Proc_InsertRow5

/* Param List */

```
@IDTableObj          UNIQUEIDENTIFIER,
@ColumnValues         NVARCHAR(4000),
@ColIDMeasUnits      NVARCHAR(4000) = NULL,
@RowContent          NVARCHAR(100),
@IDProcState         TINYINT OUTPUT,
@Message             NVARCHAR(500) OUTPUT
```

AS
BEGIN

DECLARE

```
@Error              INT,
@IDRow              INT,
@NumOfRows          INT,
@Column1Val         NVARCHAR(255),
@Column2Val         NVARCHAR(255),
@Column3Val         NVARCHAR(255),
@Column4Val         NVARCHAR(255),
@Column5Val         NVARCHAR(255)
```

```
SELECT @IDProcState = 0
SELECT @ERROR = 0
SELECT @Message = ''
```

```
SELECT @NumOfRows = COUNT(*) FROM dbo.fn_getColumnValuesTable(@ColumnValues)
--PRINT '@NumOfRows: ' + CAST(@NumOfRows AS NVARCHAR)
/* The function fn_getColumnValuesTable(@ColumnValues) should return two rows only */
```

IF @NumOfRows != 5

BEGIN

```
SELECT @IDProcState = 1
SELECT @Message = 'An error occurred while inserting the table row. '
SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + '. '
SELECT @Message = @Message + 'Error occurred in Procedure proc_InsertRow5. '
SELECT @Message = @Message + 'Procedure is terminated abnormally. '
RAISERROR(@Message,1,1) WITH SETERROR
RETURN
```

END

/*Create a temporary table and insert the values from the table returned
by the function fn_getColumnValuesTable */

```
CREATE TABLE #TempTable (ColumnID INT, ColumnValue NVARCHAR(255) )
INSERT INTO #TempTable
SELECT ColumnID,ColVal FROM dbo.fn_getColumnValuesTable(@ColumnValues)
```

```
SELECT @Column1Val = ColumnValue FROM #TempTable WHERE ColumnID = 1
SELECT @Column2Val = ColumnValue FROM #TempTable WHERE ColumnID = 2
SELECT @Column3Val = ColumnValue FROM #TempTable WHERE ColumnID = 3
SELECT @Column4Val = ColumnValue FROM #TempTable WHERE ColumnID = 4
SELECT @Column5Val = ColumnValue FROM #TempTable WHERE ColumnID = 5
```

/*Check whether the row contains column values or column specifications
A row can only contain column values or column specifications(headers).*/

IF @RowContent LIKE 'ColVals'

BEGIN

```
/* Check for the existing rows with same IDTableObj to keep track of the number
of rows the IDTableObj has if the @RowContent = 'ColumnValues' */
SELECT @IDRow = MAX(IDRow) FROM TableDefinition2 WHERE IDTableObj = @IDTableObj
IF @IDRow IS NULL
SELECT @IDRow = 1
ELSE
SELECT @IDRow = @IDRow + 1
```

Appendix 5: Supplier Database (SD) System Code

```
INSERT INTO TableDefinition5(IDTableObj, IDRow, Column1Val,
Column2Val,Column3Val,Column4Val, Column5Val, RowContent)
VALUES(@IDTableObj, @IDRow,
LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),LTRIM(RTRIM(@Column3Val)),
LTRIM(RTRIM(@Column4Val)),LTRIM(RTRIM(@Column5Val)), LTRIM(RTRIM(@RowContent)) )
SELECT @Error = @@ERROR
IF @Error != 0
BEGIN
    SELECT @IDProcState = 2
    SELECT @Message = 'An error occured while inserting the table row. '
    SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' .'
    SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
    NVARCHAR) + ' .'
    SELECT @Message = @Message + 'Error occured in Procedure proc_InsertRow5. '
    SELECT @Message = @Message + 'Procedure is terminated abnormally. '
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END
ELSE
BEGIN
    DROP TABLE #TempTable
    RETURN
END
END --End of IF @RowContent LIKE 'TableRow'
ELSE -- Now enter header information of the table.
BEGIN
    SELECT @IDRow = 0
    INSERT INTO TableDefinition5(IDTableObj, IDRow, Column1Val, Column2Val, Column3Val,
Column4Val, Column5Val, RowContent)
VALUES(@IDTableObj, @IDRow, LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),
TRIM(RTRIM(@Column3Val)),LTRIM(RTRIM(@Column4Val)),LTRIM(RTRIM(@Column5Val)),
LTRIM(RTRIM(@RowContent)) )
SELECT @Error = @@ERROR
IF @Error != 0
BEGIN
    SELECT @IDProcState = 3
    SELECT @Message = 'An error occured while inserting the table row. '
    SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' .'
    SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
    NVARCHAR) + ' .'
    SELECT @Message = @Message + 'Error occured in Procedure proc_InsertRow5. '
    SELECT @Message = @Message + 'Procedure is terminated abnormally. '
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END
ELSE
BEGIN
    DROP TABLE #TempTable
    RETURN
END
END
END -- End of Proc_InsertRow5 Procedure.
GO
```

/*

3.24 Procedure Name: dbo.proc_InsertRow6

Database: SupplierDB

Description:

This procedure enables creation of column values for table specification with 6 columns.

*/

```
CREATE Procedure Proc_InsertRow6
/* Param List */
@IDTableObj          UNIQUEIDENTIFIER,
@ColumnValues         NVARCHAR(4000),
@ColIDMeasUnits      NVARCHAR(4000) = NULL,
@RowContent          NVARCHAR(100),
@IDProcState         TINYINT OUTPUT,
@Message             NVARCHAR(500) OUTPUT
```

```
AS
BEGIN
```

Appendix 5: Supplier Database (SD) System Code

```
DECLARE
@Error          INT,
@IDRow          INT,
@NumOfRows     INT,
@Column1Val     NVARCHAR(255),
@Column2Val     NVARCHAR(255),
@Column3Val     NVARCHAR(255),
@Column4Val     NVARCHAR(255),
@Column5Val     NVARCHAR(255),
@Column6Val     NVARCHAR(255)

SELECT @IDProcState = 0
SELECT @ERROR = 0
SELECT @Message = ''

SELECT @NumOfRows = COUNT(*) FROM dbo.fn_getColumnValuesTable(@ColumnValues)
--PRINT '@NumOfRows: ' + CAST(@NumOfRows AS NVARCHAR)
/* The function fn_getColumnValuesTable(@ColumnValues) should return two rows only */

IF @NumOfRows != 6
BEGIN
    SELECT @IDProcState = 1
    SELECT @Message = 'An error occured while inserting the table row. '
    SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + '. '
    SELECT @Message = @Message + 'Error occured in Procedure Proc_InsertRow6. '
    SELECT @Message = @Message + 'Procedure is terminated abnormally. '
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN
END

/*Create a temporary table and insert the values from the table returned
by the function fn_getColumnValuesTable */

CREATE TABLE #TempTable (ColumnID INT, ColumnValue NVARCHAR(255) )
INSERT INTO #TempTable
SELECT ColumnID,ColVal FROM dbo.fn_getColumnValuesTable(@ColumnValues)

SELECT @Column1Val = ColumnValue FROM #TempTable WHERE ColumnID = 1
SELECT @Column2Val = ColumnValue FROM #TempTable WHERE ColumnID = 2
SELECT @Column3Val = ColumnValue FROM #TempTable WHERE ColumnID = 3
SELECT @Column4Val = ColumnValue FROM #TempTable WHERE ColumnID = 4
SELECT @Column5Val = ColumnValue FROM #TempTable WHERE ColumnID = 5
SELECT @Column6Val = ColumnValue FROM #TempTable WHERE ColumnID = 6

/*Check whether the row contains column values or column specifications
A row can only contain column values or column specifications(headers).*/

IF @RowContent LIKE 'ColVals'
BEGIN
    /* Check for the existing rows with same IDTableObj to keep track of the number
of rows the IDTableObj has if the @RowContent = 'Column Values' */
    SELECT @IDRow = MAX(IDRow) FROM TableDefinition6 WHERE IDTableObj = @IDTableObj
    IF @IDRow IS NULL
        SELECT @IDRow = 1
    ELSE
        SELECT @IDRow = @IDRow + 1

    INSERT INTO TableDefinition6(IDTableObj, IDRow, Column1Val,
Column2Val,Column3Val,Column4Val,
Column5Val, Column6Val, RowContent)
VALUES(@IDTableObj, @IDRow,
LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),LTRIM(RTRIM(@Column3Val)),
LTRIM(RTRIM(@Column4Val)),LTRIM(RTRIM(@Column5Val)),
LTRIM(RTRIM(@Column6Val)),LTRIM(RTRIM(@RowContent)))
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        SELECT @IDProcState = 2
        SELECT @Message = 'An error occured while inserting the table row. '
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '. '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
NVARCHAR) + '. '
        SELECT @Message = @Message + 'Error occured in Procedure Proc_InsertRow6. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
    END
END
```


Appendix 5: Supplier Database (SD) System Code

```
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        DROP TABLE #TempTable
        RETURN
    END
END --End of IF @RowContent LIKE 'TableRow'
ELSE -- Now enter header informatio of the table.
BEGIN
    SELECT @IDRow = 0
    INSERT INTO TableDefinition6(IDTableObj, IDRow, Column1Val, Column2Val, Column3Val,
    Column4Val, Column5Val, Column6Val,RowContent)
    VALUES(@IDTableObj, @IDRow, LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),
    LTRIM(RTRIM(@Column3Val)),LTRIM(RTRIM(@Column4Val)),LTRIM(RTRIM(@Column5Val)),
    LTRIM(RTRIM(@Column6Val)),LTRIM(RTRIM(@RowContent)))
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        SELECT @IDProcState = 3
        SELECT @Message = 'An error ocured while inserting the table row. '
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
        NVARCHAR) + ' '
        SELECT @Message = @Message + 'Error ocured in Procedure Proc_InsertRow6. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        DROP TABLE #TempTable
        RETURN
    END
END
END -- End of Proc_InsertRow6 Procedure.
GO
```

/*

3.25 Procedure Name: dbo.proc_InsertRow7

Database: SupplierDB

Description:

This procedure enables creation of column values for table specification with 7 columns.

*/

```
CREATE Procedure Proc_InsertRow7
```

```
/* Param List */
```

```
@IDTableObj          UNIQUEIDENTIFIER,
@ColumnValues         NVARCHAR(4000),
@ColIDMeasUnits      NVARCHAR(4000) = NULL,
@RowContent          NVARCHAR(100),
@IDProcState         TINYINT OUTPUT,
@Message             NVARCHAR(500) OUTPUT
```

```
AS
```

```
BEGIN
```

```
    DECLARE
```

```
    @Error            INT,
    @IDRow            INT,
    @NumOfRows        INT,
    @Column1Val       NVARCHAR(255),
    @Column2Val       NVARCHAR(255),
    @Column3Val       NVARCHAR(255),
    @Column4Val       NVARCHAR(255),
    @Column5Val       NVARCHAR(255),
    @Column6Val       NVARCHAR(255),
    @Column7Val       NVARCHAR(255)
```

```
    SELECT @IDProcState = 0
    SELECT @ERROR = 0
    SELECT @Message = ''
```

```
    SELECT @NumOfRows = COUNT(*) FROM dbo.fn_getColumnValuesTable(@ColumnValues)
```

Appendix 5: Supplier Database (SD) System Code

```
--PRINT '@NumOfRows: ' + CAST(@NumOfRows AS NVARCHAR)
/* The function fn_getColumnValuesTable(@ColumnValues) should return two rows only */

IF @NumOfRows != 7
BEGIN
    SELECT @IDProcState = 1
    SELECT @Message = 'An error occurred while inserting the table row. '
    SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + '. '
    SELECT @Message = @Message + 'Error occurred in Procedure Proc_InsertRow7. '
    SELECT @Message = @Message + 'Procedure is terminated abnormally. '
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN
END

/*Create a temporary table and insert the values from the table returned
by the function fn_getColumnValuesTable */

CREATE TABLE #TempTable (ColumnID INT, ColumnValue NVARCHAR(255) )
INSERT INTO #TempTable
SELECT ColumnID,ColVal FROM dbo.fn_getColumnValuesTable(@ColumnValues)

SELECT @Column1Val = ColumnValue FROM #TempTable WHERE ColumnID = 1
SELECT @Column2Val = ColumnValue FROM #TempTable WHERE ColumnID = 2
SELECT @Column3Val = ColumnValue FROM #TempTable WHERE ColumnID = 3
SELECT @Column4Val = ColumnValue FROM #TempTable WHERE ColumnID = 4
SELECT @Column5Val = ColumnValue FROM #TempTable WHERE ColumnID = 5
SELECT @Column6Val = ColumnValue FROM #TempTable WHERE ColumnID = 6
SELECT @Column7Val = ColumnValue FROM #TempTable WHERE ColumnID = 7

/*Check whether the row contains column values or column specifications
A row can only contain column values or column specifications(headers).*/

IF @RowContent LIKE 'ColVals'
BEGIN
    /* Check for the existing rows with same IDTableObj to keep track of the number
of rows the IDTableObj has if the @RowContent = 'ColumnValues' */
    SELECT @IDRow = MAX(IDRow) FROM TableDefinition7 WHERE IDTableObj = @IDTableObj
    IF @IDRow IS NULL
        SELECT @IDRow = 1
    ELSE
        SELECT @IDRow = @IDRow + 1

    INSERT INTO TableDefinition7(IDTableObj, IDRow, Column1Val,
Column2Val,Column3Val,Column4Val,
Column5Val, Column6Val,Column7Val, RowContent)
VALUES(@IDTableObj, @IDRow,
LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),LTRIM(RTRIM(@Column3Val)),
LTRIM(RTRIM(@Column4Val)),LTRIM(RTRIM(@Column5Val)), LTRIM(RTRIM(@Column6Val)),
LTRIM(RTRIM(@Column7Val)),LTRIM(RTRIM(@RowContent)))
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        SELECT @IDProcState = 2
        SELECT @Message = 'An error occurred while inserting the table row. '
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '. '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
NVARCHAR) + '. '
        SELECT @Message = @Message + 'Error occurred in Procedure Proc_InsertRow7. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        DROP TABLE #TempTable
        RETURN
    END
END
END --End of IF @RowContent LIKE 'TableRow'
ELSE -- Now insert header information of the table.
BEGIN
    SELECT @IDRow = 0
    INSERT INTO TableDefinition7(IDTableObj, IDRow, Column1Val, Column2Val, Column3Val,
Column4Val, Column5Val, Column6Val,Column7Val, RowContent)
```

Appendix 5: Supplier Database (SD) System Code

```
VALUES(@IDTableObj, @IDRow, LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),
LTRIM(RTRIM(@Column3Val)),LTRIM(RTRIM(@Column4Val)),LTRIM(RTRIM(@Column5Val)),
LTRIM(RTRIM(@Column6Val)),LTRIM(RTRIM(@Column7Val)),LTRIM(RTRIM(@RowContent)))
SELECT @Error = @@ERROR
IF @Error != 0
BEGIN
    SELECT @IDProcState = 3
    SELECT @Message = 'An error occured while inserting the table row. '
    SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '. '
    SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
    NVARCHAR) + '. '
    SELECT @Message = @Message + 'Error occured in Procedure Proc_InsertRow7. '
    SELECT @Message = @Message + 'Procedure is terminated abnormally. '
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END
ELSE
BEGIN
    DROP TABLE #TempTable
    RETURN
END
END
END -- End of Proc_InsertRow7 Procedure.
GO
```

/*

3.26 Procedure Name: dbo.proc_InsertRow8

Database: SupplierDB

Description:

This procedure enables creation of column values for table specification with 8 columns.

*/

```
CREATE Procedure Proc_InsertRow8
/* Param List */
@IDTableObj          UNIQUEIDENTIFIER,
@ColumnValues         NVARCHAR(4000),
@ColIDMeasUnits      NVARCHAR(4000) = NULL,
@RowContent          NVARCHAR(100),
@IDProcState         TINYINT OUTPUT,
@Message             NVARCHAR(500) OUTPUT

AS
BEGIN

DECLARE
@Error              INT,
@IDRow              INT,
@NumOfRows          INT,
@Column1Val         NVARCHAR(255),
@Column2Val         NVARCHAR(255),
@Column3Val         NVARCHAR(255),
@Column4Val         NVARCHAR(255),
@Column5Val         NVARCHAR(255),
@Column6Val         NVARCHAR(255),
@Column7Val         NVARCHAR(255),
@Column8Val         NVARCHAR(255)

SELECT @IDProcState = 0
SELECT @ERROR = 0
SELECT @Message = ''

SELECT @NumOfRows = COUNT(*) FROM dbo.fn_getColumnValuesTable(@ColumnValues)
--PRINT '@NumOfRows: ' + CAST(@NumOfRows AS NVARCHAR)
/* The function fn_getColumnValuesTable(@ColumnValues) should return two rows only */

IF @NumOfRows != 8
BEGIN
    SELECT @IDProcState = 1
    SELECT @Message = 'An error occured while inserting the table row. '
    SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + '. '
    SELECT @Message = @Message + 'Error occured in Procedure Proc_InsertRow8. '
    SELECT @Message = @Message + 'Procedure is terminated abnormally. '
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN
END
```

Appendix 5: Supplier Database (SD) System Code

END

/*Create a temporary table and insert the values from the table returned
by the function fn_getColumnValuesTable */

```
CREATE TABLE #TempTable (ColumnID INT, ColumnValue NVARCHAR(255) )
INSERT INTO #TempTable
SELECT ColumnID,ColVal FROM dbo.fn_getColumnValuesTable(@ColumnValues)
```

```
SELECT @Column1Val = ColumnValue FROM #TempTable WHERE ColumnID = 1
SELECT @Column2Val = ColumnValue FROM #TempTable WHERE ColumnID = 2
SELECT @Column3Val = ColumnValue FROM #TempTable WHERE ColumnID = 3
SELECT @Column4Val = ColumnValue FROM #TempTable WHERE ColumnID = 4
SELECT @Column5Val = ColumnValue FROM #TempTable WHERE ColumnID = 5
SELECT @Column6Val = ColumnValue FROM #TempTable WHERE ColumnID = 6
SELECT @Column7Val = ColumnValue FROM #TempTable WHERE ColumnID = 7
SELECT @Column8Val = ColumnValue FROM #TempTable WHERE ColumnID = 8
```

/*Check whether the row contains column values or column specifications
A row can only contain column values or column specifications(headers).*/

```
IF @RowContent LIKE 'ColVals'
BEGIN
```

```
    /* Check for the existing rows with same IDTableObj to keep track of the number
    of rows the IDTableObj has if the @RowContent = 'ColumnValues' */
    SELECT @IDRow = MAX(IDRow) FROM TableDefinition8 WHERE IDTableObj = @IDTableObj
    IF @IDRow IS NULL
        SELECT @IDRow = 1
    ELSE
        SELECT @IDRow = @IDRow + 1
```

```
    INSERT INTO TableDefinition8(IDTableObj, IDRow, Column1Val,
Column2Val,Column3Val,Column4Val,
    Column5Val, Column6Val,Column7Val, Column8Val, RowContent)
    VALUES(@IDTableObj, @IDRow,
    LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),LTRIM(RTRIM(@Column3Val)),
    LTRIM(RTRIM(@Column4Val)),LTRIM(RTRIM(@Column5Val)), LTRIM(RTRIM(@Column6Val)),
    LTRIM(RTRIM(@Column7Val)),LTRIM(RTRIM(@Column8Val)),LTRIM(RTRIM(@RowContent)))
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
```

```
        SELECT @IDProcState = 2
        SELECT @Message = 'An error occurred while inserting the table row. '
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
        NVARCHAR) + ' '
        SELECT @Message = @Message + 'Error occurred in Procedure Proc_InsertRow8. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
```

```
    END
    ELSE
    BEGIN
```

```
        DROP TABLE #TempTable
        RETURN
```

```
    END
```

```
END --End of IF @RowContent LIKE 'TableRow'
ELSE -- Now insert header information of the table.
BEGIN
```

```
    SELECT @IDRow = 0
    INSERT INTO TableDefinition8(IDTableObj, IDRow, Column1Val, Column2Val, Column3Val,
    Column4Val, Column5Val, Column6Val,Column7Val, Column8Val, RowContent)
    VALUES(@IDTableObj, @IDRow, LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),
    LTRIM(RTRIM(@Column3Val)),LTRIM(RTRIM(@Column4Val)),LTRIM(RTRIM(@Column5Val)),
    LTRIM(RTRIM(@Column6Val)),LTRIM(RTRIM(@Column7Val)),LTRIM(RTRIM(@Column8Val)),
    LTRIM(RTRIM(@RowContent)))
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
```

```
        SELECT @IDProcState = 3
        SELECT @Message = 'An error occurred while inserting the table row. '
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
    END
```

Appendix 5: Supplier Database (SD) System Code

```
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
        NVARCHAR) + ', '
        SELECT @Message = @Message + 'Error occured in Procedure Proc_InsertRow8. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        DROP TABLE #TempTable
        RETURN
    END
END
END -- End of Proc_InsertRow8 Procedure.
GO
```

/*

3.27 Procedure Name: dbo.proc_InsertRow9

Database: SupplierDB

Description:

This procedure enables creation of column values for table specification with 9 columns.

*/

```
CREATE Procedure Proc_InsertRow9
```

```
/* Param List */
```

```
@IDTableObj          UNIQUEIDENTIFIER,
@ColumnValues         NVARCHAR(4000),
@ColIDMeasUnits      NVARCHAR(4000) = NULL,
@RowContent          NVARCHAR(100),
@IDProcState         TINYINT OUTPUT,
@Message             NVARCHAR(500) OUTPUT
```

```
AS
```

```
BEGIN
```

```
    DECLARE
```

```
    @Error            INT,
    @IDRow            INT,
    @NumOfRows        INT,
    @Column1Val       NVARCHAR(255),
    @Column2Val       NVARCHAR(255),
    @Column3Val       NVARCHAR(255),
    @Column4Val       NVARCHAR(255),
    @Column5Val       NVARCHAR(255),
    @Column6Val       NVARCHAR(255),
    @Column7Val       NVARCHAR(255),
    @Column8Val       NVARCHAR(255),
    @Column9Val       NVARCHAR(255)
```

```
    SELECT @IDProcState = 0
    SELECT @ERROR = 0
    SELECT @Message = ''
```

```
    SELECT @NumOfRows = COUNT(*) FROM dbo.fn_getColumnValuesTable(@ColumnValues)
    --PRINT '@NumOfRows: ' + CAST(@NumOfRows AS NVARCHAR)
    /* The function fn_getColumnValuesTable(@ColumnValues) should return two rows only */
```

```
    IF @NumOfRows != 9
```

```
    BEGIN
```

```
        SELECT @IDProcState = 1
        SELECT @Message = 'An error occured while inserting the table row. '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + ', '
        SELECT @Message = @Message + 'Error occured in Procedure Proc_InsertRow9. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN
    END
```

```
END
```

```
/*Create a temporary table and insert the values from the table returned
by the function fn_getColumnValuesTable */
```

```
CREATE TABLE #TempTable (ColumnID INT, ColumnValue NVARCHAR(255) )
INSERT INTO #TempTable
SELECT ColumnID,ColVal FROM dbo.fn_getColumnValuesTable(@ColumnValues)
```

Appendix 5: Supplier Database (SD) System Code

```

SELECT @Column1Val = ColumnValue FROM #TempTable WHERE ColumnID = 1
SELECT @Column2Val = ColumnValue FROM #TempTable WHERE ColumnID = 2
SELECT @Column3Val = ColumnValue FROM #TempTable WHERE ColumnID = 3
SELECT @Column4Val = ColumnValue FROM #TempTable WHERE ColumnID = 4
SELECT @Column5Val = ColumnValue FROM #TempTable WHERE ColumnID = 5
SELECT @Column6Val = ColumnValue FROM #TempTable WHERE ColumnID = 6
SELECT @Column7Val = ColumnValue FROM #TempTable WHERE ColumnID = 7
SELECT @Column8Val = ColumnValue FROM #TempTable WHERE ColumnID = 8
SELECT @Column9Val = ColumnValue FROM #TempTable WHERE ColumnID = 9

```

```

/*Check whether the row contains column values or column specifications
A row can only contain column values or column specifications(headers).*/

```

```

IF @RowContent LIKE 'ColVals'
BEGIN

```

```

    /* Check for the existing rows with same IDTableObj to keep track of the number
    of rows the IDTableObj has if the @RowContent = 'ColumnValues' */
    SELECT @IDRow = MAX(IDRow) FROM TableDefinition9 WHERE IDTableObj = @IDTableObj
    IF @IDRow IS NULL
        SELECT @IDRow = 1
    ELSE
        SELECT @IDRow = @IDRow + 1

```

```

    INSERT INTO TableDefinition9(IDTableObj, IDRow, Column1Val,
Column2Val,Column3Val,Column4Val,
    Column5Val, Column6Val,Column7Val, Column8Val, Column9Val,RowContent)
    VALUES(@IDTableObj, @IDRow,
    LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),LTRIM(RTRIM(@Column3Val)),
    LTRIM(RTRIM(@Column4Val)),LTRIM(RTRIM(@Column5Val)), LTRIM(RTRIM(@Column6Val)),
    LTRIM(RTRIM(@Column7Val)),LTRIM(RTRIM(@Column8Val)), LTRIM(RTRIM(@Column9Val)),
    LTRIM(RTRIM(@RowContent)))
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        SELECT @IDProcState = 2
        SELECT @Message = 'An error occurred while inserting the table row.'
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
        NVARCHAR) + ' '
        SELECT @Message = @Message + 'Error occurred in Procedure Proc_InsertRow9.'
        SELECT @Message = @Message + 'Procedure is terminated abnormally.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        DROP TABLE #TempTable
        RETURN
    END

```

```

END --End of IF @RowContent LIKE 'TableRow'
ELSE -- Now insert header information of the table.
BEGIN

```

```

    SELECT @IDRow = 0
    INSERT INTO TableDefinition9(IDTableObj, IDRow, Column1Val, Column2Val, Column3Val,
Column4Val,
    Column5Val, Column6Val,Column7Val, Column8Val,
    Column9Val, RowContent)
    VALUES(@IDTableObj, @IDRow, LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),
    LTRIM(RTRIM(@Column3Val)),LTRIM(RTRIM(@Column4Val)),LTRIM(RTRIM(@Column5Val)),
    LTRIM(RTRIM(@Column6Val)),LTRIM(RTRIM(@Column7Val)),LTRIM(RTRIM(@Column8Val)),
    LTRIM(RTRIM(@Column9Val)),LTRIM(RTRIM(@RowContent)))
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        SELECT @IDProcState = 3
        SELECT @Message = 'An error occurred while inserting the table row.'
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
        NVARCHAR) + ' '
        SELECT @Message = @Message + 'Error occurred in Procedure Proc_InsertRow9.'
        SELECT @Message = @Message + 'Procedure is terminated abnormally.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END

```

Appendix 5: Supplier Database (SD) System Code

```
        END
        ELSE
        BEGIN
            DROP TABLE #TempTable
            RETURN
        END
    END
END -- End of Proc_InsertRow9 Procedure.
GO

/*
3.28 Procedure Name: dbo.proc_InsertRow10
Database: SupplierDB
Description:
This procedure enables creation of column values for table specification with 10 columns.
*/

CREATE Procedure Proc_InsertRow10
/* Param List */
@IDTableObj          UNIQUEIDENTIFIER,
@ColumnValues         NVARCHAR(4000),
@ColIDMeasUnits      NVARCHAR(4000) = NULL,
@RowContent          NVARCHAR(100),
@IDProcState         TINYINT OUTPUT,
@Message             NVARCHAR(500) OUTPUT

AS
BEGIN
    DECLARE
    @Error             INT,
    @IDRow            INT,
    @NumOfRows        NT,
    @Column1Val       NVARCHAR(255),
    @Column2Val       NVARCHAR(255),
    @Column3Val       NVARCHAR(255),
    @Column4Val       NVARCHAR(255),
    @Column5Val       NVARCHAR(255),
    @Column6Val       NVARCHAR(255),
    @Column7Val       NVARCHAR(255),
    @Column8Val       NVARCHAR(255),
    @Column9Val       NVARCHAR(255),
    @Column10Val      NVARCHAR(255)

    SELECT @IDProcState = 0
    SELECT @ERROR = 0
    SELECT @Message = ''

    SELECT @NumOfRows = COUNT(*) FROM dbo.fn_getColumnValuesTable(@ColumnValues)
    --PRINT '@NumOfRows: ' + CAST(@NumOfRows AS NVARCHAR)
    /* The function fn_getColumnValuesTable(@ColumnValues) should return two rows only */

    IF @NumOfRows != 10
    BEGIN
        SELECT @IDProcState = 1
        SELECT @Message = 'An error occurred while inserting the table row. '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Error occurred in Procedure Proc_InsertRow10. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN
    END

    /*Create a temporary table and insert the values from the table returned
    by the function fn_getColumnValuesTable */

    CREATE TABLE #TempTable (ColumnID INT, ColumnValue NVARCHAR(255) )
    INSERT INTO #TempTable
    SELECT ColumnID,ColVal FROM dbo.fn_getColumnValuesTable(@ColumnValues)

    SELECT @Column1Val = ColumnValue FROM #TempTable WHERE ColumnID = 1
    SELECT @Column2Val = ColumnValue FROM #TempTable WHERE ColumnID = 2
    SELECT @Column3Val = ColumnValue FROM #TempTable WHERE ColumnID = 3
    SELECT @Column4Val = ColumnValue FROM #TempTable WHERE ColumnID = 4
    SELECT @Column5Val = ColumnValue FROM #TempTable WHERE ColumnID = 5
```

Appendix 5: Supplier Database (SD) System Code

```
SELECT @Column6Val = ColumnValue FROM #TempTable WHERE ColumnID = 6
SELECT @Column7Val = ColumnValue FROM #TempTable WHERE ColumnID = 7
SELECT @Column8Val = ColumnValue FROM #TempTable WHERE ColumnID = 8
SELECT @Column9Val = ColumnValue FROM #TempTable WHERE ColumnID = 9
SELECT @Column10Val = ColumnValue FROM #TempTable WHERE ColumnID = 10

/*Check whether the row contains column values or column specifications
A row can only contain column values or column specifications(headers).*/

IF @RowContent LIKE 'ColVals'
BEGIN
    /* Check for the existing rows with same IDTableObj to keep track of the number
of rows the IDTableObj has if the @RowContent = 'ColumnValues' */
    SELECT @IDRow = MAX(IDRow) FROM TableDefinition10 WHERE IDTableObj = @IDTableObj
    IF @IDRow IS NULL
        SELECT @IDRow = 1
    ELSE
        SELECT @IDRow = @IDRow + 1

    INSERT INTO TableDefinition10(IDTableObj, IDRow, Column1Val,
Column2Val,Column3Val,Column4Val,
Column5Val, Column6Val,Column7Val, Column8Val, Column9Val,Column10Val, RowContent)
        VALUES(@IDTableObj, @IDRow,
LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),LTRIM(RTRIM(@Column3Val)),
LTRIM(RTRIM(@Column4Val)),LTRIM(RTRIM(@Column5Val)), LTRIM(RTRIM(@Column6Val)),
LTRIM(RTRIM(@Column7Val)),LTRIM(RTRIM(@Column8Val)), LTRIM(RTRIM(@Column9Val)),
LTRIM(RTRIM(@Column10Val)),LTRIM(RTRIM(@RowContent)))
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        SELECT @IDProcState = 2
        SELECT @Message = 'An error occured while inserting the table row. '
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
NVARCHAR) + ' '
        SELECT @Message = @Message + 'Error occured in Procedure Proc_InsertRow10. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        DROP TABLE #TempTable
        RETURN
    END
END --End of IF @RowContent LIKE 'TableRow'
ELSE -- Now insert header information of the table.
BEGIN
    SELECT @IDRow = 0
    INSERT INTO TableDefinition10(IDTableObj, IDRow, Column1Val, Column2Val, Column3Val,
Column4Val,
Column5Val, Column6Val,Column7Val, Column8Val,
Column9Val, Column10Val,RowContent)
        VALUES(@IDTableObj, @IDRow, LTRIM(RTRIM(@Column1Val)),LTRIM(RTRIM(@Column2Val)),
LTRIM(RTRIM(@Column3Val)),LTRIM(RTRIM(@Column4Val)),LTRIM(RTRIM(@Column5Val)),
LTRIM(RTRIM(@Column6Val)),LTRIM(RTRIM(@Column7Val)),LTRIM(RTRIM(@Column8Val)),
LTRIM(RTRIM(@Column9Val)),LTRIM(RTRIM(@Column10Val)),LTRIM(RTRIM(@RowContent)))
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        SELECT @IDProcState = 3
        SELECT @Message = 'An error occured while inserting the table row. '
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS
NVARCHAR) + ' '
        SELECT @Message = @Message + 'Error occured in Procedure Proc_InsertRow10. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        DROP TABLE #TempTable
        RETURN
    END
END
```


Appendix 5: Supplier Database (SD) System Code

```
END
END -- End of Proc_InsertRow10 Procedure.
GO
```

```
/*
3.29 Procedure Name: dbo.proc_UnAssignProduct2Category
```

```
Database: SupplierDB
```

```
Description:
```

```
This procedure enables unassigning a product from category.
```

```
*/
```

```
CREATE Procedure proc_UnAssignProduct2Category
/* Param List */
```

```
@IDProd          UNIQUEIDENTIFIER,
@IDProcState     TINYINT          OUTPUT,
@Message         NVARCHAR(500)  OUTPUT
```

```
AS
BEGIN
```

```
    DECLARE
    @Error INT
    SELECT @IDProcState = 0
    SELECT @ERROR = 0
```

```
/* This procedure unassigns a product to a category. Important thing to note is that
a product is being assigned not IDProdDef */
```

```
    DELETE Category_ProductClass
    WHERE IDProd = @IDProd
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @IDProcState = 1
        SELECT @Message = 'An error ocured while unassigning product to the category.'
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Error ocured in Procedure proc_UnAssignProduct2Category. '
        SELECT @Message = @Message + 'Procedure is terminated abnormally. '
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
```

```
    ELSE
    BEGIN
        SELECT @Message = 'Product successfully unassigned to the category.'
        RETURN
    END
```

```
END
-- End of Proc_UnAssignProduct2Category.
GO
```

/*
3.30 Procedure Name: dbo.proc_UnAssignProduct2SpecificationGroup

Database: SupplierDB

Description:

This procedure enables unassigning a product from specification group.

*/

CREATE Procedure proc_UnAssignProduct2SpecificationGroup

/* Param List */

@IDProd UNIQUEIDENTIFIER,
 @IDProcState TINYINT OUTPUT,
 @Message NVARCHAR(500) OUTPUT

AS
 BEGIN

 DECLARE
 @Error INT

 SELECT @IDProcState = 0
 SELECT @ERROR = 0

 /* This stored procedure unassigns a product to a specification group. */

 DELETE SpecificationGroupDefinition

 WHERE IDProd = @IDProd

 SELECT @Error = @@ERROR

 IF @Error != 0

 BEGIN

 ROLLBACK TRAN

 SELECT @IDProcState = 1

 SELECT @Message = 'An error occured while unassigning the product to the specification group.'

 SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '.'

 SELECT @Message = @Message + 'Error occured in Procedure proc_UnAssignProduct2SpecificationGroup.'

 SELECT @Message = @Message + 'Procedure is terminated abnormally. '

 RAISERROR(@Message,1,1) WITH SETERROR

 RETURN @@ERROR

 END

 ELSE

 BEGIN

 SELECT @Message = 'Product successfully unassigned to the specification group.'

 RETURN

 END

END -- End of proc_UnAssignProduct2SpecificationGroup Procedure.

GO

/*
3.31 Procedure Name: dbo.proc_UnAssignSubProduct2Product

Database: SupplierDB

Description:

This procedure enables unassigning a (sub) product from product.

```

*/
CREATE Procedure proc_UnAssignSubProduct2Product
/* Param List */
@IDSubProd          UNIQUEIDENTIFIER,
@IDProcState        TINYINT      OUTPUT,
@Message            NVARCHAR(500) OUTPUT

AS
BEGIN
    DECLARE
    @Error INT
    SELECT @IDProcState = 0
    SELECT @ERROR = 0
    /* This procedure unassigns a sub product to another product */
    DELETE ProductDefinition
    WHERE IDSubProd = @IDSubProd
    SELECT @Error = @@ERROR
    IF @Error != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @IDProcState = 1
        SELECT @Message = 'An error occured while unassigning the Subproduct to the product.'
        SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + ' '
        SELECT @Message = @Message + 'Error occured in Procedure proc_UnAssignSubProduct2Product.'
        SELECT @Message = @Message + 'Procedure is terminated abnormally.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        SELECT @Message = 'Subproduct successfully unassigned to the product.'
        RETURN
    END
END
-- End of proc_UnAssignSubProduct2Product Procedure.
GO

```

/*
3.32 Procedure Name: dbo.proc_UpdateProduct

Database: SupplierDB

Description:

This procedure enables update of product data.

```

*/
CREATE Procedure proc_UpdateProduct
/* Param List */
@IDProd            UNIQUEIDENTIFIER,
@ProdName          NVARCHAR(255),
@ProdDesc          NVARCHAR(4000) = NULL,
@AssignTo          NVARCHAR(60) = NULL,
@IDAssignTo        UNIQUEIDENTIFIER = NULL,
@IDProcState        TINYINT      OUTPUT,
@Message            NVARCHAR(500) OUTPUT

AS
BEGIN
    DECLARE
    @Error INT
    SELECT @IDProcState = 0
    SELECT @ERROR = 0

    BEGIN TRAN
    /*
    Update product by inserting new values into product table.
    */
    UPDATE Product
    SET @ProdName = ISNULL(@ProdName, ProdName),
        @ProdDesc = ISNULL(@ProdDesc, ProdDesc)

```

Appendix 5: Supplier Database (SD) System Code

```
FROM PRODUCT
WHERE IDProd = @IDProd
SELECT @Error = @@ERROR
IF @Error != 0
BEGIN
    ROLLBACK TRAN
    SELECT @IDProcState = 1
    SELECT @Message = 'An error occured while updating Product.'
    SELECT @Message = @Message + 'Error code is: ' + CAST(@Error AS NVARCHAR) + '.'
    SELECT @Message = @Message + 'Procedure State ID is: ' + CAST(@IDProcState AS NVARCHAR) + '.'
    SELECT @Message = @Message + 'Error occured in Procedure Proc_UpdateProduct.'
    SELECT @Message = @Message + 'Procedure is terminated abnormally.'
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END
/*
As part of updating process of a product, a product can be reassigned to a product,
specification group or a category. In the following block we check whether @AssignTo is null.
If it is null, then there is no need to reassign a product. Otherwise the product
needs to be reassigned to another product, specification group or category. As part of
accomplishing this process we need to delete earlier assignment a product has to
a product, specification group or category.
*/
IF @AssignTo IS NULL
BEGIN
    COMMIT TRAN
    SELECT @Message = 'Product Updated Successfully.'
    RETURN
END
/*
Unassign a subProduct to a product.
*/
EXEC proc_UnAssignSubProduct2Product
@IDSubProd = @IDProd,
@IDProcState = @IDProcState OUTPUT,
@Message = @Message OUTPUT
IF @IDProcState != 0
BEGIN
    ROLLBACK TRAN
    SELECT @Message = @Message + ' This Procedure was called from Proc_UpdateProduct.'
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END
/*
Unassign a Product to a specification group.
*/
EXEC proc_UnAssignProduct2SpecificationGroup
@IDProd = @IDProd,
@IDProcState = @IDProcState OUTPUT,
@Message = @Message OUTPUT
IF @IDProcState != 0
BEGIN
    ROLLBACK TRAN
    SELECT @Message = @Message + ' This Procedure was called from Proc_UpdateProduct.'
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END
/*
Unassign a Product to a category.
*/
EXEC proc_UnAssignProduct2Category
@IDProd = @IDProd,
@IDProcState = @IDProcState OUTPUT,
@Message = @Message OUTPUT
IF @IDProcState != 0
BEGIN
    ROLLBACK TRAN
    SELECT @Message = @Message + ' This Procedure was called from Proc_UpdateProduct.'
    RAISERROR(@Message,1,1) WITH SETERROR
    RETURN @@ERROR
END
/* A product now needs to be reassigned to a new product, spec group or category */
/* Assigning product to another product*/
```

Appendix 5: Supplier Database (SD) System Code

```
IF @AssignTo = 'Product'
BEGIN
    EXEC proc_AssignSubProduct2Product
        @IDProd      = @IDAssignTo,
        @IDSubProd   = @IDProd,
        @IDProcState = @IDProcState OUTPUT,
        @Message     = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from Proc_UpdateProduct.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        /*If everything goes well until this point it means that all the values are inserted
        properly into the above tables. Now we need to commit the transaction. */
        COMMIT TRAN
        SELECT @Message = 'Product Updated Successfully.'
        RETURN
    END
END -- End of IF @AssignTo = 'product'

IF @AssignTo = 'SpecificationGroup'
BEGIN
    EXEC proc_AssignProduct2SpecificationGroup
        @IDSpecGroup = @IDAssignTo,
        @IDProd      = @IDProd,
        @IDProcState = @IDProcState OUTPUT,
        @Message     = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from Proc_UpdateProduct.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        /*If everything goes well until this point it means that all the values are inserted
        properly into the above tables. Now we need to commit the transaction. */

        COMMIT TRAN
        SELECT @Message = 'Product Updated Successfully.'
        RETURN
    END
END -- IF @AssignTo = 'SpecificationGroup'

IF @AssignTo = 'Category'
BEGIN
    EXEC proc_AssignProduct2Category
        @IDCategory = @IDAssignTo,
        @IDProd     = @IDProd,
        @IDProcState = @IDProcState OUTPUT,
        @Message    = @Message OUTPUT
    IF @IDProcState != 0
    BEGIN
        ROLLBACK TRAN
        SELECT @Message = @Message + ' This Procedure was called from Proc_UpdateProduct.'
        RAISERROR(@Message,1,1) WITH SETERROR
        RETURN @@ERROR
    END
    ELSE
    BEGIN
        /*If everything goes well until this point it means that all the values are inserted
        properly into the above tables. Now we need to commit the transaction. */

        COMMIT TRAN
        SELECT @Message = 'Product Updated Successfully.'
        RETURN
    END
END -- IF @AssignTo = 'Category'
END -- End of Proc_UpdateProduct Procedure.
GO
```

/*
3.33 Procedure Name: dbo.proc_ProductSel

Database: SupplierDB

Description:

This procedure enables selection of product data.

*/

```
CREATE PROCEDURE proc_ProductSel
@IDProdClass          BIGINT,
@OrderBy              NVARCHAR(255) = NULL,
@UpDown              NVARCHAR(255) = 'ASC',
@RecordCount         INT OUTPUT
AS
DECLARE @SQL NVARCHAR(4000)
SET NoCount ON
SET @SQL = ' SELECT dbo.Product.IDProdInternal,
dbo.Product.IDProdClass,
dbo.Product.IDProdClassVer,
dbo.Product.ProdName,
dbo.Product.ProdDesc
FROM dbo.Product
WHERE IDProdClass = @IDProdClass'

IF (@OrderBy IS NOT NULL)
BEGIN
    SET @SQL = @SQL + ' ORDER BY ' + @OrderBy + ' ' + @UpDown
END
EXEC sp_executesql @SQL, N'@IDProdClass BIGINT',@IDProdClass
SELECT @RecordCount = @@rowcount
GO
```

/*
3.34 Function Name: dbo.fn_getColumnValuesTable

Database: SupplierDB

Description:

This function extracts column values from a string Column values in a string are delimited by '*****' */
 After extracting the column values the function inserts each individual column value into the table and returns the table to the called procedure.

*/

```
CREATE FUNCTION dbo.fn_getColumnValuesTable
(@ColumnValues NVARCHAR(4000))
RETURNS @ColumnValuesTable TABLE
(
    ColumnID INT,
    ColVal NVARCHAR(500)
)
AS
BEGIN
    DECLARE
    @Index INT, /* Keeps the index of position from where the delimiter starts. ie the starting position of
    '*****' in a string */
    @DONE TINYINT, /*Acts as a boolean variable. */
    @ColumnVal NVARCHAR(500), /*Holds each individial column value */
    @Counter INT

    /*The following is executed when the procedure is called with empty string as input parameter */
    SELECT @DONE = 0
    SELECT @ColumnValues = LTRIM(RTRIM(@ColumnValues))
    IF LEN(@ColumnValues) = 0
    BEGIN
        SELECT @DONE = 1
        RETURN
    END

    /* The following is executed when the string contains only one column values */
    SELECT @Index = CHARINDEX('###', @ColumnValues)
    IF @Index = 0
    BEGIN
        SELECT @ColumnVal = @ColumnValues
        SELECT @ColumnVal = LTRIM(RTRIM(@ColumnVal))
    END
```

Appendix 5: Supplier Database (SD) System Code

```
        SELECT @Counter = 1
        INSERT INTO @ColumnValuesTable (ColumnID,ColVal) VALUES (@Counter, @ColumnValues)
        SELECT @DONE = 1
        RETURN
    END

/* The following loop is executed when there are more than one column values in the string */
SELECT @Counter = 1
WHILE @DONE = 0
BEGIN
    SELECT @Index = CHARINDEX('###', @ColumnValues)
    SELECT @ColumnVal = LEFT(@ColumnValues, @Index - 1)
    SELECT @ColumnVal = LTRIM(RTRIM(@ColumnVal))
    IF LEN(@ColumnVal) > 0
        INSERT INTO @ColumnValuesTable (ColumnID,ColVal) VALUES (@Counter, @ColumnVal)
    ELSE
        INSERT INTO @ColumnValuesTable (ColumnID,ColVal) VALUES (@Counter, NULL)
    SELECT @Counter = @Counter + 1
    SELECT @ColumnValues = SUBSTRING(@ColumnValues, @Index + 3, LEN(@ColumnValues) - @Index + 2)
    SELECT @ColumnValues = LTRIM(RTRIM(@ColumnValues))
    SELECT @Index = CHARINDEX('###', @ColumnValues)
    IF @Index = 0
    BEGIN
        IF LEN(@ColumnValues) = 0
        BEGIN
            SELECT @DONE = 1
        END
        ELSE
        BEGIN
            SELECT @ColumnVal = @ColumnValues
            SELECT @ColumnVal = LTRIM(RTRIM(@ColumnVal))
            IF LEN(@ColumnVal) > 0
                INSERT INTO @ColumnValuesTable (ColumnID,ColVal) VALUES (@Counter,
@ColumnVal)
            ELSE
                INSERT INTO @ColumnValuesTable (ColumnID,ColVal) VALUES (@Counter, NULL)
            SELECT @DONE = 1
        END
    END
END
END
RETURN
END
```

/* 3.35 Function Name: dbo.fn_GetIDEntityPart

Database: SupplierDB

Description:

This function returns entity part from a complete ID.

*/

```
CREATE FUNCTION dbo.fn_GetIDEntityPart
(@IDComplete BIGINT)
RETURNS INT
AS
BEGIN
    RETURN CAST(SUBSTRING(CAST(@IDComplete AS NVARCHAR), 1, 3) AS INT)
END
```

/*

3.36 Function Name: dbo.fn_GetIDPart

Database: SupplierDB

Description:

This function returns ID part from a complete ID.

*/

```
CREATE FUNCTION dbo.fn_GetIDPart
(@IDComplete BIGINT)
RETURNS BIGINT
AS
BEGIN
DECLARE
    @IDCompleteLength TINYINT,
    @IDPart            NVARCHAR(20)
SELECT @IDCompleteLength = LEN(CAST(@IDComplete AS NVARCHAR))
SELECT @IDPart = SUBSTRING(CAST(@IDComplete AS NVARCHAR), 4, @IDCompleteLength - 3)
RETURN CAST(@IDPart AS BIGINT)
END
```

/*

3.37 Function Name: dbo.fn_GetNewID

Database: SupplierDB

Description:

This function generates a new ID for given entity such as product class, specification, etc.

*/

```
CREATE FUNCTION dbo.fn_GetNewID
(@IDEntity INT)
RETURNS BIGINT
AS
BEGIN
DECLARE
    @IDAvailable BIGINT,
    @IDNext      BIGINT
SELECT @IDAvailable = IDAvailable FROM Entity WHERE [IDEntity] = @IDEntity
SELECT @IDNext = dbo.fn_IncrementID(@IDAvailable, DEFAULT)
RETURN @IDAvailable
END
```

/*

3.38 Function Name: dbo.fn_IncrementID

Database: SupplierDB

Description:

This function increments ID.

*/

```
CREATE FUNCTION dbo.fn_IncrementID
(@IDComplete BIGINT,
 @IncrementBy INT = 1)
RETURNS BIGINT
AS
BEGIN
DECLARE
    @IDPart      BIGINT,
    @EntityPart  BIGINT
SELECT @IDPart = dbo.fn_GetIDPart(@IDComplete)
SELECT @EntityPart = dbo.fn_GetIDEntityPart(@IDComplete)
SELECT @IDPart = @IDPart + @IncrementBy
RETURN CAST(CAST(@EntityPart AS NVARCHAR) + CAST(@IDPart AS NVARCHAR) AS BIGINT)
END
```


/*

3.39 Function Name: dbo.fn_GetNextAvailableID

Database: SupplierDB

Description:

This function generates next available ID.

*/

```
CREATE FUNCTION dbo.fn_GetNextAvailableID
(@IDComplete BIGINT)
RETURNS BIGINT
AS
BEGIN
    DECLARE
        @IDPart          BIGINT,
        @EntityPart      BIGINT
    SELECT @IDPart = dbo.GetIDPart(@IDComplete)
    SELECT @EntityPart = dbo.GetIDEntityPart(@IDComplete)
    SELECT @IDPart = @IDPart + 1
    RETURN CAST(CAST(@EntityPart AS NVARCHAR) + CAST(@IDPart AS NVARCHAR) AS BIGINT)
END
```

Master Grid Service (MGS) System Code

```

/*****
*****
** 4.1 Interface: MasterGrid
** Description:
** This is an interface implemented by MasterGridImpl class which initiates the database search jobs.
*****
*****/

package uk.co.activeplan.mgs.impl;

public interface MasterGrid
{
    public String callTestGridService(String url);
    public String executeJob(String searchString, String supplierString, String gshString);
}

/*****
*****
** 4.2 Class: MasterGridImpl
** Description:
** This class implements MasterGrid interface to provide executeJob method for submitting jobs to
** Database Search Services (DSS) in a Grid environment. The executeJob method parses the
** supplierString input argument (this argument provides a list of suppliers databases to search to the
** executeJob method) to split number of supplier databases to search in equal proportions and allocate
** these supplier lists to available DSSs. The gshString input argument provides a list of DSS available
** in a Grid environment using which the search could be performed and searchString input argument
** provides product class ID pertaining to which product data has to be retrieved from supplier
** databases. The callTestGridService method provides the service of checking whether a DSS is
** available in a Grid environment and is ready to accept requests.
*****
*****/

package uk.co.activeplan.mgs.impl;

import org.w3c.dom.Document;
import org.w3c.dom.NodeList;
import org.w3c.dom.Element;
import org.w3c.dom.DOMImplementation;
import org.w3c.dom.CharacterData;

import java.net.URL;
import org.gridforum.ogsi.OGSIServiceGridLocator;
import org.gridforum.ogsi.GridService;
import org.gridforum.ogsi.Factory;
import org.gridforum.ogsi.LocatorType;
import org.globus.ogsa.utils.GridServiceFactory;
import uk.co.activeplan.mdss.DatabaseSearch.DatabaseSearchServiceGridLocator;
import uk.co.activeplan.mdss.DatabaseSearch.DatabaseSearchPortType;

public class MasterGridImpl implements MasterGrid
{
    private GshDocumentParser gshDocumentParser;
    private SupplierDocumentParser supplierDocumentParser;
    private Converter converter;
    private int gshCount;
    private int supplierCount;
    private ThreadGroup threadGroup;

```

Appendix 5: Master Grid Service (MGS) System Code

```
private DataAggregate dataAggregate;
private CallDataAggregate callDataAggregate;
private URL gsh;
private Factory factory;
private GridServiceFactory databaseSearchFactory;
private LocatorType locator;
private OGSIServiceGridLocator gridLocator;
public MasterGridImpl()
{
    converter = new Converter();
}
} // end of public MasterGridImpl constructor.
public String callTestGridService(String url)
{
    String returnStr = "";
    try
    {
        url.trim();
        gridLocator = new OGSIServiceGridLocator();
        java.net.URL gshURL = new java.net.URL(url);
        factory = gridLocator.getFactoryPort(gshURL);
        databaseSearchFactory = new GridServiceFactory(factory);
        LocatorType locator = databaseSearchFactory.createService();
        DatabaseSearchServiceGridLocator databaseSearchLocator = new
        DatabaseSearchServiceGridLocator();
        DatabaseSearchPortType databaseSearch =
        databaseSearchLocator.getDatabaseSearchService(locator);
        returnStr = databaseSearch.testGridService(url);
    }
    catch(Exception e)
    {
        returnStr = "Error Occured: Either the URL is malformed or the grid service at
        specified URL does not exist.";
    }
    return returnStr;
} // end of public String callTestGridService(String url).

public String executeJob(String searchString, String supplierString, String gshString)
{
    gshCount = converter.getElementCount(gshString, "GridServiceHandle");
    supplierDocumentParser = new SupplierDocumentParser(supplierString, gshCount);
    gshDocumentParser = new GshDocumentParser(gshString);
    callDataAggregate = new CallDataAggregate();
    dataAggregate = callDataAggregate.getDataAggregate();
    JobExecution jobExecution;
    ThreadChecker threadChecker = new ThreadChecker();
    for(int i=0; i<gshCount;i++)
    {
        jobExecution = new JobExecution(searchString,
        supplierDocumentParser.getSupplierSubDocumentString(),
        gshDocumentParser.getGsh(),callDataAggregate );
        jobExecution.setThreadChecker(threadChecker);
        Thread thread = new Thread(threadGroup, jobExecution);
        thread.start();
    } // end of for loop.
    JobAggregation jobAggregation = new JobAggregation();
    jobAggregation.setThreadChecker(threadChecker);
    jobAggregation.setThreadCount(gshCount);
    jobAggregation.setDataAggregate(dataAggregate);
    Thread jobAggregationThread = new Thread(threadGroup, jobAggregation);
}
```

Appendix 5: Master Grid Service (MGS) System Code

```
        jobAggregationThread.setPriority(Thread.MIN_PRIORITY);
        jobAggregationThread.start();
        System.out.println jobAggregation.callGetAggregateString();
        return jobAggregation.callGetAggregateString();
    } //End: public String executeJob(String searchString, String supplierString, String gshString)
} // End: public class MasterGridImpl
```

```
/*
*****
*****

```

** 4.3 Class: Converter

** Description:

```
** This class converts a string (containing data in XML format) into XML document and vice versa.
** The string can contain details on product supplier databases to search or product supplier database
** search criteria. The functionality provided by this class is needed for transportation of data/ search
** criteria (in a string format) between MGS (Master Grid Service) and individual DSS (Database
** Search Services) and for providing the same data to other classes in this package for identifying
** supplier/search criteria details by converting data into XML element objects. The two main methods
** of this class: getStringObject and getStringObject provide these functionalities. The services of this
** class are also available in DSS system component.
```

```
*****
*****/
```

```
package uk.co.activeplan.mgs.impl;
```

```
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.FactoryConfigurationError;
import javax.xml.parsers.ParserConfigurationException;
```

```
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerException;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.TransformerConfigurationException;
```

```
import javax.xml.transform.stream.StreamResult;
import javax.xml.transform.stream.StreamSource;
import java.io.StringReader;
import java.io.StringWriter;
import java.io.BufferedReader;
import java.io.FileReader;
```

```
import org.w3c.dom.Document;
import org.w3c.dom.NodeList;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.dom.DOMResult;
```

```
import org.xml.sax.InputSource;
```

```
public class Converter
```

```
{
    TransformerFactory transformerFactory;
    Transformer transformer;
    DocumentBuilderFactory factory;
    DocumentBuilder builder;
    Document document;
    Document doc;
    BufferedReader bufferedReader;
    DOMResult domResult;
    StringWriter stringWriter;
```

```
StringReader stringReader;
public Converter()
{
    try
    {

        transformerFactory = TransformerFactory.newInstance();
        transformer = transformerFactory.newTransformer();
        factory = DocumentBuilderFactory.newInstance();

    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
} //End: public Converter()

//Convert a string to document object.
public Document getDomObject(String xmlString)
{
    try
    {
        builder = factory.newDocumentBuilder();
        doc = builder.newDocument();
        //DOMResult domRes = new DOMResult(doc);
        stringReader = new StringReader(xmlString);
        bufferedReader = new BufferedReader(stringReader);
        InputSource is = new InputSource(bufferedReader);
        doc = builder.parse(is);
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    return doc;
}
public String getStringObject(Document document)
{
    try
    {
        stringWriter = new StringWriter();
        DOMSource domSrc = new DOMSource(document);
        StreamResult streamRes = new StreamResult(stringWriter);
        transformer.transform(domSrc, streamRes);
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    return stringWriter.toString();
} // end of public String getStringObject(Document document)

public int getElementCount(Document document, String elemName)
{
    NodeList nodeList1 = document.getElementsByTagName(elemName);
    System.out.println(Integer.toString(nodeList1.getLength()));
    return nodeList1.getLength();
} //End: public int getElementCount(Document document, String elemName)
```



```
TransformerFactory transformerFactory;
Transformer transformer;
DocumentBuilderFactory factory;
    DocumentBuilder builder;
DOMImplementation domImplementation;
Document document;
Element rootElement;
NodeList nodeList;
BufferedReader bufferedReader;
DOMResult domResult;
StringWriter stringWriter;
StringReader stringReader;
int tempNumber;
public DataAggregate(String docName)
{
    try
    {
        transformerFactory = TransformerFactory.newInstance();
        transformer = transformerFactory.newTransformer();
        factory = DocumentBuilderFactory.newInstance();
        builder = factory.newDocumentBuilder();
        domImplementation = builder.getDOMImplementation();
        document = domImplementation.createDocument(null, docName, null);
        rootElement = document.getDocumentElement();

    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
}
public void setTempNumber(int i)
{
    tempNumber = i;
}
public int getTempNumber()
{
    return tempNumber;
}
public void addNodes(String newDocument, String tagName)
{
    //get Document representation of the string newDocument.
    try
    {
        //wait();
        Document doc = getDomObject(newDocument);
        nodeList = doc.getElementsByTagName(tagName);
        /*
        Add the nodes to the document object by importing nodes from the
        newDocument
        object. Element rootElement represents the documentElement. i.e. the element
        created in the constructor.
        */
        //Element elem;
        //String elemStr = "";
        for(int i=0; i<nodeList.getLength(); i++)
        {

            rootElement.appendChild
            (document.importNode(nodeList.item(i),true));
        }
    }
}
```

Appendix 5: Master Grid Service (MGS) System Code

```
    }
    //System.out.println("Nodes added");
    /*Now add any error nodes returned from the search. For example, some web
    services may not be responding or URL to web services may not be correct.In
    this case an error node is returned by the grid service for that web service.The
    error nodes are
    added here.*/
    nodeList = doc.getElementsByTagName("ErrorString");
    for(int i=0; i<nodeList.getLength(); i++)
    {
        rootElement.appendChild
        (document.importNode(nodeList.item(i),true));
    }
}
catch(Exception e)
{
    e.printStackTrace();
}
} //End: public void addNodes(String newDocument, String tagName)

//Convert a string to document object.
public Document getDomObject(String xmlString)
{
    Document doc = builder.newDocument();

    try
    {
        DOMResult domRes = new DOMResult(doc);
        stringReader = new StringReader(xmlString);
        bufferedReader = new BufferedReader(stringReader);
        StreamSource ss = new StreamSource(bufferedReader);
        transformer.transform(ss, domRes);

    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    return doc;
} //End: public Document getDomObject(String xmlString)

public String getAggregateString()
{
    DOMSource domSource;
    try
    {
        stringWriter = new StringWriter();
        DOMSource domSrc = new DOMSource(document);
        StreamResult streamRes = new StreamResult(stringWriter);
        transformer.transform(domSrc, streamRes);
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    // System.out.println("The string at getAggregateString is:\n " + stringWriter);
    return stringWriter.toString();
} //End: public String getAggregateString()
} //End: public class DataAggregate
```

```
/**
*****
** 4.5 Class: CallDataAggregate
** Description:
** This class provides synchronised acces to DataAggregate class. Methods of this class provides
** synchronised access to addNodes method of DataAggregate class. Access to DataAggregate class
** needs to be synchronised because more than one job execution thread of JobExecution class accesses
** methods of DataAggregate class.
*****
*****/
```

```
package uk.co.activeplan.mgs.impl;
```

```
class CallDataAggregate
{
    private DataAggregate dataAggregate;
    public CallDataAggregate()
    {
        dataAggregate = new DataAggregate("ActivePlanDataSet");
    }
    private boolean accessible = true; //condition variable.
    public synchronized void callAddNodes(String nodesString)
    {
        while(!accessible)
        {
            try
            {
                wait();
            }
            catch(InterruptedException e)
            {
                e.printStackTrace();
            }
        } // end of while.
        accessible = false;
        dataAggregate.addNodes(nodesString,"Product");
        accessible = true;
        notify();
    } // end of public synchronized void callAddNodes(String nodesString)
    public synchronized void callAddNodes(String nodesString, String tagName)
    {
        while(!accessible)
        {
            try
            {
                wait();
            }
            catch(InterruptedException e)
            {
                e.printStackTrace();
            }
        } // end of while.
        accessible = false;
        dataAggregate.addNodes(nodesString,tagName);
        accessible = true;
        notify();
    } // end of public synchronized void callAddNodes(String nodesString)
```

```

public DataAggregate getDataAggregate()
{
    return dataAggregate;
}
} // End of class callDataAggregate.

/*****
*****
** 4.6 Class: GshDocumentParser
** Description:
** GshDocumentParser parses the GshDocument containing the GridServiceHandles of all the registered
** Grid services available as part of DSS system component. The getGsh() method returns individual
** Grid service handle strings.
*****
*****/

package uk.co.activeplan.mgs.impl;

import org.w3c.dom.*;
import org.w3c.dom.DOMImplementation;
import org.w3c.dom.CharacterData;
import uk.co.activeplan.mgs.impl.Converter;

public class GshDocumentParser
{
    private int tempIterator;
    private int iterator;
    private Element elem;
    private Document gshDocument;
    private NodeList gshNodeList;
    private String elemStr = "";
    private Converter converter;
    public GshDocumentParser(String GshString)
    {
        iterator = 0;
        converter = new Converter();
        gshDocument = converter.getDomObject(GshString);
        gshNodeList = gshDocument.getElementsByTagName("GSH");
    }

    public String getGsh()
    {
        elem = (Element)gshNodeList.item(getIterator());
        elemStr = (((CharacterData)elem.getFirstChild()).getData());
        return elemStr;
    }

    public int getIterator()
    {
        tempIterator = iterator;
        iterator = iterator + 1;
        return tempIterator;
    }
} // end of public getIterator()
} // End of class GshDocumentParser

```

```

/*****
*****
** 4.7 Class: JobExecution
** Description:
** This class invokes a Grid service (via its Grid Service Handle URL) available as part of DSS
** component in a Grid environment. An instance of this class is run as a thread which is spawned by
** MasterGridImpl. The MasterGridImpl class creates several threads of this class to invoke more than
** one Grid service in a Grid environment so that a large number of database search operations could be
** performed by one or more Grid service instances in a collaborative manner. This constructor of this
** class is provided with a list of supplier databases to search in supplierSubStr, search criteria in
** searchStr, Grid Service Handle URL in gshURL and instance of CallDataAggregate class in cda to
** accumulate search results returned by the Grid service. Since the instance of this class is run as a
** thread by MasterGridImpl, it has to notify ThreadChecker class when it has finished its operation (i.e.
** run method executed successfully). Once all the threads have finished execution, it is only then the
** product data searched from supplier databases (by these JobExecution threads) is returned to the user.
*****
*****/

```

```
package uk.co.activeplan.mgs.impl;
```

```

import java.net.URL;
import org.gridforum.ogsi.OGSIServiceGridLocator;
import org.gridforum.ogsi.GridService;
import org.gridforum.ogsi.Factory;
import org.gridforum.ogsi.LocatorType;
import org.globus.ogsa.utils.GridServiceFactory;
import uk.co.activeplan.mdss.DatabaseSearch.DatabaseSearchServiceGridLocator;
import uk.co.activeplan.mdss.DatabaseSearch.DatabaseSearchPortType;

```

```

class JobExecution implements Runnable
{
    OGSIServiceGridLocator gridLocator = new OGSIServiceGridLocator();
    URL gsh;
    Factory factory;
    GridServiceFactory databaseSearchFactory;
    LocatorType locator;
    DatabaseSearchServiceGridLocator databaseSearchLocator;
    DatabaseSearchPortType databaseSearch;
    String searchString = "";
    String supplierSubString = "";
    String gshUrlString = "";
    String str = "";
    CallDataAggregate callDataAggregate;
    ThreadChecker threadChecker;
    public JobExecution(String searchStr,String supplierSubStr, String gshURL,
                       CallDataAggregate cda)
    {
        try
        {
            searchString = searchStr;
            supplierSubString = supplierSubStr;
            gsh = new java.net.URL(gshURL);
            factory = gridLocator.getFactoryPort(gsh);
            databaseSearchFactory = new GridServiceFactory(factory);
            callDataAggregate = cda;
            gshUrlString = gshURL;
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }
}

```

```

    }
}
public void setThreadChecker(ThreadChecker tc)
{
    threadChecker = tc;
}
synchronized public void run()
{
    try
    {
        System.out.println("Supplier String: \n" + supplierSubString);
        System.out.println("Gsh URL: \n" + gshUrlString);
        locator = databaseSearchFactory.createService();
        databaseSearchLocator = new DatabaseSearchServiceGridLocator();
        databaseSearch = databaseSearchLocator.getDatabaseSearchService(locator);
        str = databaseSearch.getProductsAsString(searchString, supplierSubString);
        System.out.println(str + "\n\n");
        callDataAggregate.callAddNodes(str);
        threadChecker.setThreadsCompleted();

        //Cleanup
        GridService gridService = gridLocator.getGridServicePort(locator);
        gridService.destroy();
        gridLocator = null;
        factory = null;
        databaseSearchFactory = null;
        locator = null;
        databaseSearch = null;
        str = null;
        System.gc();
    }
    catch(Exception e)
    {
        threadChecker.setErrorsOccured();
        threadChecker.setThreadsCompleted();
        //Cleanup.
        try
        {
            GridService gridService = gridLocator.getGridServicePort(locator);
            gridService.destroy();
            gridLocator = null;
            factory = null;
            databaseSearchFactory = null;
            locator = null;
            databaseSearch = null;
            str = null;
            System.gc();
        }
        catch(Exception exception)
        {
            exception.printStackTrace();
        } // End: catch(Exception exception)
    } //End: catch(Exception exception)
} //End of synchronized public void run().
} //End of class JobExecution implements Runnable.

```

```

/*****
*****
** 4.8 Class: JobAggregation
** Description:
** An instance of this class is initialised by MasterGridImpl. It is initialised as a thread. The role of the
** run method of this class is to check at regular interval the completion of job execution threads and
** aggregation of data. It is only after all the job execution threads have completed and data aggregated,
** that it is sent to the user. Once the execution is complete and data aggregated the
** dataAggregationDone is set to true in the run method. After this the MasterGridImpl calls
** callGetAggregateString method to retrieve all the results obtained (i.e. product data) from product
** supplier databases.
*****
*****/

```

```
package uk.co.activeplan.mgs.impl;
```

```
public class JobAggregation implements Runnable
{
```

```

    private boolean dataAggregationDone = false;
    private String aggregateString = "";
    ThreadGroup threadGroup;
    int threadCount = 0;
    ThreadChecker threadChecker;
    DataAggregate dataAggregate;
    public void setThreadChecker(ThreadChecker tc)
    {
        threadChecker = tc;
    }
    public void setThreadCount(int i)
    {
        threadCount = i;
    }
    public void setDataAggregate(DataAggregate da)
    {
        dataAggregate = da;
    }
    synchronized public void run()
    {
        while(threadChecker.getThreadsCompleted() < threadCount)
        {
            try
            {
                wait(1 * 1000);
            }
            catch(Exception e)
            {
                threadChecker.setErrorsOccured();
                e.printStackTrace();
            }
        }
        System.out.println("\n\n Number of Errors Occured: " +
            Integer.toString(threadChecker.getErrorsOccured()));
        System.out.println("Number of threads running currently inside the run method: " +
            threadGroup.activeCount());
        aggregateString = dataAggregate.getAggregateString();
        dataAggregationDone = true;
    } // end of synchronized public void run()

    synchronized public String callGetAggregateString()

```

```

{
    while(!dataAggregationDone)
    {
        try
        {
            wait(1000);
        }
        catch(Exception e)
        {
            System.out.println("Exception in callGetAggregateString() method of
JobAggregation");
        }
    } // end of while.
    return aggregateString;
} //end of synchronized public String callGetAggregateString()
} //End: JobAggregation implements Runnable

```

```

/*****
*****

```

**** 4.9 Class: SupplierDocumentParser**

**** Description:**

** SupplierDocumentParser creates number of sub documents which are equally divided among the
** available GSHs. Every time a call is made to the method getSupplierSubDocumentString, it returns a
** string that is a sub document of document which contains list of suppliers. An instance of this class is
** intialised by MasterGridImpl class.

```

*****
*****/

```

```

package uk.co.activeplan.mgs.impl;

```

```

import org.w3c.dom.*;
import org.w3c.dom.DOMImplementation;
import org.w3c.dom.CharacterData;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import uk.co.activeplan.mgs.impl.Converter;

```

```

public class SupplierDocumentParser
{

```

```

    private int loopCount = 0;
    private int temp;
    private int gshCount;
    private int supplierCount;
    private int supplierPerGSH;
    private int remainder = 0;
    private int i;
    private int currentNode;
    private int currentNodeTemp;
    private DocumentBuilderFactory factory;
    private DocumentBuilder builder;
    private DOMImplementation domImplementation;
    private Element rootElement;
    private Document document;
    private NodeList supplierNodeList;
    private Converter converter;
    public SupplierDocumentParser(Document doc, int numOfGSHs)
    {
        document = doc;
        gshCount = numOfGSHs;
        supplierNodeList = document.getElementsByTagName("Supplier");
    }

```

Appendix 5: Master Grid Service (MGS) System Code

```
supplierCount = supplierNodeList.getLength();
currentNode = 0;
converter = new Converter();
supplierPerGSH = supplierCount/gshCount;
setRemainder(supplierCount % gshCount);

try
{
    factory = DocumentBuilderFactory.newInstance();
    builder = factory.newDocumentBuilder();
    domImplementation = builder.getDOMImplementation();
}
catch(Exception e)
{
    e.printStackTrace();
}
}
public SupplierDocumentParser(String docString, int numOfGSHs)
{
    converter = new Converter();
    document = converter.getDomObject(docString);
    gshCount = numOfGSHs;
    supplierNodeList = document.getElementsByTagName("Supplier");
    supplierCount = supplierNodeList.getLength();
    currentNode = 0;

    supplierPerGSH = supplierCount/gshCount;
    setRemainder(supplierCount % gshCount);

    try
    {
        factory = DocumentBuilderFactory.newInstance();
        builder = factory.newDocumentBuilder();
        domImplementation = builder.getDOMImplementation();
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
}
public String getSupplierSubDocumentString()
{
    //The following loop iterates depending on the number of suppliers
    //to be put into a xml document.
    System.out.println("Supplier per GSH is: " + Integer.toString(supplierPerGSH));
    loopCount = supplierPerGSH + getRemainder();
    System.out.println("Loop count is: " + Integer.toString(loopCount));
    document = domImplementation.createDocument(null, "SupplierList", null);
    rootElement = document.getDocumentElement();
    for(i=0; i<loopCount; i++)
    {
        rootElement.appendChild
            (document.importNode(supplierNodeList.item(getCurrentNode()),true));
    }
    return converter.getStringObject(document);
}
public void setRemainder(int rem)
{
```

```

        remainder = rem;
        System.out.println("Remainder is set. It is: " + Integer.toString(remainder));
    }
    public int getCurrentNode()
    {
        currentNodeTemp = currentNode;
        currentNode = currentNode + 1;
        return currentNodeTemp;
    }

    public int getRemainder()
    {
        if (remainder > 0)
        {
            remainder = remainder - 1;
            //System.out.println("Returning remainder");
            return 1;
        }
        else
        {
            return 0;
        }
    }
} // end of public int getRemainder()
} // End of class SupplierDocumentParser

```

```

/*****
*****

```

**** 4.10 Class: ThreadChecker**

**** Description:**

**** This class keeps count of number of job execution threads which are initiated by MasterGridImpl class. Every time a JobExecution thread completes its operations it informs this class by invoking its *setThreadsCompleted* method. The JobAggregation class invokes *getErrorsOccured* and *getThreadsCompleted* methods of this class to keep track of threads completed or errors occurred. It is only after all the threads have completed their execution that JobAggregation class signals MasterGridImpl class to send the data retrieved from supplier databases to the user.**

```

*****
*****/

```

```

package uk.co.activeplan.mgs.impl;
public class ThreadChecker
{
    private int threadsCompleted = 0;
    int errorsOccured = 0;
    public void setErrorsOccured()
    {
        errorsOccured = errorsOccured + 1;
    }
    public int getErrorsOccured()
    {
        return errorsOccured;
    }
    public void setThreadsCompleted()
    {
        threadsCompleted = threadsCompleted + 1;
    }
    public int getThreadsCompleted()
    {
        return threadsCompleted;
    }
} //End of class ThreadChecker

```


Database Search Service (DSS) System Code

```

/*****
*****
** 5.1 Interface: DatabaseSearch
** Description:
** This is an interface implemented by database search classes which perform product data search in a
** Grid environment as a Grid service.
*****
*****/

package uk.co.activeplan.mdss_4;
public interface DatabaseSearch
{
    public String getProductsAsString(String targetGSH, String searchCriteria);

    public String testGridService(String gsh);
}

/*****
*****
** 5.2 Class: DatabaseSearchImpl
** Description:
** This class performs a search of product supplier databases in a Grid environment. The functionality
** of this class is available as a Grid service in a Grid environment. This main method of this class
** "getProductsAsString" is invoked by MGS (Master Grid Service) providing product search criteria (as
** xmlSearchString) and a list of product supplier databases ( as xmlSupplierString) to search. The
** method then invokes each of the product supplier databases identified in the xmlSupplierString and
** aggregates product data retrieved from them. The aggregated product data is sent back to the calling
** MGS.
*****
*****/

package uk.co.activeplan.mdss_4;
import java.net.URL;
import org.w3c.dom. Document;
import uk.co.activeplan.SupplierWS_4. ProductSupplierWebServiceSoapStub;
public class DatabaseSearchImpl implements DatabaseSearch
{
    private Converter converter;
    private int supplierCount;
    private Document supplierDocument;
    private ProductSupplierWebServiceSoapStub stub;
    private URL url = null;
    private SupplierParser supplierParser;
    private String result = "";
    private DataAggregate dataAggregate;
    private String[] supplierDetails = new String[3];
    public DatabaseSearchImpl()
    {
    }
    }
    Element elem;
    String str = "";
    Object obj[];
    Object any[];

```

Appendix 5: Database Search Service (DSS) System Code

```
public String getProductsAsString(String xmlSearchString, String xmlSupplierString)
{
    converter = new Converter();
    dataAggregate = new DataAggregate("GridServiceResponse");
    supplierParser = new SupplierParser(xmlSupplierString);
    supplierCount = converter.getElementCount(xmlSupplierString, "Supplier");
    supplierDocument = converter.getDomObject(xmlSupplierString);

    System.out.println("Supplier count: " + Integer.toString(supplierCount));
    System.out.println("Supplier String: \n" + xmlSupplierString);

    for (int i=0;i<supplierCount;i++)
    {
        try
        {
            /*
            supplierDetails[0] contains ID of the supplier.
            supplierDetails[1] contains supplier web service URL.
            supplierDetails[2] contains name of the dataset which is expected to
            be returned by the supplier.
            */
            //System.out.println("Identifying supplier details...");
            supplierDetails = supplierParser.getSupplierDetails();
            url = new URL(supplierDetails[1]);

            System.out.println("The supplier URL: " );
            System.out.println(supplierDetails[0]);
            System.out.println(supplierDetails[1]);
            System.out.println(supplierDetails[2]);
            System.out.println("The URL is: " + url.toString() + "\n\n");

            /* Prepare stub by identifying the supplier web service url. The
            supplier Web Dervice is invoked by providing the Web Service with
            the product data search string. Set an acceptable time out so that the
            stub does not wait too long in circumstances when the supplier web
            service is unavailable/offline. If retrieving of data from supplier
            Web Service is successful then the retrieved data is aggregated into
            an xml document by DataAggregate class by identifying data
            retrieved and supplier details i.e. the supplier which provided the
            product data.
            */

            stub = new ProductSupplierWebServiceSoapStub(url,null);
            stub.setTimeout(60 * 5);
            System.out.println("The timeout is: " + stub.getTimeout());
            result = stub.getProductsAsString(xmlSearchString);
            //System.out.println(result);
            dataAggregate.addNodes(result, supplierDetails);
        }
        catch(Exception e)
        {
            /* This exception is thrown when this class is unable to invoke the
            Supplier Web Service. When the stub is unable to retried data from
            Supplie Web Service URL then error string is encoded into XML
            element and added to the returned XML document by DataAggregate
            class. This XML element identifies the supplier Web Service from
            which product data could not be retrieved.
            */
        }
    }
}
```

Appendix 5: Database Search Service (DSS) System Code

```
*/

String errStr = "\n<Error>\n\t<ErrorString>\n\t\tCould not retrieve
data from URL " + url.toString() +
"\n Either URL is malformed or connection to the web service is \n
refused (a web service " + "may not be available) or access to the
database server is denied or is
unavailable.\n\t</ErrorString>\n\t</Error>\n";
supplierDetails[2] = "Error";
dataAggregate.addNodes(errStr, supplierDetails);
} //end of catch.
finally
{
    //Release system/memory resources.
    stub = null;
    result = null;
    url = null;

} // End of finally block.
} //end of for loop.

/*
Once all product data is retrieved from identified product suppliers in SupplierDetails
array the aggregate data has to be transformed into String from XML object for
transportaion to MGS. For this call getAAggregateString method of AggregateData
class.
*/

String aggData = dataAggregate.getAggregateString();
System.out.println("The aggregate data: \n" + aggData);

//Set the following class to null to free system/memory resources.

converter = null;
dataAggregate = null;
supplierDocument = null;
supplierParser = null;

//Perform gargabe collection in order to free system/memory resources.
System.gc();
return aggData;
} //End of method: public String getProductsAsString(String xmlSearchString, String
//xmlSupplierString)

/* The following is a test method to identify whether a grid service for performing supplier
database search is available in the Grid environment.
*/
public String testGridService(String gsh)
{
    return "Grid service available at Grid Service Handle: " + gsh;
}

} //End of class: DatabaseSearchImpl
```

```
/*
*****
*****
*/
```

**** 5.3 Class: DataAggregate**

**** Description:**

** This class provides of functionality of aggregating product data retrieved from one or more suppliers into an XML document. The class is intialised by DatabaseSearchImpl class in a grid environment which retrieves product data from supplier databases and passes to this class by calling its addNodes method. This method then documents what product data is retrieved from which product supplier. Once all product supplier databases have been searched, getAggregateString method of this class is called by DatabaseSearchImpl to get all product data retrieved from all product suppliers in string format which is then returned to the MGS (Master Grid Service).

*****/

```
package uk.co.activeplan.mdss_4;
```

```
import java.net.URL;  
import org.apache.xerces.dom.*;  
import org.w3c.dom.*;
```

```
import javax.xml.parsers.DocumentBuilder;  
import javax.xml.parsers.DocumentBuilderFactory;  
import javax.xml.parsers.FactoryConfigurationError;  
import javax.xml.parsers.ParserConfigurationException;
```

```
import org.xml.sax.SAXException;  
import org.xml.sax.SAXParseException;  
import org.xml.sax.ContentHandler;  
import org.xml.sax.InputSource;
```

```
// For write operation
```

```
import javax.xml.transform.Transformer;  
import javax.xml.transform.TransformerException;  
import javax.xml.transform.TransformerFactory;  
import javax.xml.transform.TransformerConfigurationException;  
import javax.xml.transform.sax.SAXSource;  
import javax.xml.transform.stream.StreamResult;  
import javax.xml.transform.stream.*;  
import java.io.*;  
import org.w3c.dom.*;  
import javax.xml.transform.dom.DOMSource;  
import javax.xml.transform.dom.DOMResult;
```

```
public class DataAggregate  
{
```

```
    TransformerFactory transformerFactory;  
    Transformer transformer;  
    DocumentBuilderFactory factory;  
    DocumentBuilder builder;  
    DOMImplementation domImplementation;  
    Document document;  
    Element rootElement;  
    NodeList nodeList;  
    BufferedReader bufferedReader;  
    DOMResult domResult;  
    StringWriter stringWriter;  
    StringReader stringReader;  
    int tempNumber;
```

```
    public DataAggregate(String docName)  
    {  
        try
```

```
{
    transformerFactory = TransformerFactory.newInstance();
    transformer = transformerFactory.newTransformer();
    factory = DocumentBuilderFactory.newInstance();
    builder = factory.newDocumentBuilder();
    domImplementation = builder.getDOMImplementation();
    document = domImplementation.createDocument(null, docName, null);
    rootElement = document.getDocumentElement();
}
catch(Exception e)
{
    e.printStackTrace();
}
}
public void setTempNumber(int i)
{
    tempNumber = i;
}
public int getTempNumber()
{
    return tempNumber;
}
public void addNodes(String newDocument, String[] supplierDetails)
{
    //get Document representation of the string newDocument.
    try
    {
        Document doc = getDomObject(newDocument);
        nodeList = doc.getElementsByTagName(supplierDetails[2]);
        /*
        Add the nodes to the document object by importing nodes from the
        newDocument object. Element rootElement represents the documentElement.
        i.e. the element created in the constructor. supplierDetails[0] contains ID of the
        supplier. supplierDetails[1] contains supplier web service URL.
        supplierDetails[2] contains name of the dataset which is expected to be
        returned by the supplier.
        */

        Element elem;
        for(int i=0; i<nodeList.getLength(); i++)
        {
            elem = (Element)nodeList.item(i);
            elem.setAttribute("IDSupplier", supplierDetails[0]);
            elem.setAttribute("SupplierWsURL",supplierDetails[1]);
            rootElement.appendChild
            (document.importNode(nodeList.item(i),true));

        }
        //System.out.println("Nodes added");
        notify();
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }

    System.out.println(elemStr);
    elemStr = "";
}
```

Appendix 5: Database Search Service (DSS) System Code

```
NodeList nodeList = document.getElementsByTagName("ProductName");
for(int i=0; i<nodeList.getLength(); i++)
{
    rootElement.appendChild(document.importNode(nodeList.item(i),true));
    elem = (Element)nodeList.item(i);
    elemStr = elemStr + ((CharacterData)elem.getFirstChild()).getData() + "\n";
}
System.out.println("From nodeList 2: \n " + elemStr);

} //End: public void addNodes(String newDocument, String[] supplierDetails)

//Convert a string to document object.
public Document getDomObject(String xmlString)
{
    Document doc = builder.newDocument();

    try
    {
        DOMResult domRes = new DOMResult(doc);
        stringReader = new StringReader(xmlString);
        bufferedReader = new BufferedReader(stringReader);
        StreamSource ss = new StreamSource(bufferedReader);
        transformer.transform(ss, domRes);

    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    return doc;
}

public String getAggregateString()
{
    DOMSource domSource;
    try
    {
        stringWriter = new StringWriter();
        DOMSource domSrc = new DOMSource(document);
        StreamResult streamRes = new StreamResult(stringWriter);
        transformer.transform(domSrc, streamRes);

    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    System.out.println("The string at getAggregateString is:\n " + stringWriter);
    return stringWriter.toString();
}
} //End of class: DataAggregate
```

```
/*
*****
*****
*/
```

**** 5.4 Class: SupplierParser**

**** Description:**

****** This class parses the xmlSupplierString string. This string contains details on all the suppliers
****** databases that have to be searched. These details are provided by MGS (Master Grid Service). This
****** class is initialised by DatabaseSearchImpl to retrieve all the details on a supplier database(s). The
****** main method of this class getSupplierDetails goes through the xmlSupplierString (which is converted
****** into XML DOM object by converter class) and returns individual supplier details to the Grid service
****** for invoking product supplier databases.

*****/

```
package uk.co.activeplan.mdss_4;
import org.w3c.dom.*;
import org.w3c.dom.DOMImplementation;
import org.w3c.dom.CharacterData;
//import uk1.co.activeplan.mdss.impl.Converter;
```

```
public class SupplierParser
{
    private int tempIterator;
    private int iterator;
    private Element elem;
    private Document supplierListDocument;
    private NodeList supplierNodeList;
    private NodeList tempNodeList;
    private String elemStr = "";
    private Converter converter;
    private String[] supplierDetails = new String[3];
    private Element tempElement;
    public SupplierParser(String xmlSupplierString)
    {
        iterator = 0;
        converter = new Converter();
        supplierListDocument = converter.getDomObject(xmlSupplierString);
        supplierNodeList = supplierListDocument.getElementsByTagName("Supplier");
    }

    public String[] getSupplierDetails()
    {
        //Extract the supplier node from the supplierNodeList.
        elem = (Element)supplierNodeList.item(getIterator());

        //Get the "IDSupplier" node into the tempNodeList and extract the supplier id
        // into the supplierDetails[0].
        tempNodeList = elem.getElementsByTagName("IDSupplier");
        tempElement = (Element)tempNodeList.item(0);
        supplierDetails[0] = (((CharacterData)tempElement.getFirstChild()).getData());
        System.out.println("IDSupplier(SupplierDetails[0]): " + supplierDetails[0]);

        /*Get the "SupplierWS" node into the tempNodeList and extract the supplier web
        service url into the supplierDetails[1]. */
        tempNodeList = elem.getElementsByTagName("SupplierWS");
        tempElement = (Element)tempNodeList.item(0);
        supplierDetails[1] = (((CharacterData)tempElement.getFirstChild()).getData());
        System.out.println("Supplier Web Service URL (SupplierDetails[1]): " +
        supplierDetails[1]);

        //Get the "DataSetName" node into the tempNodeList and extract the name of the
        //dataset
        //that the supplier is expected to return.
    }
}
```

Appendix 5: Database Search Service (DSS) System Code

```
        tempNodeList = elem.getElementsByTagName("DataSetName");
        tempElement = (Element)tempNodeList.item(0);
        supplierDetails[2] = (((CharacterData)tempElement.getFirstChild()).getData());
        System.out.println("DataSet Name(SupplierDetails[2]): " + supplierDetails[2]);

        return supplierDetails;
    }

    public int getIterator()
    {
        tempIterator = iterator;
        iterator = iterator + 1;
        return tempIterator;
    } // end of public getIterator()
} // End of class SupplierParser

/*****
*****
** 5.5 Class: Converter
** Description:
** This class converts a string (containing data in XML format) into XML document and vice versa.
** The string can contain details on product supplier databases to search or product supplier database
** search criteria. The functionality provided by this class is needed for transportation of data/ search
** criteria (in a string format) between MGS (Master Grid Service) and individual DSS (Database Search
** Services) and for providing the same data to other classes in this package for identifying
** supplier/search criteria details by converting data into XML element objects. The two main methods of
** this class: getStringObject and getStringObject provide these functionalities.
*****
*****/

package uk.co.activeplan.mdss_4;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.FactoryConfigurationError;
import javax.xml.parsers.ParserConfigurationException;

import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerException;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.TransformerConfigurationException;

import javax.xml.transform.stream.StreamResult;
import javax.xml.transform.stream.StreamSource;
import java.io.StringReader;
import java.io.StringWriter;
import java.io.BufferedReader;
import java.io.FileReader;

import org.w3c.dom.Document;
import org.w3c.dom.NodeList;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.dom.DOMResult;

import org.xml.sax.InputSource;

public class Converter
```

```
{
    TransformerFactory transformerFactory;
    Transformer transformer;
    DocumentBuilderFactory factory;
    DocumentBuilder builder;
    Document document;
    Document doc;
    BufferedReader bufferedReader;
    DOMResult domResult;
    StringWriter stringWriter;
    StringReader stringReader;
    public Converter()
    {
        try
        {
            transformerFactory = TransformerFactory.newInstance();
            transformer = transformerFactory.newTransformer();
            factory = DocumentBuilderFactory.newInstance();

        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    } //End: public Converter()

    //Convert a string to document object.
    public Document getDomObject(String xmlString)
    {
        try
        {
            builder = factory.newDocumentBuilder();
            doc = builder.newDocument();
            stringReader = new StringReader(xmlString);
            bufferedReader = new BufferedReader(stringReader);
            InputSource is = new InputSource(bufferedReader);
            doc = builder.parse(is);
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
        return doc;
    }

    public String getStringObject(Document document)
    {
        try
        {
            stringWriter = new StringWriter();
            DOMSource domSrc = new DOMSource(document);
            StreamResult streamRes = new StreamResult(stringWriter);
            transformer.transform(domSrc, streamRes);
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

Appendix 5: Database Search Service (DSS) System Code

```
        return stringWriter.toString();
    } // End: public String getStringObject(Document document)

    public int getElementCount(Document document, String elemName)
    {
        NodeList nodeList1 = document.getElementsByTagName(elemName);
        System.out.println(Integer.toString(nodeList.getLength()));
        return nodeList1.getLength();
    } //End: public int getElementCount(Document document, String elemName)

    public int getElementCount(String documentString, String elemName)
    {
        Document stringDoc = getDomObject(documentString);
        NodeList nodeList2 = stringDoc.getElementsByTagName(elemName);
        return nodeList2.getLength();
    } //End: public int getElementCount(String documentString, String elemName)
} // end of public class Converter
```

References

- [Aga98] Agapiou, A., Flanagan, R., Norman, G. and Notman, D. The changing role of builders merchants in the construction supply chain. In: *Construction Management and Economics*. 15(3), pp.351-361, May 1998.
- [Ahm91] Ahmed, R., Smedt, P. D., Du, W., Kent, W., Ketabchi, M. A., Litwin, W. A., Rafii, A. and Shan, M.-C., The Pegasus Heterogeneous Multidatabase System. In: *Computer*, 24(12), pp.19-27, 1991.
- [Ahm08] Ahmad, F., Zakaria, N. H., Osman, W. R. S. Transforming Information-Based Agricultural Portal to Knowledge-Based Agricultural Hub, In: *Proc. 3rd International Conference on Information and Communication Technologies: From Theory to Applications, 2008. (ICTTA 2008)*, pp.1-4, 7-11 April 2008.
- [Ahm08a] Ahmed, E., Bessis, N., Yue, Y., Sarfraz, M., Matching multiple elements in grid databases: A practical approach, In: *Proc. Third International Conference on Digital Information Management, (ICDIM 2008.)*, pp.757-762, November 2008.
- [Aka08] Akahoshi, Y., Kidawara, Y., and Tanaka, K. 2008. A database-oriented wrapper for ubiquitous data acquisition/access environments. In: *Proc. 2nd international Conference on Ubiquitous information Management and Communication (ICUIMC '08)*, Suwon, Korea, January 31 - February 01, 2008.
- [Alm04] Almarimi, A. and Pokorny, J. A Mediation Layer for Heterogeneous XML Schemas. In: *The sixth International Conference on Information Integration and Web-based Applications Services (IIWAS'2004)*, Jakarta, Indonesia, 2004.
- [Amb03] Ambler, S. *Agile Database Techniques: Effective Strategies for the Agile Software Developer*, Wiley, October 2003.
- [Apa07] The Apache Tomcat Website, 2007. [Online]. Available: <http://tomcat.apache.org/>, last accessed: 23 December 2007
- [Apa07a] The Apache Axis website, 2007. [Online]. Available: <http://ws.apache.org/axis/>, last accessed: 23 December 2007.
- [Aps09] Active Plan Solutions Limited (APSL) website. [Online]. Available: <http://www.activeplan.co.uk/>. Last accessed: 18 February 2009.
- [Are93] Arens, Y., Chee, C. Y., Hsu, C.-N. and Knoblock, C. A. Retrieving and Integrating Data from Multiple Information Sources. In: *International Journal of Cooperative Information Systems*. vol. 2(2), pp.127-158, 1993.
-

-
- [Are97] Arens, Y., Hsu, C.-N. and Knoblock, C. A. Query processing in the SIMS information mediator. In: M. N. Huhns and M. P. Singh (Eds.): *Readings in agents*. Morgan Kaufmann Publishers Inc., pp.82-90, 1997.
- [Atk05] Atkinson, M., Karasavvas, K., Antonioletti, M., Baxter, R., Borley, A., Hong, N. C., Hume, A., Jackson, M., Krause, A., Laws, S., Paton, N., Schopf, J. M., Sudgen, T., Tourlas, K. and Watson, P. A New Architecture for OGSA-DAI. In: *Proc. of the UK e-Science All Hands Meeting 2005 (AHM'05)*, Nottingham, UK, 2005.
- [Aus84] Austen, A.D., Neale, R. H. (Eds.). *Managing construction projects - a guide to processes and procedures*. A. D. International Labor Office, Geneva, 1984
- [Bai05] Baily, P., Farmer, D., Jessop, D. and Jones, D. *Purchasing Principles and Management*. 9th ed., FT Prentice Hall, 2005.
- [Bak86] Bakos, J. Y. and Treacy, M.E. Information Technology and Corporate Strategy: A Research Perspective (Draft Version). In: *MIS Quarterly*. pp. 107-119, June, 1986.
- [Bar92] Barrie, D. S. and Paulson, B. C. *Professional construction management: including C.M., design-construct, and general contracting*. International ed., McGraw-Hill, 1992.
- [Bar99] Baru, C., Gupta, A., Ludäscher, B., Marciano, R., Papakonstantinou, Y., Velikhov, P. and Chu, V. XML-based information mediation with MIX. In: *Proc. ACM SIGMOD International Conference on Management of Data (SIGMOD '99)*, Philadelphia, Pennsylvania, United States, pp.597-599, May 31 - June 03, 1999.
- [Bas08] Basney, J, Martin, S., Navarro, J., Pierce, M., Scavo, T., Strand, L., Uram, T., Wilkins-Diehr, N., Wu, W., Youn, C. The Problem Solving Environments of TeraGrid, Science Gateways, and the Intersection of the Two, In: *Proc. IEEE Fourth International Conference on eScience, (eScience '08.)*, pp.725-734, December 2008.
- [Bat86] Batini, C., Lenzerini, M. and Navathe, S. B. A comparative analysis of methodologies for database schema integration. In: *ACM Comput. Surv.* 18(4), pp.323-364, 1986.
- [Bay97] Bayrado Jr., R. J., Bohrer, W., Brice, R., Cichocki, A., Fowler, J., Helal, A., Kashyap, V., Ksiezyk, T., Martin, G., Nodine, M., Rashid, M., Rusinkiewicz, M., Shea, R., Unnikrishnan, C., Unruh, A. and Woelk, D. InfoSleuth: agent-based semantic integration of information in open and dynamic environments. In: *Proc. ACM SIGMOD International Conference on Management of Data*, Tucson, Arizona, United States, 1997.
-

-
- [Bec04] Bechhofer, S., Harmelen, F. v., Hendler, J., Horrocks, I., McGuinness, D.L., Patel-Schneider, P. F. and Stein, L. A. OWL Web Ontology Language Reference. In: M. Dean and G. Schreiber (Ed.). *W3C working draft*. Feb 2004. [Online]. Available: <http://www.w3.org/TR/owl-ref/> , last accessed: 21 February 2008.
- [Bel02] Bell, W. H., Bosio, D., Hoschek, W., Kunszt, P., McCance, G. and Silander, M. Project Spitfire - Towards Grid Web Service Databases. In: *Proc UK e-Science All Hands Conference*, Sheffield, UK, September 2002.
- [Ben01] Benchikha, F., Boufaïda, M. and Seinturier, L. Integration of the viewpoint mechanism in federated databases. In: *Proc. ACM Symposium on Applied Computing*, Las Vegas, Nevada, United States, 2001.
- [Ber08] Berger, S. and Schrefl, M., From Federated Databases to a Federated Data Warehouse System, In: *Proc. 41st Annual Hawaii International Conference on System Sciences*, pp.394-394, 7-10 January 2008.
- [Bet99] Betts, M. The significance of IT. In: M. Betts (Ed.): *Strategic management of IT in construction*. Blackwell, Oxford, 1999.
- [Bet99a] Betts, M. and Clark, A. The scope of IT in construction. In: M. Betts (Ed.): *Strategic management of IT in construction*. Blackwell, Oxford, pp.133, 1999.
- [Bey97] Beynon-Davies, P., Bonde, L., McPhee, D. and Jones, C. B. A Collaborative Schema Integration System. In: *Comput. Supported Coop. Work*. 6 (1), pp.1-18, 1997.
- [Bir06] XML Schema Part 2: Datatypes Second Edition. In: P. V. Biron, K. Permanente and A. Malhotra (Eds.): *W3C Recommendation*. October 2004. [Online]. Available: <http://www.w3.org/TR/xmlschema-2/> , last accessed: 21 February 2008.
- [Bla92] Blanchard, B. S. *Logistics engineering and management*. 4th ed., Prentice-Hall, 1992.
- [Bon08] Bonifati, A., Chrysanthis, P. K., Ouksel, A. M., and Sattler, K., Distributed databases and peer-to-peer databases: past and present. *SIGMOD Rec.* 37(1), pp.5-11, March 2008.
- [Bou94] Bouguettaya, A. Large Multidatabases: Beyond Federation and Global Schema Integration. In: *Proc. Fifth Australasian Database Conference*, Christchurch, New Zealand, pp.258-273, 1994.
- [Bou00] Bouguettaya, A., Benatallah, B., Hendra, L., Ouzzani, M. and Beard, J. Supporting dynamic interactions among Web-based information sources. In: *IEEE Transactions on Knowledge and Data Engineering*. 12(5), pp.779 - 801, 2000.
-

-
- [Bre97] Bressan, S., Goh, C. H., Fynn, K., Jakobisiak, M., Hussein, K., Kon, H., Lee, T., Madnick, S., Pena, T., Qu, J., Shum, A. and Siegel, M. The Context Interchange mediator prototype. In: *Proc. ACM SIGMOD International Conference on Management of Data*. pp.525-527, 1997.
- [Bur99] Burn, J., Marshall, A. P. and Wild, A. M. Managing knowledge for strategic advantage in the virtual organisation. In: *Proc. ACM SIGCPR Conference on Computer Personnel Research*, New Orleans, Louisiana, United States, pp.19-26, 1999.
- [Bur03] Burnap, P., Joita, L., Pahwa, J. S., Gray, A., Rana, O. and Miles, J. Supporting Collaborative Working of Construction Industry Consortia via the Grid. In: *Proc. UK e-Science Programme All Hands Meeting (AHM'03)*, Nottingham, UK, 2-4 September 2003.
- [Bur04] Burnap, P., Joita, L., Pahwa, J. S., Gray, A., Rana, O. and Miles, J. Security, User and Data Management for Collaborative Virtual Teams in a Grid Environment Supporting Consortia in the Construction Industry. In: *Proc. Workshop on Requirements Capture for Collaboration in e-Science*, Sponsored by National e-Science Centre, Edinburgh, United Kingdom, pp.54-60, 14-15 January 2004.
- [Bur05] Burnap, P., Pahwa, J. S., Joita, L., Gray, W. A., Rana, O. and Miles, J. C. Grid based e-Procurement. In: *Proc. ASCE International Conference on Computing in Civil Engineering*, Cancun, Mexico, July 12–15, 2005.
- [Bus99] Busse, S., Kutsche, R.-D., Leser, U. and Weber, H. Federated Information Systems: Concepts, Terminology and Architectures. In: *Forschungsberichte des Fachbereichs Informatik Nr. 99-9, Technische Universität Berlin*. 1999.
- [Cal95] Calvert, R. E. *Introduction to building management*. 6th ed., Butterworth-Heinemann, Boston, Oxford, 1995.
- [Car95] Carey, M. J., L. M. Haas, Schwarz, P. M., Arya, M., Cody, W. F., Fagin, R., Flickner, M., Luniewski, A. W., Niblack, W., Petkovic, D., Thomas, J., Williams, J. H. and Wimmers, E. L. Towards Heterogeneous Multimedia Information Systems: The Garlic Approach. In: *Proc. 5th International Workshop on Research Issues in Data Engineering: Distributed Object Management, (RIDE-DOM '95)*, pp.124-131, 1995.
- [Cat94] The Object Database Standard: ODMG-93. In: R. Cattell (Ed.). Morgan Kaufmann, San Francisco, 1994.
- [Cat97] Cattell, R. G. G. and Hamilton, G. *Jdbc Database Access with Java: a Tutorial Annotated Reference*. ed., Addison-Wesley Longman Publishing Co., 1997.
-

-
- [Cat00] The object data standard: ODMG 3.0. In: R. G. G. Cattell, D. Barry, M. Berler, J. Eastman, D. Jordan, C. Russell, O. Schadow, T. Stanienda and F. Velez (Eds.). San Francisco: Morgan Kaufmann, 2000.
- [Cha00] Chamberlin, D., Robie, J. and Florescu, D. Quilt: An XML Query Language for Heterogeneous Data Sources. In: *International Workshop on the Web and Databases (WebDB'2000)*, Dallas, Texas, 2000.
- [Che01] Cheng, E. W. L., Li, H., Love, P. E. D. and Irani, Z. An e-business model to support supply chain activities in construction. In: *Logistics Information Management*. 14(1-2), pp.68 -78, 2001.
- [Che08] Acoustic Air Movement Limited website, The Cheetah Fan Coil Unit Brochure, 2008. [Online]. Available: http://www.caice.co.uk/admin/assets/caice%20cheetah%20fan%20coil%20unit%20brochure%202008_07_01.pdf, last accessed: 23 December 2008.
- [Chi08] Chiticariu, L., Kolaitis, P. G., and Popa, L., Interactive generation of integrated schemas. In: *Proc. 2008 ACM SIGMOD international Conference on Management of Data*, Vancouver, Canada, June 09 - 12, 2008.
- [Chr98] Christopher, M. *Logistics and supply chain management: strategies for reducing cost and improving service*. 2nd ed., Financial Times Pitman, London 1998.
- [Chu90] Chung, C. DATAPLEX: an access to heterogeneous distributed databases. In: *Communications of the ACM*. pp.70-80, 1990.
- [Cla99] XML Path Language (XPath) Version 1.0. In: J. Clark and S. DeRose (Ed.): *W3C Recommendation*. Nov. 1999. [Online]. Available: <http://www.w3.org/TR/xpath>, last Accessed: 23 February 2008.
- [Cla99a] XSL Transformations (XSLT) Version 1.0. In: J. Clark (Ed.): *W3C Recommendation*. November 1999, [Online], Available: <http://www.w3.org/TR/xslt> , last accessed: 23 February 2008.
- [Com09] The Compare.com website, 2006 [Online]. Available: <http://www.compare.com/>, last accessed: 21 February 2009.
- [Con99] Conrad, S., Hasselbring, W., Hohenstein, U., Kutsche, R. D., Roantree, M., Saake, G. and Saltor, F. Engineering federated information systems: report of EEFIS '99 workshop. In: *SIGMOD Rec.* 28(3), pp.9-11, 1999.
- [Con05] Connolly, T. M. and Begg, C. E. *Database systems: a practical approach to design, implementation, and management*. 4th ed., Addison-Wesley, Harlow, New York, 2005.
- [Cox02] Cox, A. and Ireland, P. Managing construction supply chains: the common sense approach. In: *Engineering, Construction and Architectural Management*. 9(5/6), pp.409-418, 2002.
-

-
- [Cza04] From Open Grid Services Infrastructure to WS-Resource Framework: Refactoring & Evolution, version 1.1. In: K. Czajkowski, D. Ferguson, I. Foster, J. Frey, S. Graham, T. Maguire, D. Snelling and S. Tuecke (Ed.): *The WS-Resource Framework*. [Online]. Available: http://www.globus.org/wsrif/specs/ogsi_to_wsrif_1.0.pdf, last accessed 24 February 2008, 2004.
- [Dal08] Dalsgaard, E., Kjelstrøm, K., and Riis, J., A federation of web services for Danish health care. In: *Proc. 7th Symposium on Identity and Trust on the internet, (IDtrust '08)*, Gaithersburg, Maryland, March 04 - 06, 2008.
- [Dar07] Darrah, M., Van Scoy, F., and Plunkett, P. 2007. Enabling collaboration in high performance computing. In: *Proc. 8th ACM SIGITE Conference on information Technology Education*, Destin, Florida, USA, October 18 - 20, 2007.
- [Deu98] Deutsch, A., Fernandez, M., Florescu, D., Levy, A. and Suciu, D. XML-QL: A Query Language for XML. *World Wide Web Consortium*. August, 1998. [Online], Available: <http://www.w3.org/TR/NOTE-xml-ql/> Last accessed: 24 February 2008.
- [Dom05] The W3C Document Object Model (DOM) website, 2005. [Online]. Available: <http://www.w3.org/DOM/>, last accessed: 28 February 2008.
- [Don06] Dongarra, J. and Lastovetsky, A., An overview of heterogeneous high performance and Grid computing. In: DiMartino, B., Dongarra, J., Hoisie, A., Yang, L., Zima, H. (Eds.), *Engineering the Grid: Status and Perspective*, American Scientific Publishers, 2006
- [Dti95] Report on Small firms in Britain, *Department of Trade and Industry (DTI)*, HMSO, London 1995.
- [Duw96] Duwairi, R. M., Fiddian, N. J. and Gray, W. A. A Multiple View Definition System for Supporting Interoperability among Heterogeneous and Autonomous Databases. In: *In Proc. 10th ERCIM Workshop on Heterogeneous Information Management*, Prague, Czech Republic, 1996.
- [Dze05] Dzung, R.-J. and Chang, S.-Y. Learning search keywords for construction procurement. In: *Automation in Construction*. 14(1), pp. 45-58, 2005.
- [Edu99] Edum-Fotwe, F. T., McCaffer, R., Thorpe, A. and Majid, M. Z. A. Sub-contracting or Co-contracting: Construction Procurement in Perspective. In: *Proc. 2nd International Conference on Construction Industry Development*, National University of Singapore, pp.157-163, 27-29 October, 1999.
- [Edu01] Edum-Fotwe, F. T., Thorpe, A. and McCaffer, R. Information procurement practices of key actors in construction supply chains. In: *European Journal of Purchasing and Supply Management*. 7(3), pp.155-164, 2001.
-

-
- [Ega98] Egan, Sir J. Rethinking construction: the report of the Construction Task Force to the Deputy Prime Minister, John Prescott, on the scope for improving the quality and efficiency of UK construction. *Department of the Environment, Transport and Regions (DETR)*, 1998.
- [Eis99] Eisenberg, A. and Melton, J. SQL: 1999, formerly known as SQL3. In: *SIGMOD Rec.* 28(1), pp.131-138, 1999.
- [Eur03] The European DataGrid Project website, 2003. [Online]. Available: <http://edg-wp2.web.cern.ch/edg-wp2/> . Last accessed: 24 February 2008.
- [Fal04] XML Schema Part 0: Primer Second Edition. In: D. C. Fallside and P. Walmsley (Ed.): *W3C Recommendation*. October 2004, [Online]. Available: <http://www.w3.org/TR/xmlschema-0/> , Last accessed: 24 February 2008.
- [Fan98] Fankhauser, P., Gardarin, G., Lopez, M., Munoz, J. and Tomasic, A. Experiences in Federating Databases: From IRO-DB to MIRO-Web. In: *Proc. 24th annual International Conference on Very Large Data Bases (VLDB'98)*, New York City, New York, USA, pp.655-658, 1998.
- [Fan08] Fan, H. and Liu G., Study on Proteomics Data Integration in Web Environments, In: *The 2nd International Conference on Bioinformatics and Biomedical Engineering, (ICBBE 2008.)*, pp.192-195, 16-18 May 2008.
- [Fen08] Feng, T., Xiao-bing, H., Feng-bo, W., The Heterogeneous Data Integration Based on XML in Coal Enterprise, *International Symposium on Computer Science and Computational Technology, (ISCST '08.)*, vol.1, pp.438-441, 20-22 Dec. 2008.
- [Fer03] Ferreira, L., Jacob, B., Slevin, S., Brown, M., Sundararajan, S., Lepasant, J. and Bank, J. *Globus Toolkit 3.0 Quick Start*. IBM Redbooks, IBM Corp., 2003. [Online]. Available: <http://www.redbooks.ibm.com/redpapers/pdfs/redp3697.pdf>, last accessed: 24 February 2008.
- [Fin94] Finin, T., Fritzson, R., McKay, D. and McEntire, R. KQML as an agent communication language. In: *Proc. Third international Conference on Information and Knowledge Management (CIKM '94)*, Gaithersburg, Maryland, United States, pp.456-463, November 29 - December 02, 1994.
- [Fos98] Foster, I. and Kesselman, C. The Globus Project: A Status Report. In: *Proc. Seventh Heterogeneous Computing Workshop*, pp.4-19, 1998.
- [Fos99] Foster, I. and Kesselman, C. *Computational Grids, Chapter 2 of The Grid: Blueprint for a New Computing Infrastructure*, Morgan-Kaufman, 1999.
- [Fos01] Foster, I., Kesselman, C. and Tuecke, S. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. In: *International Journal of Supercomputer Applications*. 15(3), 2001.
-

- [Fos02] Foster, I., Kesselman, C., Nick, J. and Tuecke, S. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. In: *Open Grid Service Infrastructure WG, Global Grid Forum*. June 22, 2002.
- [Fos02a] Foster, I., Voeckler, J., Wilde, M. and Zhao, Y. Chimera: A Virtual Data System for Representing, Querying, and Automating Data Derivation. In: *Proc. 14th Conference on Scientific and Statistical Database Management*, July, 2002.
- [Fra98] Franks, J. *Building procurement systems: a client's guide*. 3rd ed., Addison Wesley Longman Limited, Harlow, Essex, England, 1998.
- [Gag02] Gagliardi, F., Jones, B., Reale, M. and Burke, S. European DataGrid Project: Experiences of Deploying a Large Scale Testbed for E-science Applications. In: M. Calzarossa and S. Tucci (Ed.): *Performance Evaluation of Complex Systems: Techniques and Tools, Performance 2002, Tutorial Lectures, Lecture Notes In Computer Science*. vol. 2459. Springer-Verlag, London, pp.480-500, 2002.
- [Gan07] The Ganglia Monitoring System, 2007. [Online]. Available: <http://ganglia.sourceforge.net/>, last accessed 23 December 2007
- [Gar95] Gardarin, G., Gannouni, G., Finance, B., Fankhausera, P., Klas, W., Pastrea, D., Legoff, R. and Ramfos, A. IRO-DB: a distributed system federating object and relational databases. In: O. A. Bukhres and A. K. Elmagarmid (Eds.): *Object-Oriented Multidatabase Systems: A Solution For Advanced Applications*. Prentice Hall International (UK) Ltd., Hertfordshire, UK, pp.684-712, 1995.
- [Gar97] Garcia-Molina, H., Papakonstantinou, Y., Quass, D., Rajaraman, A., Sagiv, Y., Ullman, J., Vassalos, V. and Widom, J. The TSIMMIS Approach to Mediation: Data Models and Languages. In: *Journal of Intelligent Information Systems*. 8(2), pp.117-132, 1997.
- [Gar99] Gardarin, G., Sha, F. and T. Dang-Ngoc. XML-based Components for Federating Multiple Heterogeneous Data Sources. In: *Proc. 18th international Conference on Conceptual Modeling*, J. Akoka, M. Bouzeghoub, I. Comyn-Wattiau, and E. Métais (Eds.), LNCS, vol. 1728. Springer-Verlag, London, pp.506-519, 1999.
- [Gen97] Genesereth, M. R., Keller, A. M. and Duschka, O. M. Infomaster: an information integration system. In: *Proc. ACM SIGMOD international Conference on Management of Data*, Tucson, Arizona, United States, May 11 - 15, 1997.
- [Gen07] The GENIUS Portal website, 2007 [Online]. Available: <http://egee.cesnet.cz/en/user/genius.html> Last accessed: 23 December 2007.
-

-
- [Gie89] Gielingh, W. F. General AEC Reference Model (GARM), ISO TC184/SC4/WG1 Document N329. 1989.
- [Gil05] Gillespie, C. S., Proctor, C. J., Shanley, D. P., Wilkinson, D. J., Boys, R. J. and Kirkwood, T. B. L. Web-services for the biology community: the BASIS project. In: *Proc. of the UK e-Science All Hands Meeting 2005*, Nottingham UK, 19th - 22nd September, 2005.
- [Glo09] The Globus Alliance Website, 2009. [Online]. Available: <http://www.globus.org/>, last accessed: 22 February 2009.
- [Glo09a] The Globus Consortium Website, 2009. [Online]. Available: <http://www.globusconsortium.org/>, last accessed: 22 February 2009.
- [Glo09b] The Globus Toolkit 3.0 - Java Programmer's Guide: Core Framework, 2003. [Online]. Available: http://www.globus.org/toolkit/docs/3.0/java_programmers_guide.html, last accessed: 23 February 2009.
- [Gob03] Goble, C., Wroe, C. and Stevens, R. The myGrid project: services, architecture and demonstrator. In: *Proc. UK e-Science All Hands Meeting (AHM'03)*, Nottingham, UK, Sept. 2003.
- [Gra02] Graham, S., Simeonov, S., Boubez, T., Davis, D., Daniels, G., Nakamura, Y. and Neyama, R. *Building Web Services with Java: Making sense of XML, SOAP, WSDL, and UDD*. Sams Publishing, 2002.
- [Gra07] The P-GRADE Grid Portal website, 2007. [Online]. Available: <http://egee.cesnet.cz/en/user/p-grade.html> Last accessed: 23 December 2007.
- [Gra08] Grant, A., Antonioletti, M., Hume, A.C., Krause, A., Dobrzelecki, B., Jackson, M.J., Parsons, M., Atkinson, M.P. and Theocharopoulos, E. OGSA-DAI: Middleware for Data Integration: Selected Applications, *IEEE Fourth International Conference on eScience*, pp.343-343, December 2008.
- [Gru91] Gruber, T. R. The Role of Common Ontology in Achieving Sharable, Reusable Knowledge Bases. In: J. A. Allen, R. Fikes and E. Sandewall (Eds.): *Principles of Knowledge Representation and Reasoning*. Morgan Kaufmann, 1991.
- [Gru93] Gruber, T. R. A translation approach to portable ontology specifications. In: *Knowledge Acquisition*. vol. 5 pp.199-220, 1993.
- [Haj08] Hajmoosaei, A. and Abdul-Kareem, S., An Approach for Mapping of Domain-based Local Ontologies. In: *Proc. International Conference on Complex, intelligent and Software intensive Systems (CISIS)*, pp.865-870, 4-7 March 2008.
-

-
- [Han99] Handfield, R. B. *Introduction to supply chain management*. ed., Prentice Hall, Upper Saddle River, N.J., 1999.
- [Här97] Härder, T., Sauter, G. and Thomas, J. Design and Architecture of the FDBS Prototype INFINITY. In: *Int. CAiSE'97 Workshop, Engineering Federated Database Systems (EFDBS'97)*, Barcelona, pp.57-68, 1997.
- [Har01] Harris, F. and McCaffer, R. *Modern construction management*. 5th ed., Blackwell Science, Oxford, 2001.
- [Has00] Hasselbring, W., Heuvel, W. J. v. d., Houben, G. J., Kutsche, R. D., Rieger, B., Roantree, M. and Subieta, K. Research and practice in federated information systems. In: *SIGMOD Rec.* 29(4), pp.16-18, 2000.
- [Hei85] Heimbigner, D. and McLeod, D. A federated architecture for information management. In: *ACM Transactions on Office Information Systems*. 3(3), pp.253-278, July, 1985.
- [Hsi92] Hsiao, D. K. Federated databases and systems: part I --- a tutorial on their data sharing. In: *The VLDB Journal*. 1(1), pp.127-180, 1992.
- [Hsi92a] Hsiao, D. K. Federated databases and systems: part II --- a tutorial on their resource consolidation. In: *The VLDB Journal*. 1(2), pp.285-310, 1992.
- [Hua94] Huang, J.-W. MultiBase: A Heterogeneous Multidatabase Management System. In: *Proc. Eighteenth Annual International Computer Software and Applications Conference (COMPSAC'94)*, pp.332-339, 1994.
- [Hua06] Huang, W., Liu, J., Abali, B., and Panda, D. K. 2006. A Case for High Performance Computing with Virtual Machines. In: *Proc 20th Annual international Conference on Supercomputing*, Cairns, Queensland, Australia, June 28 - July 01, 2006.
- [Hul95] Reference Architecture for the Intelligent Integration of Information. R. Hull, R. King, Y. Arens and M. Siegel (Eds.): *Technical Report, ARPA*. 1995. [Online]. Available: http://ise.gmu.edu/I3_Arch/, last accessed 24 February 2008.
- [Hwa94] Hwang, S.-Y., Lim, E.-P., Yang, H.-R., Musukula, S., Mediratta, K., Ganesh, M., Clements, D., Stenoien, J. and Srivastava, J. The MYRIAD federated database prototype. In: *Proc. ACM SIGMOD International Conference on Management of data*, Minneapolis, Minnesota, United States, 1994.
- [Ies05] Annual Survey of Small Business: UK 2005, *Institute of Employment Studies*, University of Sussex, 2005. [Online] Available: <http://www.berr.gov.uk/files/file38247.pdf>. Last Accessed: 23 December 2007.
-

- [Iis09] The Official Microsoft Internet Information Services (IIS) web site, 2008. [Online]. Available: <http://www.iis.net/>, last accessed 02 January 2009.
- [ISO94] Industrial automation systems and integration -- Product data representation and exchange -- Part 1: Overview and fundamental principle. *International Organisation for Standardization (ISO), ISO 10303-1*, 1st edition, 1994.
- [ISO94a] Industrial automation systems and integration -- Product data representation and exchange -- Part 11: Description methods: The EXPRESS language reference manual, *International Organisation for Standardization (ISO), ISO 10303-11*, 1st edition, 1994.
- [ISO98] Industrial automation systems and integration -- Product data representation and exchange -- Part 22: Implementation methods: Standard data access interface, *International Organisation for Standardization (ISO), ISO 10303-22* 1st edition, 1998.
- [Jav07] The Java technology web site, 2007. [Online]. Available: <http://java.sun.com/>, last accessed: 23 December 2007
- [Jit05] Jithesh, P. V., Kelly, N., Donachy, P., Harmer, T., Perrott, R., McCurley, M., Townsley, M., J. Johnston and McKee, S. GeneGrid: grid based solution for bioinformatics application integration and experiment execution. In: *Proc. 18th IEEE Symposium on Computer-Based Medical Systems*, pp.523-528, 2005.
- [Joh93] Johannesson, P. A logical basis for schema integration. In: *Third International Workshop on Research Issues in Data Engineering, Proc. Interoperability in Multidatabase Systems*, pp.86-95, 1993.
- [Joi04] Joita, L., Pahwa, J. S., Burnap, P., Gray, A., Rana, O. and Miles, J. Supporting Collaborative Virtual Organisations in the Construction Industry via the Grid. In: *Proc. of the UK e-Science All Hands Meeting, Nottingham UK, 31st Aug-3rd Sep, 2004*.
- [Joi04a] Joita, L., Rana, O., Burnap, P., Pahwa, J. S., Gray, A. and Miles, J. A Grid-Enabled Security Framework for Collaborative Virtual Organisations. In: *Proc. Of the 5th IFIP Working Conference on Virtual Enterprises (PRO-VE) (alongside the IFIP World Computer Congress)*, Toulouse, France, 23-26 August 2004.
- [Jon00] Jones, A. C., Xu, X., Pittas, N., Gray, W. A., Fiddian, N. J., White, R. J., Robinson, J., Bisby, F. A. and Brandt, S. M. SPICE: A Flexible Architecture for Integrating Autonomous Databases to Comprise a Distributed Catalogue of Life. In: *Proc. 11th International Conference on Database and Expert Systems Applications*, London, UK, pp.981-992, September, 2000.
-

References

- [Jun08] Junfang Z., Wancheng N., Lin C., Yu L., Modeling RFID Data to Support Information Sharing, *In: Third International Conference on Convergence and Hybrid Information Technology, (ICCIT '08.)*, vol.1, pp.1137-1141, 11-13 November 2008
- [Kar95] Karlapalem, K., Li, Q. and Shum, C. HODFA: an architectural framework for homogenizing heterogeneous legacy databases. *In: SIGMOD Rec.* 24(1), pp.15-20, 1995.
- [Kel05] Kelly, N., Jithesh, P. V., Donachy, P., Harmer, T. J., Perrott, R. H., McCurley, M., Townsley, M., Johnston, J. and McKee, S. GeneGrid: a commercial grid service oriented virtual bioinformatics laboratory. *In: IEEE International Conference on Services Computing*, pp.43- 50, 2005.
- [Kif89] M. Kifer and G. Lausen, "F-logic: a higher-order language for reasoning about objects, inheritance, and scheme," *Proc. of the 1989 ACM SIGMOD international Conference on Management of Data, Portland, Oregon, United States, J. Clifford, B. Lindsay, and D. Maier, Eds. SIGMOD '89*, pp134-146, ACM Press, New York, 1989.
- [Kif89] Kifer, M. and Lausen, G. F-logic: a higher-order language for reasoning about objects, inheritance, and scheme. *In: Proc. ACM SIGMOD international Conference on Management of Data, Portland, Oregon, United States*, pp.134-146, 1989.
- [Kin85] King, R. and McLeod, D. A database design methodology and tool for information systems. *In: ACM Transactions on Office Information Systems.* 3(1), pp.2-21, 1985.
- [Kon04] Kong, S. C. W., Li, H., Hung, T. P. L., Shi, J. W. Z., Castro-Lacouturec, D. and Skibniewskid, M. Enabling Information Sharing between E-commerce Systems for Construction Material Procurement. *In: Automation in Construction*, pp.261-276, 2004.
- [Kor98] Kornelius, L. and Wamelink, J. W. F. The virtual corporation: learning from construction. *In: Supply Chain Management.* 3(4), pp.193-202, 1998.
- [Koz07] Kozlova, I., Ritter, N., and Husemann, M., Providing semantically equivalent, complete views for multilingual access to integrated data. *In: 26th international Conference on Conceptual Modeling, Auckland, New Zealand, November, 2007.*
- [Kra08] Krawczyk, S. and Bubendorfer, K., Grid resource allocation: allocation mechanisms and utilisation patterns. *In Proc. of the Sixth Australasian Workshop on Grid Computing and E-Research*, vol. 82, Wollongong, NSW, Australia, January 2008, W. Kelly and P. Roe, (Eds.), *Conferences in Research and Practice in Information Technology Series*, vol. 333. Australian Computer Society, Darlinghurst, Australia, 73-81.
-

References

- [Kum00] Kumaraswamy, M., Palaneeswaran, E. and Humphreys, P. Selection matters – in construction supply chain optimisation. In: *International Journal of Physical Distribution & Logistics Management*. 30(7/8), pp.661-680, 2000.
- [Kum01] Kumaraswamy, M. M. and Dissanayaka, S. M. Developing a decision support system for building project procurement. In: *Building and Environment*. 36(3), pp.337-349, 2001.
- [Lam93] Lamming, R. *Beyond Partnership: Strategies for Innovation and Lean Supply*. Prentice-Hall International (UK) Limited, Hertfordshire, 1993.
- [Lat94] Latham, S. M. *Constructing the team: Joint Review of Procurement and Contractual Arrangements in the United Kingdom Construction Industry: final report*. HMSO, London, 1994.
- [Law01] Lawrence, R. and Barker, K. Integrating relational database schemas using a standardized dictionary. In: *Proc. ACM symposium on Applied computing*, Las Vegas, Nevada, United States, pp.225-230, 2001.
- [Lee97] Lee, Y., Liu, L. and Pu, C. Towards interoperable heterogeneous information systems: an experiment using the DIOM approach. In: *Proc. ACM Symposium on Applied Computing (SAC '97)*, San Jose, California, United States, pp.1997, 1997.
- [Lee02] Lee, K., Min, J. and Park, K. A Design and Implementation of XML-Based Mediation Framework (XMF) for Integration of Internet Information Resources. In: *Proc. 35th Annual Hawaii International Conference on System Sciences (HICSS'02)*, IEEE Computer Society, Washington, DC, 202, 2002.
- [Lee08] Lee, J., Nam, D., Hwang, S., Byeon, O., A Grid-Enabled Problem Solving Environment for Supporting Collaborative Aerodynamic Engineering Process, In: *Proc. IEEE Fourth International Conference on eScience, (eScience '08.)*, pp.770-777, 7-12 December 2008.
- [Leh04] Lehti, P. and Fankhauser, P. XML Data Integration with OWL: Experiences and Challenges. In: *Proc. International Symposium on Applications and the Internet (SAINT'04)*, Los Alamitos, CA, USA, 2004.
- [Lim95] Lim, E., Hwang, S., Srivastava, J., Clements, D. and Ganesh, M. Myriad: Design and Implementation of a Federated Database Prototype. In: *Software - Practice and Experience*. 25(5), pp.533-562, 1995.
- [Lit85] Litwin, W. An overview of the multidatabase system MRDSM. In: *Proc. ACM Annual Conference on the Range of Computing: Mid-80's Perspective*, Denver, Colorado, United States, ACM '85. ACM Press, New York, NY, 1985.
-

- [Liu95] Liu, L. and Pu, C. The distributed interoperable object model and its application to large-scale interoperable database systems. In: *Proc. of the Fourth international Conference on information and Knowledge Management (CIKM '95)*, Baltimore, Maryland, United States, November 29 - December 02, 1995.
- [Liu96] Liu, L. and Pu, C. An Object-Oriented Approach to Interoperation of Heterogeneous Information Sources. In: *Proc. Seventh International Hong Kong Computer Society Database Workshop*, Hong Kong, pp.49-63, 1996.
- [Lev96] Levy, A. Y., Rajaraman, A. and Ordille, J. J. Querying Heterogeneous Information Sources Using Source Descriptions. In: *Proc. Twenty-second International Conference on Very Large Databases*, Bombay, India, pp.251-262, 1996.
- [Loo06] The Loom Project Home Page, 2006 [Online.] Available: <http://www.isi.edu/isd/LOOM/LOOM-HOME.html> . Last accessed: 10 December 2006.
- [Lui93] Luiten, G., Froese, T., Bjork, B.-C., Cooper, G., Junge, R., Karstila, K. and Oxman, R. An Information Reference Model For Architecture, Engineering, And Construction. In: *Proc. First International Conference on the Management of Information Technology for Construction*, Singapore, pp.391-406, 1993.
- [Luk96] Lukovic, I. and Mogin, P. An approach to relational database schema integration. In: *IEEE International Conference on Systems, Man, and Cybernetics*. vol.4, pp.3210-3215, 1996.
- [Lyn08] Lynden, S., Pahlevi, S. M., and Kojima, I., Service-based Data Integration using OGSA-DQP and OGSA-WebDB, In: *Proc. The 9th IEEE/ACM International Conference on Grid Computing (Grid 2008)*, Tsukuba, Japan, September 2008. IEEE Computer Society Press.
- [Lyn09] Lynden, S., Mukherjee, A., Hume, A.C., Fernandes, A.A.A., Paton, N.W., Sakellariou, R. and Watson, P. The design and implementation of OGSA-DQP: A service-based distributed query processor, *Future Generation Computer Systems*, vol. 25(3), pp. 224-236, March 2009.
- [Lys00] Lysons, K. *Purchasing and supply chain management*. 5th ed., Pearson Education Limited, England, 2000.
- [Mah04] Mahmoud-Jouini, S. B., Midler, C. and Garel, G. Time-to-market vs. time-to-delivery Managing speed in Engineering, Procurement and Construction projects. In: *International Journal of Project Management*. 22(5), pp.359-367, 2004.
- [Mas02] Masterman, J. W. E. *An introduction to building procurement systems*. 2nd ed., Spon Press, London, New York, 2002.
-

- [McC99] McCreedia, M. and Rice, R. E. Trends in analyzing access to information. Part I: cross-disciplinary conceptualizations of access. In: *Information Processing & Management*. 35(1), pp.45-76, 1999.
- [McH97] McHugh, J., Abiteboul, S., Goldman, A. R., Quass, D. and Widom, J. Lore: a database management system for semistructured data. In: *SIGMOD Rec.* 26 (3), pp.54-66, 1997.
- [McJ06] McJones, P. "Multics Relational Data Store (MRDS)" [Online]. Available: http://www.mcjones.org/System_R/mrds.html, 2006 Last accessed: 14 October 2006.
- [Mic07] The Microsoft .NET technology website, 2007. [Online]. Available: <http://www.microsoft.com/net/default.aspx>, last accessed: 23 December 2007.
- [Mic07a] The Microsoft SQL Server website, 2007. [Online]. Available: <http://www.microsoft.com/sql/default.msp>, last accessed 23 December 2007.
- [Mil02] Miles J.C., Rana, O. and Gray, W.A. Collaborative Virtual Teams. *Research project proposal*, Cardiff University, 2002.
- [Mil04] Miles, J., Joita, L., Pahwa, J. S., Burnap, P., Gray, A. and Rana, O. Collaborative Engineering Virtual Teams in a Grid Environment Supporting Consortia in the Construction Industry. In: *Proc. of the 10th International Conference on Computing in Civil and Building Engineering (ICCCBE)*, Weimar, Germany, 2-4 June, 2004.
- [Myl96] Mylopoulos, J., Chaudhri, V., Plexousakis, D., Shrufi, A. and Topologlou, T. Building knowledge base management systems. In: *The VLDB Journal*. 5(4), pp.238-263, 1996.
- [Nak08] Nakanishi, T., Zettsu, K., Kidawara, Y., Kiyoki, Y., Approaching the Interconnection of Heterogeneous Knowledge Bases on a Knowledge Grid, In: *Proc. Fourth International Conference on Semantics, Knowledge and Grid, (SKG '08.)*, pp.71-78, 3-5 December 2008.
- [Nam02] Nam, Y., Goguen, J. and Wang, G. A metadata integration assistant generator for heterogeneous distributed databases. In: *Proc. International Conference on Ontologies, DataBases, and Applications of Semantics for Large Scale Information Systems*, R. Meersman and Z. Tari (eds.), LNCS, vol. 2519. Springer-Verlag, London, pp.1332-1344, 2002.
- [Nin97] Nink, U. Using the STEP standard and databases in science. In: *Proc. Ninth International Conference on Scientific and Statistical Database Management*, Olympia, WA, pp. 196-207, Aug, 1997.
-

- [Oas09] The OASIS Web Services Resource Framework (WSRF) TC, 2009. [Online]. Available: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf, last accessed: 22 February 2009
- [Ogs07] OGSA-DAI User Documentation. [Online]. Available: <http://www.ogsadai.org.uk/documentation/>, Last accessed: 17 December 2007
- [OMG06] The OMG's CORBA Website. [Online]. Available: <http://www.corba.org/>
- [Pah04] Pahwa, J. S., Burnap, P., Joita, L., Gray, A., Rana, O. and Miles, J. Creating a Virtual Distributed Database - Data Definition and Search Model for Collaborative Virtual Teams in the Construction Industry. In: *21st Annual British National Conference on Databases, Proceedings Volume 2*, Edinburgh, UK, pp.3-17, 7-9 July, 2004.
- [Pah06] Pahwa, J. S., Brewer, P., Sutton, T., Yesson, C., Burgess, M., Xu, X., Jones, A. C., R. J. White, Gray, W. A., Fiddian, N. J., Bisby, F. A., Culham, A., Caithness, N., Scoble, M., Williams, P. and Bhagwat, S. Biodiversity World: A Problem-Solving Environment for Analysing Biodiversity Patterns. In: *Proc. 6th IEEE International Symposium on Cluster Computing and the Grid (CCGRID 2006)*, Singapore, 2006.
- [Pah06a] Pahwa, J. S., White, R. J., Jones, A. C., Burgess, M., Gray, W. A., Fiddian, N. J., Sutton, T., Brewer, P., Yesson, C., Caithness, N., Culham, A., Bisby, F. A., Scoble, M., Williams, P. and Bhagwat, S. Accessing Biodiversity Resources in Computational Environments from Workflow Applications. In: *Workshop on Workflows in Support of Large-Scale Science (in conjunction with the 15th IEEE International Symposium on High Performance Distributed Computing)*, Paris, France, 2006.
- [Pah06b] Pahwa, J. S., Burnap, P., Gray, W. A. and Miles, J. MDSSF - A Federated Architecture for Product Procurement. In: *17th International Conference on Database and Expert Systems Applications (DEXA '06)*, Andrzej Frycz Modrzewski Cracow College, Krakow, Poland, 4 -8 September 2006.
- [Pal03] Palopoli, L., Terracina, G. and Ursino, D. DIKE: a system supporting the semi-automatic construction of cooperative information systems from heterogeneous databases. In: *Software—Practice & Experience archive*. 33(9), pp.847 - 884, 2003.
- [Pap95] Papakonstantinou, Y., Garcia-Molina, H. and Widom, J. Object Exchange Across Heterogeneous Information Sources. In: *Proc. Eleventh international Conference on Data Engineering (ICDE'95)*, Washington, DC, pp.251-260, March 06 - 10, 1995.
-

References

- [Pap96] Papakonstantinou, Y., Garcia-Molina, H. and Ullman, J. MedMaker: A Mediation System Based on Declarative Specifications. In: *Proc. Twelfth International Conference on Data Engineering (ICDE'96)*, IEEE Computer Society, Washington DC, pp.132-141, 1996.
- [Pap96a] Papakonstantinou, Y., Abiteboul, S. and Garcia-Molina, H. Object Fusion in Mediator Systems. In: *Proc. 22nd International Conference on Very Large Data Bases*, pp.413-424, 1996.
- [Par98] Parent, C. and Spaccapietra, S. Issues and approaches of database integration. In: *Communications of the ACM*. 41(5), pp.166-178, 1998.
- [Pat99] Patterson, J. L., Forker, L. B. and Hanna, J. B. Supply chain consortia: the rise of transcendental buyer-supplier relationships. In: *European Journal of Purchasing and Supply Management*. vol. 5 pp.85 - 93, 1999.
- [Pie97] Pietroforte, R. Communication and governance in the building process. In: *Construction Management and Economics*. vol.15, pp.71-82, 1997.
- [Por98] Porter, M. E. *Competitive advantage of nations*. Macmillan Business, Basingstoke, 1998.
- [Pri09] The Pricerunner.com website, 2009. [Online]. Available: <http://www.pricerunner.co.uk/>, last accessed: 21 February 2009.
- [Qua08] Quartel, D., Pokraev, S., Pessoa, R.M., van Sinderen, M., Model-Driven Development of a Mediation Service, In: *Proc. 12th International IEEE Enterprise Distributed Object Computing Conference, (EDOC '08.)*, pp.117-126, 15-19 September 2008.
- [Rad95] Radeke, E., Böttger, R., Burkert, B., Engel, Y., Kachel, G., Kolmschlag, S. and Nolte, D. Efendi: federated database system of Cadlab. In: *Proc. ACM SIGMOD International Conference on Management of Data*, San Jose, California, United States, 1995.
- [Ran05] Rana, O., Hilton, J., Joita, L., Burnap, P., Pahwa, J. S., Miles, J. and Gray, W. A. Secure Virtual Organisations: Protocols and Requirements. In: S. Paulus, N. Pohlmann, and H. Reimer (eds.), *Information Security Solutions Europe (ISSE) 2005 - The Independent European ICT Security Conference and Exhibition*, Budapest, Hungary, 27-29 September 2005.
- [Red94] Reddy, M. P., Prasad, B. E., Reddy, P. G. and Gupta, A. A Methodology for Integration of Heterogeneous Databases. In: *IEEE Transactions on Knowledge and Data Engineering*. 6(6), pp.920-933, 1994.
- [Row99] Procurement Systems: A Guide to Best Practice in Construction. In: S. Rowlinson and P. McDermott (Ed.): *E & FN Spon*. London: 1999.
-

References

- [Sar08] Sarraipa, J., Silva, J., Jardim-Goncalves, R., Monteiro, A., MENTOR — A methodology for enterprise reference ontology development, In: *Proc. 4th International IEEE Conference on Intelligent Systems, IS '08.*, vol.1, pp.6-32-6-40, 6-8 Sept. 2008.
- [Sau97] Saunders, M. *Strategic purchasing and supply chain management*. Financial Times, Pitman, London 1997.
- [Sek06] The SEEK Project Proposal, 2004. [Online]. Available: <http://seek.ecoinformatics.org/Wiki.jsp?page=SEEKProjectProposal> Last accessed: 21 October 2006.
- [Sfc02] Accelerating Change: A report by the Strategic Forum for Construction chaired by Sir John Egan, *Strategic Forum for Construction*, London 2002
- [She90] Sheth, A. P. and Larson, J. A. Federated database systems for managing distributed, heterogeneous, and autonomous databases. In: *ACM Computing Surveys*. 22 (3), pp.183-236, 1990.
- [Sho07] Shore, J. and Warden, S. *The Art of Agile Development* (Paperback), O'Reilly Media Inc., Oct 2007.
- [Sin97] Singh, M. P., Cannata, P. E., Huhns, M. N., Jacobs, N., Ksiezyk, T., Ong, K., Sheth, A. P., Tomlinson, C. and Woelk, D. The Carnot Heterogeneous Database Project: Implemented Applications. In: *Distrib. Parallel Databases*. 5(2), pp.207-225, 1997.
- [Sin05] Singh, G., Deelman, E., Mehta, G., Vahi, K., Su, M.-H., Berriman, G. B., Good, J., Jacob, J. C., Katz, D. S., Lazzarini, A., Blackburn, K. and Koranda, S. The Pegasus Portal: Web Based Grid Computing. In: *Proc. of the ACM symposium on Applied computing*, Santa Fe, New Mexico, pp.680–686, 2005.
- [Som01] Sommerville, I. *Software engineering*. 6th ed., Addison-Wesley, Harlow, New York, 2001.
- [Spa92] Spaccapietra, S., Parent, C. and Dupont, Y. Model independent assertions for integration of heterogeneous schemas. In: *The VLDB Journal*. 1(1), pp.81-126, 1992.
- [Spi03] The Spitfire Task webpage, 2003. [Online]. Available: <http://edg-wp2.web.cern.ch/edg-wp2/spitfire/index.html> . Last accessed: 23 December 2007
- [Su95] Su, S. Y. W., Lam, H., Arroyo-Figueroa, J., Yu, T. and Yang, Z. An extensible knowledge base management system for supporting rule-based interoperability among heterogeneous systems. In: *Proc. Fourth International Conference on Information and Knowledge Management (CIKM '95)*, Baltimore, Maryland, United States, 1995.
-

References

- [Sub95] Subrahmanian, V. S., Adali, S., Brink, A., Emery, R., Lu, J., Rajput, A., Rogers, T., Ross, R. and Ward, C. Hermes: A heterogeneous reasoning and mediator system. In: *Technical report, University of Maryland*. 1995.
- [Tan00] Tan, J., Zaslavsky, A. and Bond, A. Meta object approach to database schema integration. In: *Proc. International Symposium on Distributed Objects and Applications*, pp.145-154, 2000.
- [Tay03] Taylor, I., Shields, M., Wang, I. and Philp, R. Grid Enabling Applications using Triana. In: *Workshop on Grid Applications and Programming Tools*, Seattle, USA, 2003.
- [Tha05] Thain, D., Tannenbaum, T. and Livny, M. Distributed Computing in Practice: The Condor Experience. In: *Concurrency and Computation: Practice and Experience*. 17(2-4), pp.323–356, 2005.
- [Tho06] XML Schema Part 1: Structures Second Edition. In: H. S. Thompson, D. Beech, M. Maloney and N. Mendelsohn (Ed.): *W3C Recommendation*. October 2004. [Online]. Available: <http://www.w3.org/TR/xmlschema-1/> , Last accessed: 12 October 2006.
- [Tom88] Tomlinson, C., Kim, W., Scheevel, M., Singh, V., Will, B. and Agha, G. Rosette: An object-oriented concurrent systems architecture. In: *Proc. ACM SIGPLAN Workshop on Object-Based Concurrent Programming*, San Diego, California, United States, pp.91-93, 1988.
- [Tom98] Tomasic, A., Raschid, L. and Valduriez, P. Scaling Access to Heterogeneous Data Sources with DISCO. In: *IEEE Transactions on Knowledge and Data Engineering*. 10(5), pp.808-823, 1998.
- [Tue03] Open Grid Services Infrastructure (OGSI) Version 1.0. In: S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. K. T. Maquire, T. Sandholm, D. Snelling and P. Vanderbilt (eds.): *Open Grid Services Infrastructure - Working Group*. 2003.
- [Vid94] Vidal, V. M. P. and Winslett, M. Preserving update semantics in schema integration. In: *Proc. Third International Conference on Information and knowledge management*, Gaithersburg, Maryland, United States, pp.263-271, 1994.
- [Vie00] Vieira, R. *Professional SQL Server 2000 Programming* Wrox Press Ltd., Birmingham, UK, 2000.
- [Vis08] Visual Studio .NET 2003 Solution Center website, 2008. [Online]. Available: <http://support.microsoft.com/ph/3040>, last accessed: 02 January 2009.
-

- [Wal97] Walker, D. H. T. and Betts, M. Information Technology Foresight: The Future Application of the World Wide Web in Construction. In: *CIB W78 Workshop, Information Technology Support for Construction Process Re-Engineering (IT-CPR97)*, James Cook University, Cairns, Queensland, 1997.
- [Wal99] Walker, D. H. T. and Lloyd-Walker, B. M. Organisational learning as a vehicle for improved building procurement. In: S. Rowlinson and P. McDermott (Ed.): *Procurement Systems: A Guide to Best Practice in Construction*. E & FN Spon, London, 1999.
- [Wal99a] Walker, D. H. T. and Rowlinson, S. Use of World Wide Web technologies and procurement process implications. In: S. Rowlinson and P. McDermott (Ed.): *Procurement Systems: A Guide to Best Practice in Construction*. E & FN Spon, London, 1999.
- [Wan08] Wang, X, Huang, L., Zhang, Y., A Grid Middleware DISQ for Data Integration, In: *Proc. International Conference on Computer Science and Software Engineering*, vol.3, pp.62-65, December 2008.
- [Wel07] The Welsh e-Science Centre website. 2007. [Online]. Available: <http://www.wesc.ac.uk/>, last accessed 23 December 2007.
- [Wen01] Weng, R. and Zhu, Y. aecXML Framework. 2001. [Online]. Available: <http://www.iai-na.org/aecxml/documents.php>, last accessed 09 November 2006.
- [Wie92] Wiederhold, G. Mediators in the architecture of future information systems. In: *Computer*. 25(3), pp.38-49, 1992.
- [Wie93] Wiederhold, G. Intelligent integration of information. In: *Proc. ACM SIGMOD International Conference on Management of Data*, Washington, D.C. United States, May 25 - 28, 1993.
- [Wil98] Wilson, P. R., "STEP and EXPRESS" [Online]. Available: <http://deslab.mit.edu/DesignLab/dicpm/step.html>, 1998. Last accessed: 03 October 2006.
- [Wix98] Wix, J. and Liebich, T. Industry foundation classes: some business questions examined. In: *Proc. of the second EC-PPM conference*, Watford, UK, Oct 19-21, 1998.
- [Woe93] Woelk, D., Cannata, P., Huhns, M., Shen, W.-M. and Tomlinson, C. Using Carnot for enterprise information integration. In: *Proc. Second International Conference on Parallel and Distributed Information Systems*. pp.133-136, 1993.
- [Wsr07] The WS-Resource Framework, 2007. [Online] Available: <http://www.globus.org/wsrf/>, last accessed 23 December 2007.
-

- [WWW07] The World Wide Web (W3C) Consortium website. [Online]. Available: <http://www.w3c.org/>. Last Accessed: 23 December 2007.
- [Wys03] Wyss, C. M., James, A., Hasselbring, W., Conrad, S. and Höpfner, H. Report on the Engineering Federated Information Systems 2003 workshop (EFIS 2003). In: *SIGSOFT Softw. Eng. Notes*. 29(2), pp.1-3, 2003.
- [XMI06] Object Management Group, The XML Metadata Interchange (XMI) Specification v2.1, [Online]. Available: <http://www.omg.org/technology/documents/formal/xmi.htm>, 2006, Last Accessed: 10 October 2006.
- [XML07] The Extensible Markup Language (XML). [Online]. Available <http://www.w3c.org/XML/> Last Accessed: 23 December 2007.
- [Xu02] Xu, X., Jones, A. C., Gray, W. A., Fiddian, N. J., White, R. J. and Bisby, F. A. Design and performance evaluation of a web-based multi-tier federated system for a catalogue of life. In: *Proc. 4th International Workshop on Web Information and Data Management*, McLean, Virginia, USA, pp.104-107, 2002.
- [Yan07] Yang, H., Ye, F., and Liu, Z. 2007. Resource discovery in federated systems with voluntary sharing. In: *Proc. 2007 ACM/IFIP/USENIX international Conference on Middleware Companion*, Newport Beach, California, November 26 - 30, 2007.
- [Yeo02] Yeo, K. T. and J.H.Ning. Integrating supply chain and critical chain concepts in engineer-procure-construct (EPC) projects. In: *International Journal of Project Management*. 20(4), pp.253-262, 2002.