



BINDING SERVICES

Tel +44 (0)29 2087 4949

Fax +44 (0)29 20371921

e-mail bindery@cardiff.ac.uk

**HYPERMEDIA-BASED
PERFORMANCE SUPPORT SYSTEMS
FOR THE WEB**

A thesis submitted to
the University of Wales
for the degree of
Doctor of Philosophy

by

Ammar M. Huneiti

BSc., MSc.

Intelligent Information Systems Group
Cardiff School of Engineering
University of Wales – Cardiff

2004

UMI Number: U584665

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 U584665

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

SYNOPSIS

The work reported in this thesis is an attempt to apply integrated knowledge-based and adaptive hypermedia technologies in the area of electronic performance support. Moreover, this work is a contribution in the direction of “structured” hypermedia authoring of technical documentation. It tackles the main challenges associated with the systematic development of Web-based technical documentation which include the design, authoring, and implementation, and the creation of supporting CASE tools. The main contribution of this research is a systematic methodology for the development of hypermedia-based Performance Support Systems (PSSs) for the Web which adheres to the main characteristics of advanced PSSs. These characteristics are outlined in a conceptual model that complies with state-of-the-art technologies and current practices in the field of user performance support.

First, the thesis suggests a conceptual model for advanced PSSs. These are characterised as mainly consisting of two loosely coupled components that are designed and accessed in a task-based and user-centred manner. The first component is a freely browsed technical documentation of the application domain. The second component is the expert advisor that provides assistance for more specific, complex, and difficult to learn tasks. The integrated technologies utilised in advanced PSSs include Web-based hypermedia and knowledge-based systems.

Second, the thesis concentrates on the first component of advanced PSSs i.e. technical documentation. It suggests a usage-based data model for the design of technical documentation. The proposed model abstracts the intended purpose of the documentation, the tasks supported by the documentation, and the functional characteristics of documents. These abstractions are integrated in a usage-based semantic network where rules and valid relationships are identified. This design framework can then be used by authors in order to organise, generate, and maintain

the technical documentation i.e. authoring. In addition, this model is also used to support a strategy for the adaptive retrieval of hypermedia documents.

Third, the thesis suggests a model-driven hypermedia authoring approach for Web-based technical documentation. This approach utilises the usage-based data model for the design of technical documentation (described above). In addition, it complies with the principled guidelines of structured authoring.

Finally, the thesis focuses on “intelligent” PSSs. It promotes the provision of intelligent performance support through the utilisation and integration of technologies used in developing knowledge-based diagnostic Expert Systems (ES) and adaptive hypermedia systems. This integration is implemented through the use of hypermedia which allows supporting content to be synchronized with the diagnostic ES inference process. The integrated adaptive diagnostic ES supports the user by providing what-to-do and how-to-do type of information tailored (adapted) to the user’s knowledge of the subject domain. The special organisation of displays in an HTML-based user interface allows users, while employing the ES for fault diagnosis, to request detailed information about a certain diagnosis procedure, and then return to the ES to continue from where they left off.

The solutions proposed in this thesis are demonstrated through the development of a prototype PSS for an all-terrain fork-lift truck. The performance support is provided through (i) a technical manual, (ii) a diagnostic ES for locating and correcting braking system faults, and (iii) an adaptive information retrieval utility.

ACKNOWLEDGEMENTS

IN THE NAME OF ALLAH, MOST GRACIOUS, MOST MERCIFUL.

“and say: my lord, increase me in knowledge” [Holy Quran 20:114].

First and last all praise is due to Allah for all his uncountable blessings and favours.

I will always be grateful to my parents whom without their help and encouragement, I could not have achieved this. Not to forget my wife who has put up with my absence throughout my research. Thanks to my wife and my son “Hareth”.

Equally, my thanks and gratitude goes to my supervisor Professor D. T. Pham for his unique, kind, caring, and sincere supervision, and for all his personal support, encouragement, and guidance.

Last but not least, the cooperation of the members of the Intelligent Systems Laboratory as a whole is highly appreciated. In particular, I would like to thank the I2S Group members for their good companionship.

TABLE OF CONTENTS

SYNOPSIS	<i>i</i>
ACKNOWLEDGEMENTS	<i>iii</i>
DECLARATION AND STATEMENTS	<i>iv</i>
LIST OF FIGURES	<i>xi</i>
LIST OF TABLES	<i>xiv</i>
ABBREVIATIONS	<i>xvi</i>
NOMENCLATURE	<i>xix</i>

1 INTRODUCTION	<i>1</i>
-----------------------	-----------------

1.1 MOTIVATION	1
1.2 OBJECTIVES	4
1.3 OUTLINE OF THE THESIS	6

2 SYSTEMATIC DEVELOPMENT OF ADVANCED PERFORMANCE	<i>9</i>
---	-----------------

SUPPORT SYSTEMS	
------------------------	--

2.1 PERFORMANCE SUPPORT SYSTEMS	10
2.1.1 Definition and Objectives	10
2.1.2 Static Methods of Performance Support	11
2.1.3 Electronic Performance Support Systems	12
2.2 ADVANCED PERFORMANCE SUPPORT SYSTEMS	15
2.2.1 Conceptual Model for Advanced PSSs	15
2.2.2 Discussion	19

2.3	INTEGRATED TECHNOLOGIES FOR ADVANCED PERFORMANCE SUPPORT	20
2.3.1	Web-based Hypermedia	21
2.3.2	Knowledge-Based Systems	24
2.3.3	Adaptive Hypermedia	26
2.3.4	Discussion	30
2.4	A SOFTWARE ENGINEERING PERSPECTIVE ON THE DESIGN AND AUTHORIZING OF DATA-INTENSIVE WEB-BASED HYPERMEDIA APPLICATIONS	31
2.4.1	Authoring: Techniques and Methods	31
2.4.1.1	Traditional Authoring	31
2.4.1.2	Structured Authoring	33
2.4.1.2.1	Structured Authoring Methods	33
2.4.1.2.2	Structured Authoring Techniques	35
2.4.2	Systematic Development of Data-Intensive Web-Based Hypermedia Applications	36
2.4.2.1	Static Hypermedia Authoring	36
2.4.2.2	Problems Associated with Development of Data-Intensive Web-Based Applications	37
2.4.2.3	Why a Systematic Development Approach?	38
2.4.2.4	Software Engineering in Support of a Systematic Development Approach	40
2.4.2.4.1	The Software Engineering Process	40
2.4.2.4.2	Software Process for Web-Based Information Systems	41
2.4.3	Structured Hypermedia Design	45
2.4.3.1	Reference Models for Hypermedia Systems	47
2.4.3.2	A Review of Structured Design Methods for Hypermedia Applications	48
2.5	SUMMARY	51

3 *USAGE-BASED DATA MODEL FOR THE DESIGN OF TECHNICAL DOCUMENTATION*

3.1 MODELLING KNOWLEDGE WITHIN TECHNICAL DOCUMENTATION	53
3.1.1 Usage-Based Analysis of Technical Information	54
3.1.1.1 Purpose of Technical Documentation	55
3.1.1.2 Tasks Supported by Technical Documentation	56
3.1.1.3 Functional Characteristics of Technical Documents	60
3.1.2 Semantic Data Model for Representing Knowledge in Technical Documentation	61
3.1.2.1 Knowledge Representation Using Semantic Networks	61
3.1.2.2 Usage-Based Semantic Network for Technical Documentation	64
3.1.3 Mapping from the Semantic Data Model to a Database Schema	69
3.2 CASE STUDY: USAGE-BASED ANALYSIS AND DATA MODEL FOR TECHNICAL MANUALS	72
3.2.1 Product-Based Technical Manuals	72
3.2.2 Information Analysis of Technical Manuals	73
3.2.3 Abstract Tasks and Activities in Technical Manuals	78
3.2.4 Semantic Database Schema for a Technical Manual	79
3.3 SUMMARY	84

4 *MODEL-DRIVEN HYPERMEDIA AUTHORIZING APPROACH FOR WEB-BASED TECHNICAL DOCUMENTATION*

4.1 HYPERMEDIA AUTHORIZING FOR THE WEB	86
4.1.1 Traditional Process to Web-Based Hypermedia Authoring	86
4.1.2 Model-driven Approach to Web-Based Hypermedia Authoring	89
4.1.3 Application Example	92
4.2 EDITING AND MANAGEMENT OF MULTIMEDIA DATA ELEMENTS	93

4.3	AN APPROACH FOR BUILDING THE HYPERMEDIA-BASED TECHNICAL DOCUMENTATION STRUCTURE	96
4.3.1	Relationships in Hypermedia	96
4.3.2	Attributes for Indexing the IOs of the Technical Documentation	97
4.3.3	Hypermedia Structure for the Technical Manual	101
4.3.3.1	Dynamic Identification Codes for Maintaining the Structural and Semantic Properties of IOs	105
4.3.3.2	Structure Builder - An Authoring Tool	109
4.3.3.3	Automatic Generation of the Hypermedia Pages	111
4.4	NAVIGATION BASED ON INFORMATION SEMANTICS	114
4.4.1	Access Methods	116
4.4.2	Automatic Identification of Semantically-Based Navigational Relationships	120
4.4.2.1	Context-Driven Navigational Relationships	120
4.4.2.2	Purpose-Driven Navigational Relationships	123
4.5	FRAME-BASED PRESENTATION TEMPLATES	127
4.6	SYSTEM ARCHITECTURE	132
4.7	SUMMARY	135

5 INTELLIGENT PERFORMANCE SUPPORT THROUGH INTEGRATED KNOWLEDGE-BASED ADAPTIVE HYPERMEDIA

5.1	INTELLIGENT PERFORMANCE SUPPORT THROUGH KNOWLEDGE-BASED DIAGNOSTIC SYSTEM	137
5.1.1	Encapsulating Diagnosis Knowledge of Experts in an Expert System	137
5.1.2	Integrated Knowledge Engineering Process for Diagnostic ESs	139
5.1.3	Integrated Shallow and Deep Knowledge Model	142
5.1.4	Rule-Based KB for Fault Diagnosis	147
5.1.4.1	Application Domain Example	147
5.1.4.2	Building the Shallow KB	147
5.1.4.3	Abstract Rule Format for the Diagnostic Strategy	151

5.1.4.4 Automatic Generation of the Rule-Based KB in e2gLite ES Shell Format	152
5.1.4.5 Automatic Update of Deep Knowledge Data	159
5.1.5 General Architecture for the Diagnostic Expert System	161
5.2 INTELLIGENT PERFORMANCE SUPPORT THROUGH ADAPTIVE HYPERMEDIA	163
5.2.1 Adaptive Retrieval of Hypermedia-Based Diagnostic Information	163
5.2.1.1 Stereotype Model for User Knowledge	164
5.2.1.2 User Knowledge-Based Strategy for Adaptive Support Using Conditional Semantic Rules	167
5.2.1.3 Adaptive Hypermedia Support for Diagnosis Information	174
5.2.2 General Architecture for the Adaptive Hypermedia System	178
5.3 INTEGRATED ADAPTIVE DIAGNOSTIC EXPERT SYSTEM	180
5.4 SUMMARY	183
 6 CONTRIBUTIONS, CONCLUSIONS AND FUTURE WORK	 185
6.1 CONTRIBUTIONS	185
6.2 CONCLUSIONS	190
6.3 FUTURE RESEARCH	192
 APPENDIX A. INFORMATION OBJECTS METADATA	 195
APPENDIX B. CODING INFORMATION OBJECTS USING STRUCTURE BUILDER	204
APPENDIX C. AUTOMATICALLY EXTRACTED PURPOSE-DRIVEN NAVIGATIONAL RELATIONSHIPS	207
APPENDIX D. FRAME-BASED PRESENTATION TEMPLATES	220
D.1 Index Template	220
D.2 Collection Template	221

D.3 Guided Tour Template	221
--------------------------	-----

APPENDIX E. AUTOMATIC GENERATION OF KNOWLEDGE BASES 223

IN e2gLite ES SHELL FORMAT

E.1 KB Generator – Source Code	224
--------------------------------	-----

E.2 the Complete Rule-Based KB Generated in e2gLite ES Shell Format	231
---	-----

REFERENCES 238

LIST OF FIGURES

Chapter 2

2.1 Conceptual Model for Advanced PSSs	18
2.2 Software Process and Web-Based Development	42

Chapter 3

3.1 Classifications of Technical Documents According to their Purpose	57
3.2 Primitive Tasks and Abstract Task Types	59
3.3 Usage-Based Semantic Network for Technical Documentation	65
3.4 Semantic Rules and Valid Relationships	68
3.5 Abstract E-R Diagram Representing the Semantic Database Schema	71
3.6 Activity-Based Database Schema for Technical Manuals	81
3.7 Schema Editor	83

Chapter 4

4.1 Conceptual Model-The Traditional Web-Based Hypermedia Authoring Process	87
4.2 Model-Driven Approach to Web-Based Hypermedia Authoring	90
4.3 Information Objects	95
4.4 Relationships in Hypermedia	98
4.5 Attributes of Information Objects	99
4.6 Braking System-Logical Structure with Semantic Domains	103
4.7 Identification Codes for Information Objects	108
4.8 Structure Builder - An Authoring Tool	110
4.9 Structural Outline of Automatically Generated HTML Pages	113
4.10 XML-Based Technical Manual Rendered Using CSS and XSL Style Sheets	115
4.11 Access Methods	117

4.12 Semantic Domains - Alternative Views for the Technical Manual's Home Page	119
4.13 Generation of Context-Driven Navigational Hyperlinks	121
4.14 Purpose-Driven Navigational Relationships	124
4.15 Keyword-Based Referential Hyperlinks	128
4.16 Physical Organisation of Presentation Templates of Access Methods	130
4.17 System Architecture	133

Chapter 5

5.1 General Architecture of a Typical Knowledge Engineering Process	140
5.2 Integrated Knowledge Engineering Process for Developing Diagnostic Expert Systems	143
5.3 Integrated Shallow and Deep Knowledge Model for Diagnostic Experts Systems	144
5.4 E-R Diagram Representing the Database Schema of the Integrated Knowledge Model	146
5.5 Structured Shallow Knowledge Data for Diagnosing Braking System Faults in the Forklift Truck	149
5.6 Automatically Generated e2gLite Rules and Prompts for Fault No.1 (Damaged Brake Fluid Pipes)	155
5.7 Decision Flow Chart for Determining Fault No.1 (Damaged Brake Fluid Pipes)	157
5.8 User Interface and Example Scenario of the Diagnostic ES	158
5.9 Identification of Faults Diagnosis and Correction Procedures	160
5.10 General Architecture of Diagnostic Expert System	162
5.11 Adaptive Hypermedia Systems - Abstract Overview	166
5.12 Main Principle of the Adaptive Strategy	168
5.13 User Knowledge-Based Conditions for Identifying "Relevant" Information Objects	173
5.14 Alternative Views of "Check the Servo Brake" Diagnosis Procedure for Different User Knowledge Stereotypes	176
5.15 General Architecture for the Adaptive Hypermedia System	179
5.16 User Interface for the Integrated Adaptive Diagnostic Expert System	181

5.17 General Architecture of the Integrated On-Line Adaptive Expert Diagnostic System	182
---	-----

Appendix B

B.1 Textual Procedure “Wheel Dismount”	205
B.2 Clarification Image “Unscrew the nuts”	205
B.3 Clarification Animation “Unscrew the nuts”	206
B.4 Clarification Image “Take off the spring washers”	206

LIST OF TABLES

Chapter 2

2.1 Supporting Complex Systems (Paper-Based vs. Electronic)	14
2.2 Comparison - Structured Hypermedia Design Methodologies	49

Chapter 3

3.1 Functional Characteristics of Documents and their Abstract Types	62
3.2 Classification of Information Categories of Technical Manuals According to their Purpose	75
3.3 Abstract Purpose of Product Information	76
3.4 Functional Characteristics of Product Information	77
3.5 Information Categories and Primitive Tasks in Support of the Abstract Activities of Technical Manuals	80

Chapter 4

4.1 Information Objects Metadata-Attributes and Values	106
4.2 Automatically Generated “Supported_By” Relationships	126
4.3 Visual Presentation Icons for Functions of Information Objects	131

Chapter 5

5.1 Relationship between Adaptive Support and User Knowledge	170
--	-----

Appendix A

A.1 Information Objects and Metadata	196
--------------------------------------	-----

Appendix C

C.1 Automatically-Generated Purpose-Driven “Action_Applied”	208
Navigational Relationships	
C.2 Automatically-Generated Purpose-Driven “Plan_Info”	209
Navigational Relationships	
C.3 Automatically-Generated Purpose-Driven “Supported_By”	215
Navigational Relationships	

ABBREVIATIONS

ADAPTS	Adaptive Diagnostics and Personalised Technical Support
AH	Adaptive Hypermedia
AHA	Adaptive Hypermedia Architecture
AHAM	Adaptive Hypermedia Architecture Model
AI	Artificial Intelligence
ANN	Artificial Neural Network
CAD	Computer Aided Design
CASE	Computer Aided Software Engineering
CBT	Computer Based Training
CGI	Common Gateway Interface
ChemicalML	Chemical Markup Language
CSS	Cascading Style Sheet
DBMS	Database Management System
DBPL	Database Programming Language
DTD	Document Type Definition
El-Tech	Electronic Technician
EORM	Enhanced Object-Relationship Model
E-R	Entity Relationship
ES	Expert System
FLC	Fuzzy Logic Controller
GUI	Graphical User Interface
HB1	HyberBase 1
HBMS	HyperBase Management System

HCI	Human Computer Interaction
HDM	Hypermedia Design Model
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IE	Inference Engine
IETM	Interactive Electronic Technical Manual
IHDM	Index-Driven Hypermedia Design Methodology
IO	Information Object
IPM	Intelligent Product Manual
IR	Information Retrieval
JDBC-ODBC	Java Database Connectivity- Open Database Connectivity
JDK	Java Development Kit
JSP	Java Server Pages
JSWDK	Java Server Web Development Kit
KB	Knowledge Base
KE	Knowledge Engineering
KM	Knowledge Management
MathML	Mathematics Markup Language
MMA	Mars Medical Assistant
MUCH	Many Using and Creating Hypermedia
MusicML	Music Markup Language
O-O	Object-Oriented
OOHDM	Object-Oriented Hypermedia Design Method
PC	Personal Computer
PDA	Personal Data Assistant

PDM	Product Data Management
PSS	Performance Support System
RICH	Reusable Intelligent Collaborative Hypermedia
RMM	Relationship Management Methodology
SE	Software Engineering
SGML	Structured Generalised Markup Language
SOHDM	Scenario-Based Object Oriented Design Methodology
SQL	Structured Query Language
UI	User Interface
UML	Unified Modelling Language
URL	Universal Resource Locator
WAP	Wireless Application Protocol
WebML	Web Markup Language
WHDM	Workflow-Based Hypermedia Development Methodology
WML	Wireless Markup Language
WWW	World Wide Web
XML	eXtensible Markup Language
XSL	eXtensible Style Language

NOMENCLATURE

Adv	Advice
AHG	Adaptive Hypermedia Generator
ARU	Adaptive Rendering Utility
ASE	Adaptive Support Engine
CF	Certainty Factor
Cla	Clarification
ConForm	Container Form
DI	Disposal Instructions
Fund	Fundamental
HIST	Handling, Installation, Storage, and Transit
ICat	Information Category
ID	Identification Code
MI	Modification Instructions
OI	Operating Instructions
Org	Organisation
PL	Parts List
PP	Purpose and Performance
Proc	Procedure
Spec	Specification
SU	Search Utility
T_ID	Topic Identifier

TD	Technical Description
TM	Technical Maintenance
TMS	Technical Maintenance Schedule

CHAPTER 1

INTRODUCTION

1.1 MOTIVATION

Supporting the performance of workers in modern hi-tech job environments has become an increasingly complex, time consuming and costly task which requires advanced performance support methods. Simple tasks of low information volume can be efficiently supported using traditional performance support methods such as paper documentation, lectures, job aids, and instructor-led courses. However, many problems are associated with these traditional methods especially when used to support complex products or systems. Bezanson [1995] states that “traditional user support methods are no longer effective”, because “traditional training methods are not responsive to individual needs as they emphasize training rather than learning”. He also points out that traditional performance support methods involve cumbersome manuals and labour-intensive updates of workers’ document sets.

Reliable Performance Support Systems (PSS) can enhance productivity, and reduce training costs, time to achieve proficiency, and errors. Meanwhile, they increase quality, accuracy, task completion rate, and worker autonomy [Desmarais et al., 1997; McGraw, 1997]. However, according to Fischer and Horn [1997], without tools that are primarily PSS tools and with no clear methodology for building them or

measuring their performance, PSSs will be limited to being just “*an approach*”.

Technical documentation is a major component of any PSS. Recent technological advances in the field of information processing, storage, presentation, and retrieval, had a huge impact on authoring of technical documentation. However, powerful authoring tools and advanced technologies cannot help improve the quality of the content of documentation unless similar powerful and advanced authoring methods are utilised [Thibeuau, 1996; and Csinger et al., 1995]. Moreover, as the documentation becomes more complex, it exhibits emergent behaviours, and it demands new attitudes, concepts, and work from the technical communicators [Price, 1997]. In addition to technical documentation, expert diagnostic systems are another major performance support component. As current products, equipment, and systems increase in size and complexity, the difficulty of diagnosing their faults increases, and hence the need for utilising expert diagnosis knowledge in supporting user performance is increasingly essential [Patel et al., 1996].

Furthermore, Performance Support is a concept that capitalises on recent advances in many technological areas such as Artificial Intelligence (AI), Human Computer Interaction (HCI), Computer-Based Training (CBT), Knowledge Management (KM), and the Internet technology. Recently developed advanced PSSs are greatly influenced by the integration of more than one technology. The technologies that are employed in developing advanced PSSs should reflect their main characteristics. The Integrated Technology solution to PSSs is best outlined by Raybould [2000] in his 21st century vision of building performance-centred Web-based information and knowledge management systems. He states that “the Human Computer Interaction,

Expert Systems, and Technical Documentation fields have all been moving closer to those approaches advocated by the performance support community. Technical books have become interactive electronic manuals, stand-alone ESs have been embedded in information systems, and instructor-led training courses have become Web-based training modules integrated with hyperlinked background reference information”.

Two major technologies have emerged that significantly influenced the development of PSSs, namely *hypermedia* as an authoring tool and the *Web* as a novel communication medium. On-line hypermedia have dramatically changed the way people use and present information, so much so, that there is a need to have new theories and models for understanding how technology and content are related in this new communication environment. Nevertheless, most researchers argue that there is a weakness in the current methodologies that support the development of Web-based hypermedia applications, especially data-intensive applications [Brusilovsky et al, 2002; Gomez et al, 2001; Rossi et al., 2001; Segor et al., 2000; Fraternali, 1999; and Coda et al, 1998]. In addition, there is no consensus on a general design process model for these applications. Currently, most Web developers manually generate low-level implementations of mark-up language-dependent files or use commercially available tools to produce them. The development of data-intensive Web-based hypermedia applications is usually a collaborative process involving team members with different expertise, knowledge, skill(s), aims, and backgrounds. Due to this team effort, it is vital to acquire a common communication language between team members, which maps their understanding into a uniform model [Ding et al., 2002; Klusch, 2001; and Levy and Weld, 2000].

Furthermore, users of PSSs have different levels of knowledge, expertise, and qualifications, and they also have different goals and objectives. According to Pham and Setchi [2003] “information that is presented to users has to vary in its focus, level of detail, and presentation format. It has to be adapted to the information needs of the users”. Therefore, according to Brusilovsky [1999], electronic PSSs is a new and challenging area for the application of adaptive hypermedia techniques. He also states that “we do not know any other (*than his*) PSS equipped with adaptive hypermedia, but we hope that more systems will appear in the near future”.

1.2 OBJECTIVES

The scope of the research reported in this thesis is the intelligent and reliable support of users’ performance through adaptive Web-based environment that provides for all user needs. The overall objective is to provide advanced users’ performance support by integrating technical information with expert-based advice capabilities, adapted to the user’s knowledge of the performed tasks. The work of this research is an attempt to overcome limitations associated with traditional approaches used in the development of conventional PSSs. Particular attention is applied to the provision of advanced design and authoring techniques for hypermedia-based technical documentation and the provision of intelligent performance support methods. This research project should be relevant to all people involved in the development of Web-based PSSs. These include domain experts, system analysts, authors, content managers, designers, knowledge engineers, hypermedia editors, style architects, Web administrators, and others.

The individual research objectives of this project are:

1. To produce a conceptual model for advanced PSSs.
2. To develop a data model for the design of technical documentation.
3. To develop a structured approach for hypermedia authoring of Web-based technical documentation, and an architecture for implementing this approach.
4. To create a structured method and architecture for providing intelligent diagnosis support through a knowledge-based Expert System (ES).
5. To create a method and architecture for the adaptive delivery of hypermedia-based technical documentation.
6. To develop an architecture for integrating the diagnostic ES and the adaptive hypermedia documentation system.

1.3 OUTLINE OF THE THESIS

The main body of this thesis comprises Chapters 2 to 5. Chapter 2 is mainly a review chapter that provides the required background knowledge for the work reported in the rest of this thesis. Chapter 2 also introduces a conceptual model for advanced PSSs derived from work reported in the literature. Chapters 3, 4, and 5 address objectives 2-6 listed above. The final chapter, Chapter 6, summarises the contributions and conclusions of the work reported in this thesis and makes suggestions for future research.

Chapter 2 addresses the first research objective. This chapter comprises two main parts. The first part gives a review of PSSs and their enabling technologies. A conceptual model for advanced PSSs sums up the results of a survey of the state of practice in recently developed PSSs. The chapter also discusses the existing state-of-the-art technologies that can be utilised to deliver this advanced performance support concept. The second part of this chapter addresses issues related to the design and authoring of data-intensive Web-based hypermedia applications from a Software Engineering (SE) perspective. Structured authoring approaches are presented as an improved substitute for traditional authoring. The SE principles that are used to support a systematic development of data-intensive Web-based hypermedia applications are outlined. Finally, a survey of structured hypermedia design methods is presented.

Chapter 3 addresses the second research objective. It presents a usage-based data model for the design of technical documentation. A semantic data model for designing technical documentation is proposed based on an abstract usage analysis of technical information. A case study is conducted using a product-related technical manual in order to demonstrate the validity of this design approach.

Chapter 4 focuses on the third research objective. It presents a model-driven methodology for Web-based hypermedia authoring which utilises the usage-based semantic data model. This methodology is demonstrated through the construction of a hypermedia-based technical manual for a fork-lift truck. The chapter also introduces a navigational model based on information semantics. Furthermore, a presentation technique using frame-based templates, icons, and colours is introduced. Finally, the system architecture that is used to demonstrate the authoring methodology is presented.

Chapter 5 focuses on the fourth, fifth, and sixth research objectives. It is organised in three main sections. The first section introduces a methodology for providing intelligent diagnosis support through knowledge-based expert systems. At the core of this methodology is an integrated knowledge engineering process for diagnostic ESs, which includes an integrated knowledge model. An expert system for locating and correcting braking system faults in a forklift truck is used to demonstrate this methodology. In the second section, an approach for retrieving diagnosis information using adaptive hypermedia is described. A strategy for providing adaptive support based on a stereotype model of the knowledge of the users is discussed. The third

section presents a general architecture for the integration of both systems in one adaptive hypermedia diagnostic ES.

Finally, Chapter 6 summarises the contributions made and the conclusions reached, and suggests directions for further investigation in this area.

CHAPTER 2

SYSTEMATIC DEVELOPMENT OF ADVANCED PERFORMANCE SUPPORT SYSTEMS

This review chapter comprises two main parts. The first part gives a review of Performance Support Systems (PSS) and their enabling technologies. First, the concept of “performance support” is presented through a fundamental discussion of traditional paper-based and electronic PSSs. Next, a conceptual model for advanced PSSs sums up the results of a survey of the state of practice in recently developed PSSs. Then, the chapter discusses the existing state-of-the-art technologies that can be utilised to deliver this advanced performance support concept. The second part of this chapter addresses issues related to the design and authoring of data-intensive Web-based hypermedia applications from a Software Engineering (SE) perspective. First, structured authoring approaches are presented as an improved substitute for traditional authoring. Methods and techniques associated with structured authoring are discussed. Next, the SE principles that are used to support a systematic development of data-intensive Web-based hypermedia applications are outlined. Development issues related to existing limitations in static hypermedia authoring and the development of data-intensive Web-based applications are also discussed. Finally, a survey is carried out of structured hypermedia design including reference models for hypermedia systems and structured design methods for hypermedia applications.

2.1 PERFORMANCE SUPPORT SYSTEMS

2.1.1 Definition and Objectives

Although there are many definitions for Performance Support Systems (PSS), there is a consensus among researchers that their main objective is to enhance the performance of users by supporting their daily work activities and tasks. According to Bezanson [1995], the process of performance support is “a product or process attribute that aims to enhance user performance, through a user interface and support environment that anticipates user needs and supports them conveniently and effectively”. He also defines PSSs as “systems that provide just-in-time training, information, and help functions on a system or product”. Cantando [1996] similarly defines PSSs as “integrated, readily available set of tools that help individuals do their job and increase their productivity with minimal support”. Desmarais et al. [1997] identify the fundamental objective of PSSs as to provide assistance in learning and in performing some sets of tasks. Sleight [1993] likewise state that “PSSs are used to help with how to do a task”, and she adds that “PSSs are also used for finding information, and presenting it in alternate forms of presentation”. Through the perspective of “just-in-time support”, Bezanson [1995] indicates that PSSs should allow workers to control their own learning and to give them the ability to retrieve information at the workplace at the moment they need it. Furthermore, Raybould [1995] outlines the broadest objective of PSSs by identifying a PSS as an infrastructure that captures, stores, and distributes individual and corporate knowledge assets throughout an organisation.

It is clear that the crucial cost justification factor will always influence the decision to develop such systems, which can be costly and time consuming. In [Desmarais et al., 1997] it is reported that a number of companies have claimed large gains by deploying PSSs to support their operators. The inclusion of a PSS with a complex product or system can enhance productivity, reducing training costs, time to achieve proficiency, and errors while increasing quality, accuracy, task completion rate, and worker autonomy [Desmarais et al., 1997; McGraw, 1997].

2.1.2 Static Methods of Performance Support

The determination of the form of the PSS to develop is influenced by many important factors, which may include the complexity of the supported tasks, volume of information, work environment and the information needs of different user groups. Simple tasks with low information volume can be efficiently supported using traditional (static) performance support methods such as paper documentation, lectures, job aids, instructor-led courses, human experts. Traditional job aids may include pocket reference cards, colour codes for switches, buttons and keys, lists of abbreviations, etc. In addition, video and audio tapes are traditionally used to provide performance support.

However, many problems are associated with this conventional static type of performance support especially when used to support tasks related to complex products or systems. Bezanson [1995] states that “traditional user support methods are no longer effective”, because “traditional training methods are not responsive to individual needs as they emphasize training rather than learning”. He also points out

that traditional performance support methods involve cumbersome manuals and labour-intensive updates of workers' document sets. Furthermore, Pham et al., [1999] and Ventura [2000] identify some of the problems associated with "conventional" performance support methods in the technical documentation field. These problems are associated with the portability, complexity, accuracy, reliability, and maintainability of information [Ventura, 2000]. There are also other problems associated with collecting, integrating, and retrieving the information, with the static structure of the presented material, with the restricted support provided to users (reference systems vs. active support systems) and with the limitation in presentation methods [Pham et al., 1999].

2.1.3 Electronic Performance Support Systems

Electronic PSSs¹ are computer-based information systems that have benefited from the digital revolution, and inherited many advantages associated with it. These systems overcome most of the problems associated with static methods of performance support. According to Cantando [1996], the purpose of an electronic PSS is to replace or supplement human experts, paper-based documentation, and costly training programs. Sleight [1993] outlines five "key" characteristics for electronic PSSs namely that are computer-based, provide access to discrete and specific task-related information during the task performance, are used on the job, are controlled by the user, and reduce the need for prior training in order to accomplish a task. According to Sleight the last four characteristics are the ones which distinguish PSSs from other computerised tools.

¹ Some researchers use the term Electronic Performance Support System (EPSS). Throughout this thesis, the term "PSS" will be used, to also refer to EPSS unless it is stated otherwise.

Electronic PSSs can range in complexity from a single help file for a specific task to a complete Expert System (ES) for complex problem solving. They may include an information database, expert and advisory system, help system, application and productivity software, modular learning experiences, assessment feedback, and monitoring systems [Cantando, 1996], or any combination of these. In general, all forms of electronically-based run-time assistance, which can be used to support the performance of users can be classified as electronic PSSs. These include (computer-based) interactive task advising, tests for understanding, wizards, tutors, help files, coaching, training, intervention, and feedback. A specialised and product-oriented type of performance support exists in the form of a product support. According to Pham et al. [2000], product support consists of everything necessary to allow the continued use of a product, including user training, technical manuals, help lines, servicing, spare part ordering, and maintenance management. A comparison between traditional paper-based and electronic performance support methods for supporting complex systems or products is presented in Table 2.1. These methods are compared with regard to volume and weight, accuracy, complexity, portability, access to information and reliability.

According to their deployment method, electronic PSSs can be classified as “standalone” or “embedded”. A stand-alone system is designed with no reference to an existing system and has built-in performance support, e.g. the El-Tech performance support and training system for electronic technicians [Coffey et al., 2003].

Table 2.1 Supporting Complex Systems (Paper-based vs. Electronic)

	Paper-based Performance Support	Electronic Performance Support
<i>Volume and Weight</i>	Increasingly exponential to product complexity.	Very low (e.g.: one CD-ROM)
<i>Information Accuracy, Completeness and Credibility</i>	Difficult and very expensive to update regularly. Thus it becomes inaccurate, incomplete, and out-dated after a certain time.	Comparatively easy and relatively cheap to update. This improves accuracy, completeness and credibility.
<i>Information Retrieval Complexity</i>	Increasingly exponential to the complexity of the supported system. Information becomes disorienting and difficult to retrieve.	Usually provided with good user interfaces and equipped with advanced search facilities.
<i>Portability</i>	High volume and weight make them hard to transport, hard to store, and more difficult for users to carry around.	No need to transport or store. Can be mounted on/with the supported product/system.
<i>Access to Information and Availability</i>	Poor transportability, storage, and limited number of copies prevent the required support from being available at certain locations.	Fast and easy access to information. One copy can be accessed by many at the same time.
<i>Reliability</i>	Inaccurate or incomplete support is unreliable support. Information can become so complex and disorienting which make it unusable.	Cheap and easy to update, providing up-to-date and therefore reliable information.

An “embedded” system, conversely, supplements an existing system, e.g. the wizards within Microsoft™ products such as Word™ and Excel™. Sleight [1993] adopts a finer deployment-based classification of PSSs, which differentiates between a front end to an existing system, a supplement to an existing system, a stand-alone tool for specific tasks, and a new system with integrated performance support. Raybould [2000] classifies performance support as “intrinsic”, i.e. support embedded in the tool or software interface, “extrinsic”, i.e. support that is linked to the tool such as wizards, cue cards, advisors, help, etc., or “external”, i.e. support that is separate from the tool, such as tutorials, computer-based training, telephone hotlines, etc. In addition, the presentation format used in these systems can range from simple text to full multimedia support which includes images, video, animation, audio, virtual reality, etc.

2.2 ADVANCED PERFORMANCE SUPPORT SYSTEMS

2.2.1 Conceptual Model for Advanced PSSs

Supporting the performance of workers in modern hi-tech job environments has become an increasingly complex, time consuming and costly task, which requires advanced performance support methods. Many advanced characteristics of PSSs were highlighted either implicitly or explicitly in previous research in the field of performance support. Analysis of these early research endeavours has highlighted a common direction for advanced performance support, namely to provide task-specific and user-centred support in conjunction with information and problem solving capabilities. For instance, McGraw [1997] identifies the primary components of a

“true” PSS. These include task-based and user-aware interfaces, help systems, coaches or advisors, wizards, and tutors. She states that “a system of performance support provides multiple, interlinked types of computer guidance and information, integrated with the normal working environment”. Cantando [1996] similarly classified the main features of a typical PSS as task-specific skills training, task-specific information access, task-specific templates and forms, and expert advice needed to solve job-performance problems. According to Bezanson [1995], performance support means creating environments that integrate *business information* with task structuring support and tools. Performance support also means enabling rapid and consistent *expert performance* by all employees. Sleight [1993] on the other hand, identifies the integration of information, advice, and learning experience as one of the main characteristics of PSSs.

The above early performance support directions are clearly evident in many recent advanced PSSs. These later PSSs consist mainly of two integrated abstract components that are designed and accessed in a task-based and user-centred manner. The first component is a freely browsed technical documentation of the application domain i.e. “how-to-do” type of information. The second component is the expert advisor that provides assistance for more specific, complex, and difficult to learn tasks i.e. “what-to-do” type of information.

These features are, to different extents, evident in all the more recent advanced solutions that reviewed. These solutions include, a performance support and training system for electronic technicians El-Tech [Coffey et al., 2003], the Adaptive Diagnostic and Personalised Technical Support system ADAPTS [Brusilovsky et al.,

2002; Brusilovsky, et al., 1999; and Cooper et al., 1999], the adaptive Intelligent Product Manual IPM [Pham and Setchi, 2003; Pham et al., 2000; Setchi, 2000; and Pham et al., 1999], and the Mars Medical Assistant MMA [Francisco-Revilla et al., 2000]. These advanced practical solutions are designed and implemented in order to support users within their job environments through the two main components mentioned earlier. For instance, the technical documentation support in ADAPTS is provided through the Interactive Electronic Technical Manual IETM, and in IPM through ProManual. Moreover, the expert advice is provided in ADAPTS through the diagnostic system, and in MMA through the adaptive diagnostic aid. These performance support components are loosely coupled, thus the inclusion of a new component or the exclusion of an existing component should not affect the operation and performance of other components. The availability of these components is intended to provide the most appropriate type of performance support on the basis of the performer's needs, the task complexity, the consequence of poor performance, error rates, and other factors [McGraw, 1997].

Figure 2.1 summarises the ideas mentioned above through a conceptual model for advanced PSSs which resembles the current state of the practice. With respect to a certain job-related task, the user either directly requests information, or the system “smartly” detects his needs, or a combination of both. User-related features and system-supported tasks are modelled, to different extents, in the system. These models are consulted by the system's main components either after every information request or automatically during task performance. User and task modelling are reviewed later in this chapter. Job-related tasks of the abstract type “how-to-do” are processed through the system's technical documentation component. Depending how advanced

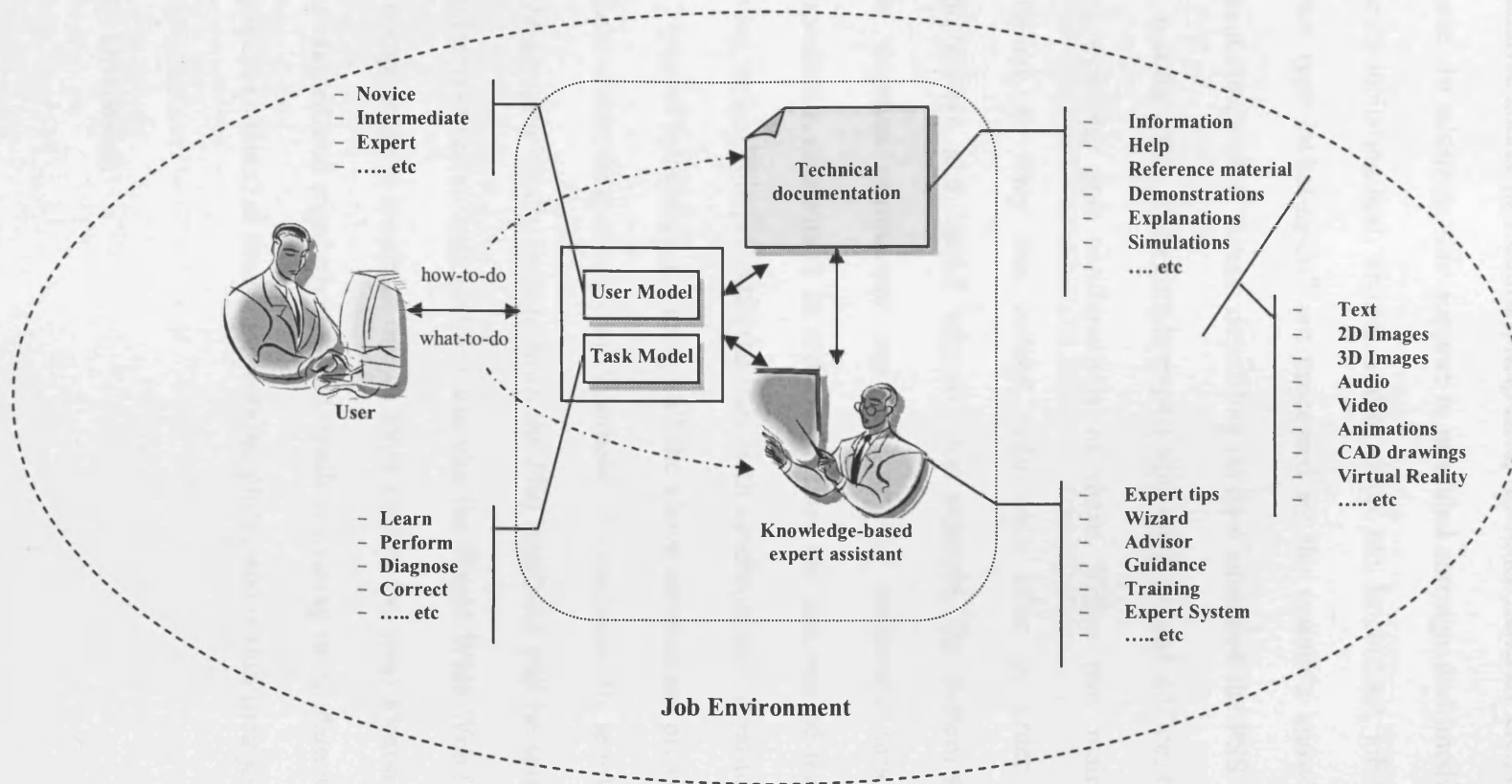


Figure 2.1 Conceptual Model for Advanced PSSs

the PSS is, this type of support can be provided in the form of structured information (declarative and/or procedural), help files, reference material, etc., or any combination of these. In addition, this support is provided through multimedia-rich information elements including text, images, audio, video, etc. In contrast, job-related tasks of the abstract type “what-to-do” are processed by the system’s knowledge-based expert assistant component. Again, depending on how advanced the PSS is, this support can vary in complexity from simple expert tips, wizards, and advice, to fully functioning expert systems, and combinations of these. These two main components are interlinked, so they can consult with each other in order to provide more comprehensive and useful support. For example, the system’s knowledge-based expert assistant component can pass on the performed task to the technical documentation component in order to retrieve the task-related technical information and vice versa. This provides the user with synchronised “what-to-do” and “how-to-do-it” type of information. Finally, all the above services are provided just-in-time to the users within their own job environments. Furthermore, these job environments can now be geographically remote from the PSS itself. As will be shown later, advances in the Internet technology and in particular the World Wide Web (WWW) enable the PSS components to reside at one or more central (servers) locations, where they can be maintained and regularly updated, while providing up-to-date services instantly to many users (clients) at their own working places and within their job environments.

2.2.2 Discussion

This section has highlighted the need for advanced performance support methods to deal with the increasing complexity of modern products and systems. The review of

early and more recent research endeavours in the field of performance support, has demonstrated that the level of user performance support can be substantially enhanced by integrating factual information and explanatory capabilities within knowledge-based expert assistant, adapted to the user's knowledge of the performed task. These fundamental characteristics of advanced PSSs have been blended together in a conceptual model for "advanced" PSSs. This conceptual model has been verified against a number of state-of-the-art PSSs, which have been shown to fall, more or less, within its boundaries.

2.3 INTEGRATED TECHNOLOGIES FOR ADVANCED PERFORMANCE SUPPORT

Performance support is a concept that capitalises on recent advances in many technological areas such as Artificial Intelligence (AI), Human Computer Interaction (HCI), Computer-Based Training (CBT), Knowledge Management (KM), and Internet technology. The technologies that are employed for developing advanced PSSs should reflect their advanced characteristics. These include two main technologies namely *hypermedia* and *knowledge-based systems*. The former exploits techniques associated with multimedia presentation, online documentation and Web technology. The latter exploits techniques associated with expert systems and knowledge modelling. Integrating hypermedia and knowledge-based technologies provide an important hybrid technology, namely adaptive hypermedia (AH). AH is utilised in order to enhance the knowledge delivery process. Moreover, the conceptual model shown in Figure 2.1 demonstrates that advanced PSSs result from the integration of more than one technology, with no one technology on its own able comprehensively to provide

the required advanced characteristics. This integrated technological solution is adopted, to a certain extent, in the above mentioned research endeavours, such as those by Coffey et al. [2003], Pham and Setchi [2003], Brusilovsky et al. [2002] and [1999], Francisco-Revilla et al. [2000], Pham et al. [2000] and [1999], and Cooper et al. [1999].

2.3.1 Web-Based Hypermedia

Hypermedia is an interconnected collection of multimedia-based documents. Within the context of performance support, hypermedia helps in the process of conveying information to users through hyperlinks which associate information items together, enabling users to browse the related information on demand (user-controlled operation). In addition, the activation of a hyperlink is triggered by the user's desire to accomplish a task (task-specific and user-controlled operation). Moreover, information quality and quantity can be controlled by managing the provision of the multimedia elements and/or associated hyperlinks. This can be achieved through personalisation, for example to provide general information to novice users and, in contrast, more detailed information to experienced users (user-centred operation). Furthermore, when the multimedia items are blended together in the "right way" they can dramatically improve the learning process. Thus, hypermedia is a technology that has a revolutionary effect, greatly influencing the development of technical documentation [Thomson et al., 2001; Pham et al., 1999; Lee et al., 1997; Reiter et al., 1995 and Coleman, 1991], CBT systems [Bodendorf et al., 1997 and Langer et al., 1994], and multimedia presentation systems [Saarela, 1997; and Chung et al., 1996]. Hypermedia is also the natural format for publishing on the Web [Rossi et al., 2001;

Raybould, 2000; Ceri et al., 2000, Fraternali, 2000 and 1999; and Thibeau, 1996], providing an efficient medium for knowledge and information dissemination and exchange. According to Setchi [2000], the adoption of hypermedia in training and performance support results in increased retention of information, reduced learning time, increased understanding and consistency, and enhanced user performance.

Traditional performance support tools are useful for small organisations or product vendors that are geographically close to their customers. Organisations with distributed global operations require global PSSs, providing reliable support to all, regardless of their location, platform, or limited resources. The ultimate goal of any organisation is to deliver up-to-date information to the right people, in the right way, at the right place, and at the right time. Since the creation of the WWW, this aim has become more realistically achievable. Geographical distances have become shorter, knowledge and expertise can be shared more efficiently, and the reliable information necessary for people to perform their jobs has become easily available [Staab et al., 2000; Schwabe et al., 1998; Takahashi et al., 1997]. According to Raybould [2000] “Web-based systems (both Internet and Intranets) are becoming a major focus of software engineers and human performance technologists”. In addition, this pioneering technology has reduced the remoteness within the business chain in terms of vendors, suppliers, distributors, technical supporters, clients, etc. It has also provided a platform-independent environment, together with increased reliability, credibility, and content control. Furthermore, Thibeau [1996] outlines the benefits that the WWW can bring about in organisational performance. These include improvement of the distribution and updating processes, centralised control over information, and the reuse of information fragments. In contrast with first generation

hypermedia applications, usually delivered in CD-ROMS, kiosks, etc., Rossi et al. [2001] argue that “Web-based hypermedia applications are the second generation of hypermedia applications” and that “good Web applications should be, first of all, good hypermedia applications”. According to Rossi et al. [2001], the former applications were not supposed to be updated and, in general, were not critical for any organisation. However, the latter applications are constantly updated, and permanently enriched with new services and navigation and interface features, according to the organisation’s marketing policy.

Web-based hypermedia applications are also distinguished by their mobility, i.e. they can be interrogated not only using Personal Computers (PCs) or laptops but also using any Web-enabled device, such as Personal Data Assistants (PDAs), digital televisions and WAP-enabled mobile phones [Gomez et al., 2001]. This dynamic communication feature is useful in supporting the user performance because it preserves the mobility of the operators within their work place. In addition, Web technology is based on the client-server architecture [Fraternali, 1999], where all data are held in one or more central repositories (servers) and distributed over a “network” to all users (clients). This architecture provides centralised control that enables up-to-date information to be published as soon as it exists. It also enables security and privilege control of who can see what documents and when. Other features include platform independence and minimal client-side installation with only a standard Web browser required.

2.3.2 Knowledge-Based Systems

On-the-job performance is supported through knowledge-based systems that encapsulate expert knowledge in a specific domain and make it available to users in the form of assistance, guidance, training, advice, coaching, etc. A knowledge-based job performance support tool helps a less experienced worker to perform at the level of more skilled users by encapsulating this expertise.

Knowledge-based systems intelligently support the performance of users by providing task-specific, user-tailored, and expert knowledge-based information. Different types of knowledge are processed and manipulated by knowledge-based PSSs, including domain knowledge, user-related knowledge, task-related knowledge, and expert-related knowledge. In order for this knowledge to be utilised for performance support it needs to be captured (*knowledge acquisition*) and then represented (*knowledge representation*) within a knowledge engineering process (KE); this is the process of building intelligent systems [Negnevitsky, 2002].

There is a consensus among researchers that knowledge acquisition is the most difficult stage in the KE process [Negnevitsky, 2002; Raybould, 2000; Rolston, 1988]. Some of the knowledge acquisition techniques are reported by Coffey [2003], including structured interviews, unstructured interviews, and “contrived” techniques such as decision analysis, and rating/sorting of tasks. The knowledge acquisition technique adopted in the Performance Support Mapping® methodology [Raybould, 2000], involves talking directly to job performers and subject matter experts about the work and identifying goals and barriers to performance. Raybould states that “the

design process is therefore data-driven according to the work and the performers rather than suppositions by the design team”. He specifies the rule of three “actuals”, which includes observing the actual work (not simulated work), observing the actual job performers (not ex-job performers), and observing the actual work places (not an interview room). Moreover, Rolston [1988] reports other “conventional” knowledge acquisition techniques which include conducting surveys, focus groups, and reviewing case studies.

General-purpose knowledge representation techniques lie on a continuum from informal (easily understood by humans) to formal (capable of being evaluated by machines) [Coffey, 2003; Rolston, 1988]. Knowledge representation techniques may include formal logic, production rules, Object-Oriented (O-O) models, Unified Modelling Language (UML) notations, Entity-Relationship (E-R) principles, ontology, semantic networks, and graphs/maps.

The type of the knowledge to be represented determines the type of the knowledge representation technique to use. For example, in rule-based expert systems, production rules (IF-THEN) are used for representing the expert-related knowledge in a knowledge base (KB) [Negnevitsky, 2002, Patel et al, 1996]. Logical domain knowledge in hypermedia systems may be represented using O-O/UML modelling [Gomez et al., 2001; Rossi et al., 2001; Segor et al., 2000; Schwabe et al., 1998; and Saarela et al., 1997], E-R diagrams [Ceri et. al., 2000; Takahashi et al., 1997; and Isakowitz et. al., 1995], ontology [Ding et al., 2002; Staab et al., 2000; Studer R. and Sure Y. 2000], graph theory formal logic [Wang et al., 1998], and semantic networks [Bonfigli et al., 2000; Langer et al., 1994; and Schnase et al., 1993].

Task- and user-related knowledge is used to suggest what and how performance support information should be presented. Task-based knowledge is represented using semantically-rich task hierarchies (special types of semantic networks) [Brusilovsky et al., 2002; Francisco-Revilla et al., 2000; Garlatti et al. 1999; and Vassileva, 1996]. According to Garlatti et al. [1999], task modelling in Web-based adaptive hypermedia systems can help in determining a view of hyperspace, communicating with the users to get certain parameters, determining the current goal, and defining the adaptation method and its parameters. User-related knowledge (user model) is represented using either overlay models, which are based on the structural model of the subject domain [De Bra et al., 1998], stereotypes models [Pham and Setchi, 2003], or a combination of both [Bonfigli et al., 2000]. Normally, in order to put task- and user-related knowledge into a system these knowledge types need to coexist and be interlinked with the domain model. For example, in ADAPTS [Brusilovsky, et al., 2002], the domain, task, and user models are fully integrated. This combination is also evident in MMA [Francisco-Revilla et al., 2000], which in addition applies a stereotype model of the current “situation” e.g. emergency, educational, etc.

2.3.3 Adaptive Hypermedia

Adaptive hypermedia systems are defined by Brusilovsky [1996] as “all hypermedia systems which reflect some features of the user in the user model and apply this model to adapt various visible aspects of the system to the user”. Adaptive hypermedia attempts to solve several problems associated with the freedom of exploration provided by hypermedia, including that of disorientation and cognitive

overload. According to Cooper et al. [1999], these problems exist when authors define a path through hyperspace that may not correspond to what a user needs or wants to follow. In addition, users are frequently overwhelmed by the quantity of information they must deal with and, as a result, become lost either in a navigational sense or in terms of losing sight of their original objectives. Adaptive hypermedia tackles these problems by enhancing the quality of the delivered information, i.e. delivering information that is concise, specific, relevant, and easy to understand. This feature is an essential requirement for providing advanced performance support. Combining adaptive behaviour with advanced hypermedia functionalities can significantly enhance the type of support provided to performers.

This review of adaptive hypermedia systems is inspired by the survey conducted by Brusilovsky [1996], which is one of the most comprehensive reviews in the field. According to Brusilovsky, these systems can be useful in any application area where the system is expected to be used by people with different goals and knowledge and where the domain space is reasonably large. This includes educational hypermedia, on-line information systems, help systems, information retrieval hypermedia systems, institutional information systems, and systems for managing personalised views. In adaptive hypermedia systems, the content, navigation, and presentation of information can be tailored (personalised) to the user's needs by means of a user model [Francisco-Revilla et al., 2000; Cooper et al., 1999; De Bra, 1999; and Brusilovsky, 1996]. Brusilovsky [1996] outlines in detail the methods and techniques used in the content and navigation (link) levels of adaptation. Content adaptation methods are identified as additional explanation, prerequisite and comparative explanation, explanation variant, and sorting content. Content adaptation techniques include

conditional text, stretch text, fragment/page variant, and a frame-based technique. Navigation adaptation methods are identified as global guidance, local guidance, local orientation support, global orientation support, and managing personalised views. Navigation adaptation techniques used include matrix of relevance, personal relevance network, rule-based technique, tasks hierarchy, conceptual network, case-based and a neural network technology.

The representation of the user's state of mind is called a user model [De Bra, 1999]. The techniques for acquiring user models fall along a continuum from "explicit" to "implicit" in "adaptable" and "adaptive" systems, respectively [Csinger et al., 1995]. In [De Bra, 1999], hypermedia systems are classified as *adaptable* or *adaptive*, according to their personalisation techniques. In the former systems, the user provides (explicitly) some profile (through a dialogue) and the system provides a version of the hypermedia application that corresponds to the selected profile. In contrast, the latter systems monitor (implicitly) the user's behaviour (browsing action) and adapt the presentation accordingly. In his answer to the question "adapting to what?", Brusilovsky identified five features related to the current context of the user's work and to the user as an individual which can be taken into consideration by an adaptive system. These are the users' goals, knowledge, background, hyperspace experience, and preferences. The first two are the most important features that are used in most of the adaptive hypermedia systems referenced in this review. These features can be represented using overlay or stereotype user models or a combination of both. In addition to these two representation methods, Bonfigli et al. [2000] reports a third type of user model namely, the "buggy" model which represents the user's knowledge based on the deviations from an expert knowledge, i.e. user misconception.

Furthermore, Francisco-Revilla et al. [2000] argue that there is a drawback in relying, solely, on a single source of information, namely a user model. They state that “the system may require taking into account considerations that are not properly related to the person using the system, since the same user may present different requirements in different cases”. They therefore integrated user, task, and situation (e.g. emergency, educational, etc.) models and used them to create adaptive hypermedia structures by piecing together different information fragments.

Some adaptive applications have become available that use Web technology. De Bra [1999] highlights an additional issue specific to adaptive Web-site design, which is the communication between different adaptive Web-site engines. He states that “when adaptive Web-sites can exchange information about the same user they can adapt to a user more quickly and in a better way. Therefore adaptive Web-sites should be able to exchange (parts of) user models”. He demonstrated this capability in the AHA system [De Bra et al., 1998]. Furthermore, adaptive Web-sites that improve their organisation and presentation by learning from visitor access patterns (logs) are reported in [He et al. 2002; Perkowitz et al., 2000 and 1999; Boley et al., 1999; and Broder et al., 1997]. A common characteristic of these systems is that they apply data mining techniques to user access logs, which record the user behaviour at the site, in order to tune the site to the users’ needs (by clustering documents). Moreover, Bodendorf et al. [1997] integrate AI methods, such as fuzzy logic and Artificial Neural Networks (ANN), with multimedia databases and hypermedia to adapt information to a user’s preferences, motivation, and experiences.

2.3.4 Discussion

The main objective of an advanced PSS is to “efficiently” deliver the required knowledge to task performers. Efficient delivery of knowledge has two main requirements. First, the delivery process must be quick and easy. The user must have access to the required knowledge, on the job, when it is needed, with minimal time and training, and upon request. This requirement is most important when the PSS is employed in situations where the time factor is critical, for instance, when using a help system to solve emergencies. Secondly, the delivered knowledge must be task-specific, relevant, and easily understood by the user. The advanced PSS must minimise information disorientation and cognitive overload by including only task-relevant information and discarding irrelevant information. This eliminates confusion and speeds up the learning process. This section has outlined that efficient knowledge delivery is achievable through an integrated technological solution which includes Web-based hypermedia, knowledge-based systems, and adaptive hypermedia.

The integrated technology solution to PSSs is best outlined by Raybould [2000] in his 21st century vision of building performance-centred Web-based information and knowledge management systems. He states that “the Human Computer Interaction, Expert Systems, and Technical Documentation fields have all been moving closer to those approaches advocated by the performance support community. Technical books have become interactive electronic manuals, stand-alone expert systems have been embedded in information systems, and instructor-led training courses have become Web-based training modules integrated with hyperlinked background reference information”.

2.4 A SOFTWARE ENGINEERING PERSPECTIVE ON THE DESIGN AND AUTHORING OF DATA-INTENSIVE WEB-BASED HYPERMEDIA APPLICATIONS

The main component of any advanced PSS is the technical documentation of the supported system/product. Technical documentations are, normally, large and complex applications that involve large amount of data. These types of applications are referred to as “data-intensive”. The complexity of these applications is proportional to the complexity of the supported system/product. As suggested in section (2.3.1), Web-based hypermedia is the main technology used for the development of these applications. Therefore, issues related to the design and authoring of data-intensive Web-based hypermedia applications are of paramount importance and directly affect the development of advanced PSSs. This section discusses these issues from a Software Engineering (SE) perspective.

2.4.1 Authoring: Techniques and Methods

2.4.1.1 Traditional Authoring

According to Csinger et al. [1995], authoring is about the tradition of *collecting*, *structuring*, and *presenting* information in the form of static documents rendered in some medium or media. The task of a “traditional” author is to collect a coherent body of information, structure it in a meaningful and interesting way, and present it in

an appropriate fashion to a set of readers or viewers of the eventual work. Aikat et al. [1996] points out some of the generic tasks involved in the authoring process, which include cross-referencing, building glossaries, chunking information into categories, indexing, utilising style guides and templates, selecting reader sequences, and using illustrations and graphical aids. The principal limitation of the traditional authoring approach is the production of “one-size-fits-all” static documents. This is symbolised by the familiar book format, which once printed, cannot be changed. The origin of this limitation lies in the “static” authoring approach, which commits the author to the form as well as to the content of the work, well in advance of the actual time at which it is presented.

Recent technological advances in the field of information processing, storage, presentation, and retrieval have had a great influence on the authoring process. However, there is a consensus among researchers that powerful and advanced authoring tools and technologies cannot help to improve the quality of the content of documentation unless similar powerful and advanced authoring *methods* are utilised [Price, 1997; Thibeu, 1996; and Csinger et al., 1995]. For instance, Csinger et al. [1995] argue that “we now have fast graphics, powerful reasoning engines and other technology, but what are we going to do with them?”. Thibeu [1996] discusses online authoring and states that “the tools available today are very powerful, and over time they will become even more powerful. However, no technology that is currently on the market, or that is even on the horizon, will help improve the quality of the content of these online applications. Unless the information meets reader needs, in the way reader needs to see it, these tools will never reach their potential”. He even went as far as arguing that technology can help readers get more information faster, but if it

is not the appropriate information, or if that information is incorrect, the technology could actually increase an organisation's error per hour. Moreover, Price [1997] points out that as the publishing system becomes much more complex, it exhibits emergent behaviours, and it demands new attitudes, concepts, and work from the technical communicators.

2.4.1.2 Structured Authoring

The above mentioned problems associated with the traditional static authoring approach affect the production of all types of documentation, including not only printed, but also electronic, hypermedia-based, and online formats. These problems are tackled through the introduction of new authoring attitudes and guidelines, and enhanced methods and techniques.

2.4.1.2.1 Structured Authoring Methods

Structured² authoring methods are controlled by explicitly-defined and abstract data models, notations, rules, guidelines, etc. Structured authoring supports common principal guidelines for enhancing the authoring process. These include the need to: (i) consider the end user (reader) at a very early stage of the authoring process, (ii) separate authoring activities such as structure, content and presentation, and (iii) break down the available information into fine-grained pieces of information (information elements), which can be easily managed and reused. For instance, Csinger et al. [1995] present the intent-based authoring method. This method is based on the

² Unless stated otherwise, the word “structured” is used throughout this thesis to refer to approaches that are controlled by explicitly defined models, notations, rules, guidelines, etc.

explicit identification of the intent(s) of the author when documents are specified, and the separation of information and presentation spaces. These spaces are bridged by various knowledge sources e.g. presentation, media, domain etc., in addition to a user model that permits user-tailored determination of content at run-time. Thibeau [1996] presents the structured writing method for online information. This method is based on an analysis of people (authors and readers), information, and technology. It also separates information organisation (structure) from presentation. Price [1997] presents an Object-Oriented (O-O) method for structuring information for electronic publication. He utilises ideas from O-O programming to clarify and revive the structure of the material in existing documents before publishing them online. Other research that describes structured methods for the authoring and management of complex hypermedia applications are found in [Wilkinson et al., 1999; Saarela et al., 1997; Lee et al., 1997; and Kim et al., 1996].

As far as the performance support domain is concerned, and in particular the technical documentation branch, the performance-centred authoring method arises as a result of the need to consider the tasks in hand with an emphasis on getting started quickly. The main focus of this authoring method is on tasks and users. This approach introduces a shift from the conveying of generalised information to the provision of specific (performance and task oriented) knowledge. It is reported in [Setchi, 2000] that the primary goal of the performance-centred approach is to reduce the “time of competency” by delivering “just-in-time” knowledge to users at the time they perform their tasks, i.e. at the right time, in the right place, in the correct amount, and in the most useful format. According to Raybould [2000], “just making knowledge available electronically is not sufficient. Only by having a performance centred interface built

on the knowledge base is the knowledge rendered useful to achieving business goals”. This approach is supported by McGraw [1997] through a task-based and user-aware interface which forms the foundation of all other performance support. Graham [1997] describes a method for designing documentation that relates information to acceptable user performance in acquiring specific skills and knowledge. The method involves selecting, organising, and presenting all information using a performance-based development model which maps the structure of information to user tasks.

2.4.1.2.2 Structured Authoring Techniques

Structured authoring techniques are formal to semi-formal standards that act as a framework for describing the documentation. For instance, the Standard Generalised Markup Language (SGML) [ISO, 1986] was developed for describing the structure of complex documents. According to Pham et al. [2000], structured documentation is most clearly seen in systems based on SGML, where a DTD (Document Type Definition) precedes authoring. DTDs specify what document elements are permissible and in what order they may appear. Based on SGML, simpler and more Web-oriented standards were also introduced including the popular Hypertext Markup Language (HTML), and the eXtensible Markup Language (XML) [W3C, 2000]. The latter is a set of rules for defining semantic tags that break a document into parts and identify the different parts of the document. Harold [1999] identifies three main features of XML: (i) it is a meta-markup language that defines a syntax used to define other domain-specific, semantic structured markup languages (e.g. MusicML, MathML, ChemicalML, etc.) (ii) its tags can be documented in a DTD (iii) it describes structure and semantics, but not formatting, which can be added to a

document using a stylesheet (e.g. CSS, XSL, etc). XML is useful for designing a domain-specific markup language, for self-describing data, for interchange of data among applications, and for large, complex, structured, and integrated data.

Virtual documentation is another structured authoring technique. Virtual documents are hypermedia documents for which the content of pages (nodes), including links, are created on-the-fly, as needed, and upon demand. This technique is mostly used for online (Web-based) authoring. According to Milosavljevic, [1999], there already exist several kinds of virtual documents on the Web for which the content is determined dynamically. First, a template can be used for which node contents are substituted at runtime. Second, applications can be used to generate values for one time use. Third, CGI scripts and search engines can be used to compose virtual documents from fragments of other documents for the user on demand. Fourth, metadata can be generated for summarization, where the extraction and summarization is done on the fly for the user. Finally, natural language generation techniques can be employed to dynamically construct virtual documents from underlying data contained in data or knowledge bases.

2.4.2 Systematic Development of Data-Intensive Web-Based Hypermedia Applications

2.4.2.1 Static Hypermedia Authoring

Hypermedia authoring is concerned with the authoring tasks of collecting, structuring, and presenting information, with an emphasis on creating nodes and relationships, or

in hypermedia terms pages and hyperlinks. Furthermore, different types of multimedia data elements have to be collected, edited, classified and structured to form the hypermedia network. The traditional static hypermedia authoring approach involves inserting hyperlinks within the documents, which refer to other documents by means of Universal Resource Locators (URLs). This simple hypermedia architecture makes it extremely easy to develop simple and small-size applications. However, as the application grows in size, with large numbers of documents and hyperlinks, i.e. it becomes data-intensive, it can become disturbingly unmanageable and authoring becomes a very complicated process, especially in a multi-author environment. In addition to the limitations associated with the static authoring approach (see 2.4.1.1), traditional hypermedia authoring has its own specific shortcomings. Kappe [1999] outlines these shortcomings, which he identifies as due to its “static structure”. These include *user disorientation* (lost in hyperspace syndrome), *broken links* which occurs when deleting a document without deleting its links from other documents, and *orphan documents* which occur when the last link pointing to a document is removed, and hence it becomes unreachable.

2.4.2.2 Problems Associated with Development of Data-Intensive Web-Based Applications

There is a consensus among researchers that there is a weakness in the current methodologies that support the development of Web-based applications, especially those for large and complex data-intensive applications [Brusilovsky et al, 2002; Gomez et al, 2001; Rossi et al., 2001; Segor et al., 2000; Fraternali, 1999; Coda et al, 1998; and Thibeau, 1996]. In contrast, there is no consensus on a general design

process model for Web-based applications. Currently, most Web developers manually generate low-level implementations of language-dependent files or use commercially available tools to produce them [Fraternali 1999]. This approach focuses on the physical Web pages (documents) in terms of their content, relationship to other pages, and graphical presentation. It is also an implementation-oriented technique with little or no attention to a formal requirements specification or to a design process [Segor et al., 2000]. Moreover, the use of such mechanisms is not guided by a systematic methodology³ that provides the developer with a higher-level view of the document structure and a well-defined development process supported by suitable tools [Coda et al., 1998]. Therefore, management tasks such as enforcing integrity constraints, updating, and restructuring of Web-based documents are tedious to perform. In addition, it is difficult to reuse previously developed artefacts [Wang et al., 1998]. These burdens are magnified, more critical, and more difficult to implement within data-intensive applications, containing large amounts of data.

2.4.2.3 Why a Systematic Development Approach?

According to Fraternali [2000], Fernandiz et al. [2000] and [1999], and Coda et al. [1998], it is important to adopt a systematic development approach with explicitly declared concepts and design models when developing and, then, maintaining data-intensive Web applications. A systematic development approach would:

- Enable the testing and verification of the application at a very early stage in the development process, as early as the design stage.

³ Systematic methodology: a methodology that has a well defined process and uses structured design methods with explicitly declared concepts, and models.

- Enable the enforcement of integrity constraints, which insure that the resulting application satisfies a set of desired properties and business rules. Fernandez et al. [2000] state that “it is essential to be able to verify constraints against the application’s definition, and not against a particular instance of the application”.
- Facilitate the creation of multiple versions or views of the system from the same data set. This feature contributes strongly to the adaptive knowledge delivery feature associated with advanced PSSs.
- Decrease the complexity, effort, and time to re-structure the application.
- Enable the automatic generation of the application’s code because formal and well-formed models are easily translated into code.
- Facilitate the reuse of previously developed artefacts and simplify the management of the application.

To help understand the current state of practice in Web-based development, Fraternali [1999] grouped tools for Web-based development into six categories: (1) visual editors and site managers, (2) Web-enabled hypermedia authoring tools, (3) Web-DBPL (database programming language) integrators, (4) Web form editors, report writers, and database publishing wizards, (5) multi-paradigm tools, and (6) model-driven application generators. According to him, the order of presentation of these different categories reflects the increasing level of support that tools in each category offer to the systematic development approach for Web applications. It is within the last category of tools (model-driven application generators) that software engineering principles are truly applied to Web-based development.

2.4.2.4 Software Engineering in Support of a Systematic Development Approach

Online technologies such as the WWW have dramatically changed the way information is presented and used, so much so, that we also need to have new theories and models for understanding how technology and content are related in this new communication environment. In pursuit of a software engineering approach to Web development, Coda et al. [1998] state that “An approach similar to software engineering approach has to be followed in order to bring WWW development out of its immaturity. The problem of WWW site development must be tackled by providing methodological and technological support for each phase of the development process”.

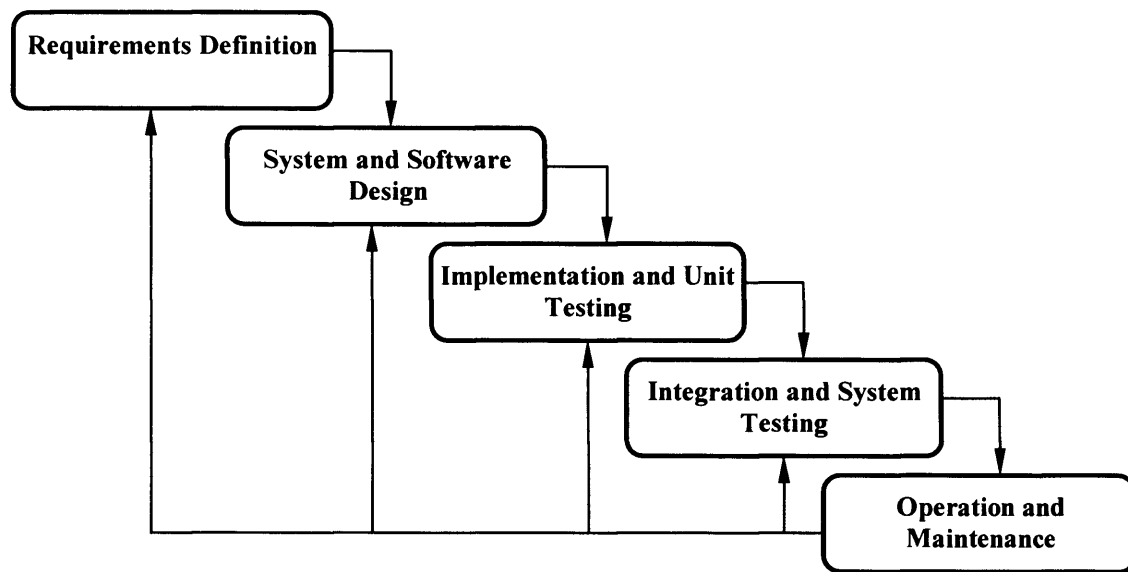
2.4.2.4.1 The Software Engineering Process

Software Engineering (SE) is an engineering discipline that is concerned with all aspects of software production. It is defined in [IEEE, 1990] as the application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software; that is, the application of engineering to software. This systematic approach is supported by a rigorous software process, SE methods, and (often) a set of Computer-Aided Software Engineering (CASE) tools. According to Sommerville [2001], the software process is a set of activities whose goal is the development of software. SE methods are structured approaches to software development, which include systems models, notations, rules, design advice, and guidance. These methods are normally supported by a set of CASE tools that provide automated support for software process activities.

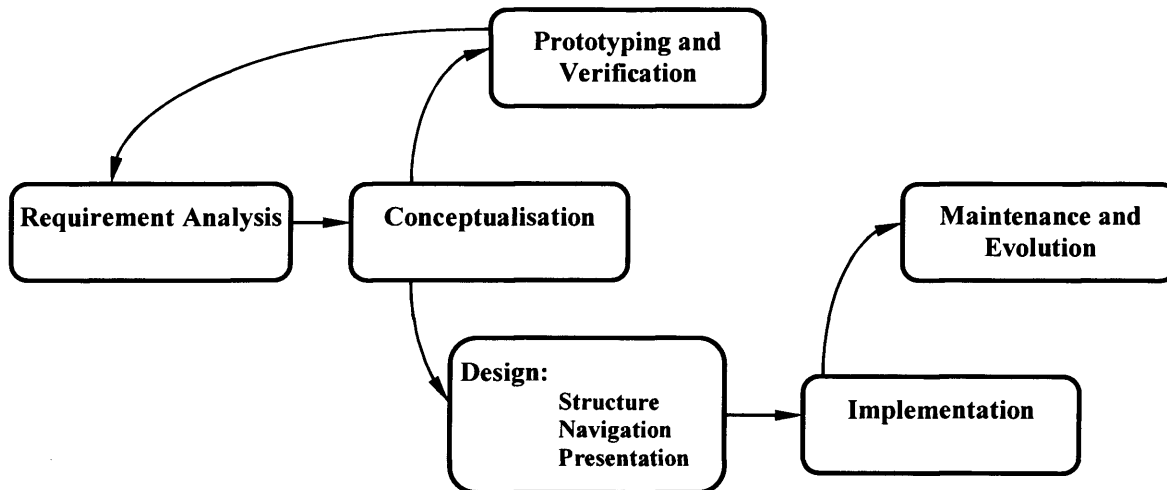
There are many abstract models for the software process, which include the *waterfall model*, the *evolutionary development model*, the *formal transformation development model*, and the *reuse-based development model* [Sommerville, 2001]. Furthermore, hybrid models such as the *incremental* and the *spiral* models have evolved, through the need to use different approaches for different parts of the system and to support process iteration where parts of the process are repeated as system requirements evolve. This list is not an exhaustive list and some organisations have even developed their own tailored and *ad hoc* models. Even within the same organisation, different software processes can be employed. However, there are fundamental activities, which are common in most process models, including software specification, design, implementation, and validation. From these models, the “waterfall” model was chosen to clarify the SE process because it partitions the software development process into a set of distinctive stages, and identifies the process activities in an explicit manner. Figure 2.2(a) outlines a generic “waterfall” model for the software development process [Sommerville, 2001].

2.4.2.4.2 Software Process for Web-Based Information Systems

Web-based information systems are the new generation of information systems that have emerged in response to the advances in Internet technology. They can be described as a hybrid between a hypermedia system and an information system. Their development relies on the adaptation of techniques originated in the SE [Sommerville,



(a) The Waterfall Model for the Software Development Process [Sommerville, 2001]



(b) The Lifecycle of a Web Application [Fraternali, 1999]

Figure 2.2 Software Process and Web-Based Development

2001], structured hypermedia [Ceri et al., 2000; Kemp et al., 1999; Schwabe et al., 1998; Isakowitz et al., 1995; Halasz et al., 1994; and Garzotto et al., 1993], and database [Oxborrow, 1989] disciplines. These techniques are blended into a development methodology and supporting environment. The development of Web-based applications benefits from the advantages provided by these disciplines. For instance, SE provides a firm and systematic development approach, structured hypermedia design provides powerful modelling methods, and database technology provides architectural solidity for structured data. However, it is of utmost importance to take into consideration the unique requirements and characteristics of Web-based applications.

The utilisation of well-tested SE methods when developing Web-based applications enables the inheritance of all the advantages, experience, and knowledge associated with this well-established engineering discipline. In [Fraternali et al., 2000] it is stated that “the development of Web applications needs to be organised into a well-defined process, amenable to the benefits of software engineering”. These methods must consider the specificity of hypermedia as an authoring tool and the Web as a novel communication medium [Rossi et al., 2001 and 1995; Segor et al., 2000; Fraternali, 2000 and 1999; and Coda et al., 1998].

In contrast with the applications of traditional information systems, Web-based applications have a more diverse, less structured, and more heterogeneous information space. They also have more emphasis on navigation paths and on information access in an exploratory manner rather than through “canned” interfaces. These applications

are used by a large variety of end-users in terms of their experience and background knowledge and they typically use high quality graphics to support their presentation.

Fraternali [1999] presents a generic process model for the development of Web-based applications that is based on SE principles. The Web application life cycle, which is shown in Figure 2.2(b) is organised into six main stages. These are requirement analysis, conceptualisation, prototyping and verification, design, implementation, and maintenance and evolution. By comparing the two approaches depicted in Figures 2.3 (a) and (b), it can be deduced that the latter is a specialisation of the former with added emphasis on conceptual modelling, and Web-based and hypermedia-specific design parameters such as navigation. In addition, this model clearly adopts the separation of design activities (structure, navigation, and presentation), which is one of the main principles of structured authoring (see 2.4.1.2.1).

The domain conceptualisation in Web-based applications focuses on capturing abstract objects and relationships as they will appear to users, rather than as they will be represented within the software system. These abstract objects and relationships are abstracted using different modelling schemas such as E-R diagrams [Ceri et al., 2000; Fraternali et al., 2000; and Isakowitz et al., 1995], O-O models [Schwabe et al., 1998 and Rossi et al., 1995], UML class diagrams [Gomez et al., 2001; Rossi et al., 2001; and Segor et al., 2000]. Structure, navigation, and presentation designs are sequentially executed within the design stage. The structure design is concerned with constructing the system's backbone by identifying the information objects that constitute each page and the hierarchical organisation of these pages [Price, 1997; and Langer et al., 1994]. The navigation design is concerned with the access methods for

pages and the linkage criteria, i.e. the contextual and non-contextual relationships between pages and the way they interact with each other [Rossi et al., 2001 and Fraternali et al., 2000]. The presentation design is the visual specification of the application's interface, which determines its appearance to the users [Rossi et al., 1995].

A very similar SE approach to development of Web-based applications was suggested by Coda et al. [1998]. They proposed to break down the development process into a number of phases, namely requirements analysis and specification, design, and implementation. After the application has been implemented and delivered its structure and content are maintained and evolved.

2.4.3 Structured Hypermedia Design

The development of data-intensive Web-based hypermedia applications is usually a collaborative process involving team members with different expertise, knowledge, skill(s), aims, and backgrounds. The development team may consist of domain experts, content managers, system analysts, programmers, authors, hypermedia editors, style architects, graphics designers, Web administrators, and others. This list is not necessarily exhaustive nor does it imply that all these roles will require separate individuals. Due to this team effort, it is vital to acquire a common communication language between team members, which maps their understanding into a uniform model [Ding et al., 2002; Klusch, 2001; and Levy and Weld, 2000]. Therefore in addition to the reasons mentioned in 2.4.2.2, it is essential to adopt a structured design

approach, i.e. explicit and declarative models, and to use high-level and implementation language-independent primitives for the description of these models.

In their attempt to classify and categorise hypermedia design methodologies, Kemp et al. [1999], state that “there appear to have been no efforts as yet to develop any kind of comparative framework specifically for the study of hypermedia design methodologies”. They also state that “the task of comparison between hypermedia design methodologies is fraught with difficulties”. In addition, they argue that whether one considers hypermedia system design in terms of methodologies or in terms of authoring strategies says much about one’s preconceptions about the design process i.e. do we write hypermedia or do we design it? A more recent survey of hypermedia design methodologies can be found in [Suh et al., 2001].

In order to establish an up-to-date insight into structured hypermedia design, a review of the related research has been conducted. This includes endeavours related to:

- ***General hypertext reference models:*** [De Bra, 1999; Garzotto et al., 1995; Halasz et al., 1994; and Schnase et al., 1993],
- ***Structured documentation:*** [Pham and Setchi, 2001; Pham et al., 2000; Tucker et al., 1997; and Price, 1997],
- ***Hypermedia design methodologies:*** [Suh et al., 2001; Ceri et al., 2000; Schwabe et al., 1998; Coda et al., 1998; and Isakowitz et al., 1995],
- ***General-purpose structured hypermedia systems:*** [Rossi et al., 2001 and 1995; Segor et al., 2000; Fraternali et al. 2000; Staab et al., 2000; Takahashi et al., 1997, and Langer et al., 1994],

- *Hypermedia authoring tools*: [Fernandez et al., 2000 and 1999; and Kappe, 1999],
- *Modelling languages and notations*: [Ceri et al., 2000 and Wang et al., 1998].

It is important to note that the boundaries between these research areas are not clear cut, but rather overlap. For example, the work introduced in [Isakowitz et al, 1995] can be considered as a reference model and as a design methodology, while the work introduced in [Ceri et al., 2000] can be considered as a design methodology and as a modelling language.

At this stage, it is important to differentiate a hypermedia *system* from a hypermedia *application*, and accordingly to differentiate their design methodologies. According to Isakowitz et al. [1995], the former is an environment that facilitates the creation of the latter. They state that “a data model for a hypermedia system details its internal architecture but is of little value in modelling hypermedia applications. This is because describing the layout of a general purpose engine is quite different from modelling an application domain”.

2.4.3.1 Reference Models for Hypermedia Systems

The Dexter hypertext reference model [Halasz et al., 1994] is an early attempt to provide a principled basis for comparing hypermedia systems as well as for developing interchange and interoperability standards. The model is divided into three layers namely the *storage layer*, which describes the network of nodes and links, the *run-time layer*, which describes mechanisms supporting the user’s interaction with the hypertext, and the *with-in component layer*, which covers the content and

structures within the hypermedia nodes. The AHAM reference model [De Bra, 1999] is an extended version of this generic reference model which accommodates the fundamental principles in adaptive hypermedia. The MUCH (Many Using and Creating Hypermedia) and the RICH (Reusable Intelligent Collaborative Hypermedia) systems by Wang et al. [1998 and 1995] are also based on the dexter hypertext model and treat the storage layer as a semantic network. Although it can be used in the design of hypermedia applications, the Hypertext Design Model (HDM) by Garzotto et al., [1995], is another hypertext reference model characterised by the adaptation of techniques used in SE and database design. A less used reference model is the HB1 [Schnase et al., 1993], which is referred to as a Hyperbase Management System (HBMS).

2.4.3.2 A Review of Structured Design Methods for Hypermedia Applications

This review of structured design methods for hypermedia applications included the following exclusive list of methods:

- Object-Oriented Hypermedia Design Method (OOHDM) [Schwabe et al., 1998].
- Relationship Management Methodology (RMM) [Isakowitz et al., 1995].
- HDM extended methodology HDM-Lite [Fraternali et al., 2000].
- Web Modelling Language WebML [Ceri et al., 2000].

A comparison between these methodologies is outlined in Table 2.2, which gives the primary modelling approaches, the design phases, and a brief description of these methodologies. Furthermore, in a recent survey, Suh et al. [2001] identify more hypermedia design methodologies, which in addition to the ones mentioned above

Table 2.2 Comparison - Structured Hypermedia Design Methodologies

	OOHDM	RMM	HDM-Lite	WebML
Primary Modelling Approach	O-O	E-R	E-R	E-R
Design Phases	<ul style="list-style-type: none"> ▪ Conceptual design ▪ Navigational design ▪ Abstract Interface design 	<ul style="list-style-type: none"> ▪ E-R design ▪ Slice (Entity) design ▪ Navigation design ▪ Conversion Protocol design ▪ User Interface design ▪ Run-time design ▪ Construction design 	<ul style="list-style-type: none"> ▪ Structure design ▪ Navigation design ▪ Presentation design 	<ul style="list-style-type: none"> ▪ Data design ▪ Hypertext design <ul style="list-style-type: none"> – in-the-large – in-the-small ▪ Presentation design ▪ User and Group design ▪ Customisation design
Brief Description	Based on HDM. Describes a four-step process towards building hypermedia applications, beginning with domain analysis and proceeding through navigational design and abstract interface design to final implementation.	Based on data modelling techniques, specifically E-R modelling. This methodology is most suited to applications that have a regular structure, especially where there is a frequent need to update the information to keep the system current.	Includes a notation for specifying presentation at a conceptual level, which coupled to primitives for describing structure and navigation. It covers all aspects of a Web application and enables automatic implementation.	Includes a notation for specifying complex Web sites at the conceptual level. It enables the high level description of Web-based hypermedia under distinct orthogonal dimensions namely data content, the pages that compose it, the topology of links between pages, the layout and graphic requirements for page rendering, and the adaptive features of content delivery.

include Scenario-Based Object Oriented Design Methodology SOHDM, Enhanced Object-Relationship Model EORM, Workflow-Based Hypermedia Development Methodology WHDM, and Index-Driven Hypermedia Design Methodology IHDM. They categorise hypermedia applications, with regard to their purpose, into two types: (i) process-oriented applications for supporting organisational processes, and (ii) content-oriented applications focusing on information services. Accordingly, hypermedia design methodologies can be classified into a *task-driven* approach (SOHDM, EORM, and WHDM) and a *content-driven* approach (OOHDM, RMM, HDM-Lite, WebML, and IHDM). The primary objective of the former approach is to support organisational tasks, i.e. tasks are defined first and then the content required for them is tackled. In contrast, the main objective of the latter approach is organising hypermedia information in a manner to satisfy the users' cognitive demands.

It can be noticed that these design methods attempt to achieve a set of common objectives. These objectives are within the founding principles of SE as outlined by Coda et al. [1998], which include:

- ***Rigor and formality*** to provide a clear definition of the entities involved in the design process, their relationships, and associated semantics;
- ***Clear separation of concern*** within the hypermedia document components, namely into content, structure, navigation, and presentation. In adaptive hypermedia, this principle is extended to include the separation of adaptation issues from other document components;
- ***Modularity*** to divide a complex system, using construct and abstractions, into smaller and simpler components, which can then be easily used and reused;

- ***Abstraction*** away from low level, unimportant details, identifying important concepts and relationships;
- ***Anticipation of change*** which provides for the application's maintenance and evolution, and
- ***Generality*** through mechanisms that support the development of implementation constructs and allow the developer to create *ad hoc* constructs and to customise the existing ones.

These principles were, to a certain extent, taken into consideration, either explicitly or implicitly, and implemented in all the reviewed research endeavours. These design methods were described as structured mainly because they satisfy and adhere to the principles of software engineering.

Finally, this review of methodologies aimed to identify some of the “well-known” approaches to structured hypermedia design. The review is not intended exhaustive, however, many other methodologies are available for generic or special-purpose use.

2.5 SUMMARY

In pursuit of the “optimal” PSS, the objectives, forms, and characteristics of traditional paper-based and electronic PSSs have been investigated. In addition, a review of the state of practice in recently developed PSSs has been conducted. As a result, a conceptual model for advanced PSSs has been synthesised. Furthermore, the state-of-the-art technologies that can be utilised to achieve this advanced performance support concept have been discussed. Particular attention has been devoted to the

integration of these technologies in order to provide the advanced characteristics required for these “optimal” PSSs.

Web-based hypermedia has been suggested as the main technology for the development of technical documentation applications which are the core of any advanced PSS. These applications are normally characterised as data-intensive, therefore, issues related to the design and authoring of data-intensive Web-based hypermedia applications from a SE perspective have been discussed. The chapter has reviewed traditional and structured approaches to authoring, highlighting the limitations of the former approach and the benefits of the latter approach. In particular, research that supports a systematic development approach to Web-based development has been presented. This systematic approach has a well defined development process which uses explicitly declared concepts and models. It utilises techniques that originated in the SE, structured hypermedia, and databases disciplines. Structured hypermedia design, which is considered to be pivotal for systematic development approaches, has also been reviewed in terms of its reference models and design methods.

CHAPTER 3

USAGE-BASED DATA MODEL FOR THE DESIGN OF TECHNICAL DOCUMENTATION

This chapter presents a usage-based data model for the design of technical documentation. First, a semantic data model for designing technical documentation is proposed based on an abstract usage analysis of technical information. This analysis is aimed at abstracting the intended purpose of the documentation, the tasks supported by the documentation, and the functional characteristics of documents. Next, these abstractions are integrated into a usage-based semantic network and a set of rules and constraints, which are then mapped into a database schema. Finally, a case study is conducted using a product-related technical manual in order to demonstrate the validity of this design approach.

3.1 MODELLING KNOWLEDGE WITHIN TECHNICAL DOCUMENTATION

The main objective of domain data modelling is to transform aspects of the real world into a formal data model and to provide a commonly agreed framework for the domain, which can be reused and shared between authors, developers, tools, and applications. If the information requirements of an organisation are to be satisfied, it is essential that the data model adequately reflect reality. Reality is described in

[Oxborrow, 1989] as effectively boundless and with no horizon, so that only perceived reality can be modelled and hence bounded.

In the technical documentation domain, authors need to organise and classify the domain data in a way that closely links them with their future use. It is essential to identify the knowledge or skill that the user intends to acquire by accessing a specific document at a very early stage in the technical documentation development process. This rapid identification enables the delivery of the exact required information to users to enhance their knowledge about a certain domain related topic. Data modelling involves shaping the facts collected during the data analysis process into data model concepts. The basic facts, plus some constraints and rules, are of primary importance at this stage.

3.1.1 Usage-Based Analysis of Technical Information

Information has embedded structures, and these structures vary based on the intended use of information. It is important for the author to categorise the information fragments used in the documentation into abstract types and sub-types that reflect their subsequent usage [Pham et al., 2003]. Technically-oriented documentation can be analysed with regard to its usage from three different perspectives: (i) the intended purpose of the documentation, (ii) the tasks supported by the documentation, and (iii) the functional characteristics of documents.

3.1.1.1 Purpose of Technical Documentation

Technical documents can be categorised with regard to the context in which people will use their information and the intended content of these documents i.e. their purpose of existence. A set of high level and abstract categories are realised by conducting a purpose-driven analysis of information within technical documentation. For instance, Thibeau [1996] introduces two alternative methods for categorising technical information with regard to their usage:

- *Action versus Knowledge*: **Action** information describes the set of skill-related information, which tells the user what to do. On the other hand, **Knowledge** information describes the set of information that assists the user in performing actions,
- *Main versus Support*: **Main** information describes the set of primary, required, and “must know” information. On the other hand, **Support** information describes the set of the supplementary and “no harm in knowing” information.

The former categorisation is more suitable for instructional-centred documentation. Identifying and distinguishing between action- and knowledge-based documents requires considerable knowledge of the content of these documents. On the other hand, the latter categorisation requires an early understanding of the needs of the users in order to identify and distinguish between “main” and “support” documents. Moreover, Pham et al. [2000] identify the types of information contained in technical documentation and in technical manuals in particular, as **action**, **support**, and **planning**.

Accordingly, Figure 3.1 presents an abstract classification paradigm for technical documents according to their purpose. The figure outlines two main categories, which include:

- *Action*: this is skill-related information that assists a person in doing something with or to the system/product. Documents of this type are concerned with answering questions about the supported system/product such as how to use it, how to prepare it for use, how to keep it working, how to change it, how to dispose of it, etc.
- *Knowledge*: this category can be further classified into two abstract sub-categories: (i) *support* which is supplementary information that principally exists to assist in answering questions related to enhancing knowledge about the system/product, such as how it works, what it consists of, etc. (ii) *planning* which is fundamental information that principally exists to assist in answering questions related to the management of the system/product, such as what is it for, and what is done when, etc.

3.1.1.2 Tasks Supported by Technical Documentation

From a performance support perspective, the main objective of technical documentation is to support users to accomplish their work tasks. These tasks can be abstracted and categorised at the design stage. The categorisation of the tasks supported by the technical documentation enables the association of every document with a specific task. In this way the type of the task being later performed by the user is able directly to influence the delivery of documents that will support his/her

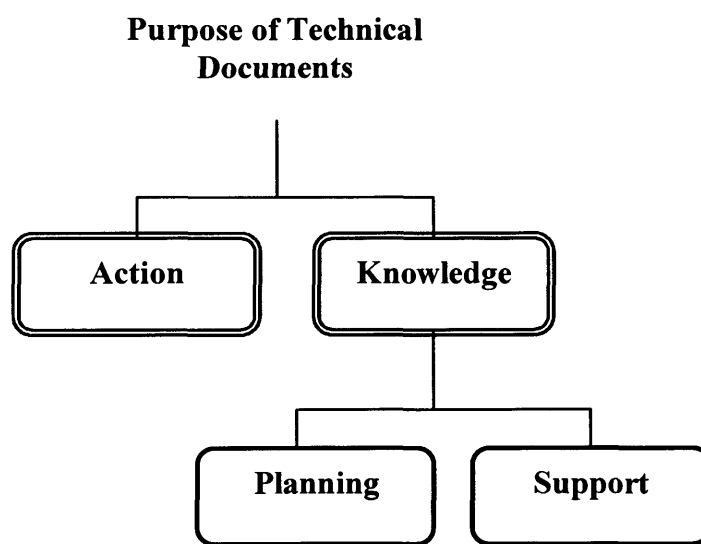


Figure 3.1 Classifications of Technical Documents According to their Purpose

performance, i.e. enabling performance-based information retrieval. It is important to design the technical documentation in a way that relates information to acceptable user performance in acquiring specific skill and knowledge.

The tasks that users need to accomplish reflect their objective(s) of accessing the technical documentation i.e. their cognitive demands. In general, these objectives belong to one of two abstract types: *learn* or *perform*. Learning tasks are described by static information about the system/product. In contrast, performing tasks are described by information that illustrates some dynamic system/product-related action. These two abstract task types can be further broken down into more specific and “primitive” tasks. Despite their variety and diversity, the identification of these primitive tasks depends on the objective that they will convey and the application domain. The domain expert can identify a list of primitive tasks by considering the principal distinctive features of the system/product and its supported data. A sizeable list of primitive tasks is available for domain experts to choose from, for instance, *introduce, plan, assess, launch, handle, check, assemble*, etc. For example, the first three primitive tasks are of type “learn” while the remainder are of type “perform”.

Figure 3.2 outlines an example set of primitive tasks and their abstract types. An exhaustive list of primitive tasks can be finally obtained depending on the application domain. For example, Graham [1997] identified a large list of primitive tasks for operation and maintenance that included, operate, configure, describe, monitor, test, order, plan, replace, bill, and install.

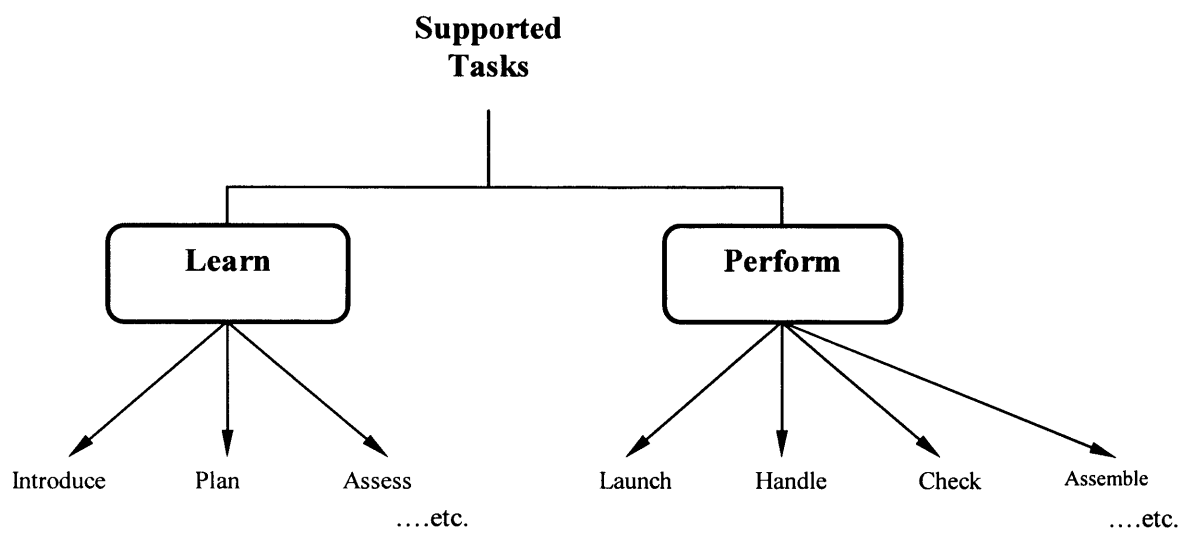


Figure 3.2 Primitive Tasks and Abstract Task Types

The strength of task modelling can be observed when integrated with the information purpose model, described earlier. As it will be described later, these two abstractions provide a firm framework to model the application domain information.

3.1.1.3 Functional Characteristics of Technical Documents

In general, the word “function” is defined in [Oxford Dictionary, 1998] as a “*proper or necessary role, official or professional duty*”. Every individual document, which contributes to a domain topic, has a functional characteristic that reflects its proper role in describing this concept, and the exact type of knowledge conveyed when invoked by the user. Functional characteristics of documents are used to optimise the delivery of knowledge by enabling advanced retrieval of information through the adaptation of information content and presentation to pre-determined users’ knowledge and preferences, and hence, enhancing the user’s job performance.

Characterising documents according to their function was employed in [Pham and Setchi, 2003] and [Thibeau, 1996]. They identified a list of functions, which includes procedure, process, structure, concept, principle, fact, definition, description, example, explanation, comment, requirement, recommendation, reason, and classification. Other functional characteristics of information elements may also be identified such as specification, clarification, advice, etc. It is suggested that these functions can be abstracted into six main categories based on the similarity of the knowledge that they convey and their usage, namely, Fundamental (Fund), Clarification (Cla), Procedure (Proc), Advice (Adv), Specification (Spec), and Organisation (Org).

Definitions, facts, principles, and concepts usually state essential, **fundamental** or generic knowledge. **Procedures** and processes are utilised when a continuous or discrete series of events takes place. Examples, comments, and illustrations are used for **clarification**. Recommendations, requirements, warnings, and cautions provide **advice**, commands and explain motives. Descriptions and explanations are used for conveying deep knowledge and additional detailed **specification**. Finally, structures and classifications are employed to describe a particular type of **organisation**. Table 3.1 outlines the functional characteristics of documents and their abstract types. This list of functions is not definitive or exhaustive and many others may be added to the list while others may be omitted from the list by the domain expert. In addition, some functions can be placed into a separate category in order to highlight their importance in illustrating the system/product information, while others can be incorporated within other existing categories.

3.1.2 Semantic Data Model for Representing Knowledge in Technical Documentation

3.1.2.1 Knowledge Representation Using Semantic Networks

A semantic network is defined in [Rolston, 1988] as a graphical representation of relations between elements in a domain. The basic components of a semantic network are *nodes* and *links*. Nodes are used to represent domain elements and are labelled with the element's name. Links (or arcs) represent relations between elements and are labelled with the name of the represented relation. Wang et al. [1998] perceive

Table 3.1 Functional Characteristics of Documents and Their Abstract Types

Abstract Type	Functional Characteristic			
Fundamental(Fund)	Definition	Fact	Principle	Concept
Procedure(Proc)	Series of Events	Process		
Clarification(Cla)	Example	Comment	Illustration	
Advice(Adv)	Recommendation	Requirement	Warning	Caution
Specification(Spec)	Description	Explanation		
Organisation(Org)	Structure	Classification		

semantic networks as directed graphs in which concepts are represented as nodes and relations between concepts are represented as links. The graph becomes semantic when each node and link is assigned a meaning. They also identified the benefits of using semantic networks in information management as to provide consistent categories of all concepts represented in the domain, and to provide a set of useful relations between these categories of concepts.

Perhaps the most important feature of semantic models is their ability to construct complex element types from atomic types. The two most common types of abstract relationships in semantic networks are *generalisation (is_a)* and *aggregation (part_of)*. As described in [Ter Bekke, 1992], generalisation is the recognition of similar properties of various types and combining these in a new property; this is sometimes referred to as an “inheritance link”. Aggregation is a collection of a certain number of properties in a type, which itself can be regarded as a new property.

Many researchers [Staab et. al., 2000; Wang et. al., 1998; and Schnase et. al., 1993] have highlighted the benefits of using semantic networks in modelling hypermedia systems in particular. They argue that the logical model of hypermedia is largely that of a semantic network and that hypermedia systems, with their node/link metaphor, naturally coincide with semantic networks. Typed links carry some semantic information, which provide potential for a system to manage data more efficiently on behalf of the user of the system.

3.1.2.2 Usage-Based Semantic Network for Technical Documentation

A deterministic factor in information retrieval is the user's main request or pursuit. Activity is defined in [Oxford Dictionary, 1998] as "an occupation or pursuit". Within the context of technical documentation, user activities are the user's general motive for accessing a particular topic or piece of information. Thus, the system's supported activities should match with the pursuit of the users.

Normally, when a user refers to the technical documentation, s/he has in mind an informational purpose, and an associated task to accomplish. The abstract activities supported by the documentation can be identified by logically integrating the supported tasks and the information purpose categories. Recall that abstract task categories are "learn" and "perform", and abstract information purpose categories are "planning", "support", and "action". Planning and support information is associated with knowledge acquisition i.e. learning, while action information is associated with skill and performance enhancement. The activities supported by the documentation can be classified using different abstract views, which depend on the knowledge that they provide. High-level and abstract user activities are identified as:

- Learn planning information.
- Learn support information.
- Perform action.

The semantic network shown in Figure 3.3 is based on the usage analysis of domain information described earlier. The network uses high-level primitives in its description. These primitives include information purpose, task, activity, function,

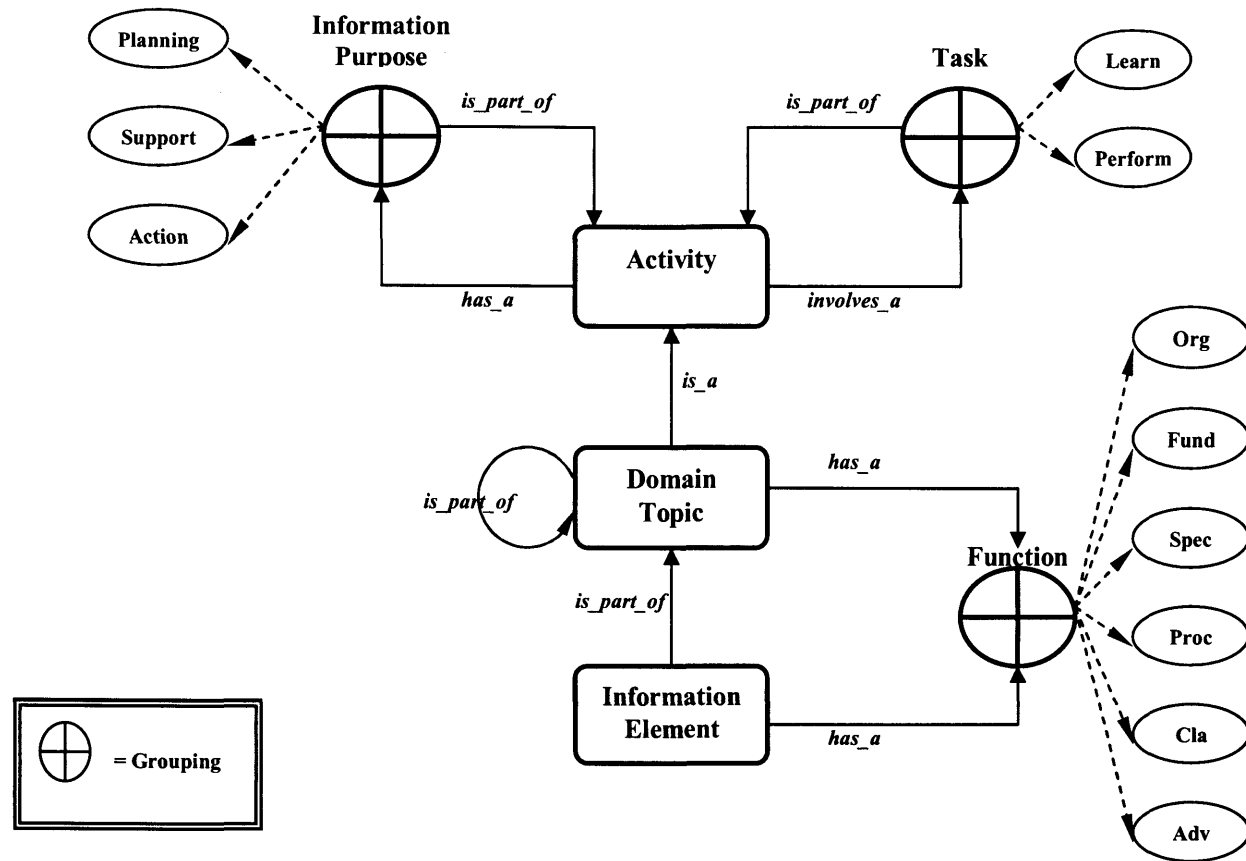


Figure 3.3 Usage-Based Semantic Network for Technical Documentation

domain topic, and information element. The semantic network is supported by the following hypotheses:

- ◆ Every activity has an informational purpose and involves a task.
- ◆ A domain topic is an abstract entity, which has an organisational function. It acts as a data structure, which organises information elements. Every domain topic is associated with a user activity, and it inherits the activity's informational purpose and the associated task. In addition, a domain topic can be aggregated from one or more topics and/or information elements.
- ◆ Information elements (documents) are discrete pieces of information that describe or contribute to the description of a domain topic. The aggregation of one or more information elements creates a domain topic. Within a domain topic, every information element has a function to perform, which describes its role. Information elements are reusable i.e. the same information element can perform different functions or roles within different topics.

The *grouping* constructor represented by the \oplus symbol is used to represent entities of the same category. It may be noticed that the activity entity performs a central role within the model. It integrates the information purpose and the system-supported tasks and associate them with their corresponding domain topics and vice versa. The entities “domain topics” and “information elements” will be fully described in the hypermedia authoring process (Chapter 4). At this stage, it is sufficient to know that they are abstract entities performing distinct functions. The former has a unique organisational role, i.e. its function can only be of type “organisation”. The latter can perform any of the other types of function depending on its role within the illustrated

topic. In the next section, it will be seen how easily this semantic network can be represented by means of a relational database schema.

A number of semantically related rules and constraints can be applied to the relationships of the semantic data model. These rules govern the validity and applicability of some types of relationships. For example, a task of type “perform” can only be associated with “action” type information within a certain activity. This relationship results in a “perform action” activity which is described using a hierarchy of documents. These documents are themselves described using information fragments of function type “procedure” or can be further illustrated using a “clarification” or “advice” type fragments. Similarly, a task of type “learn” can be associated with “planning” or “support” type of information within a certain activity. The former relationship results in a “learn planning” activity, and the latter results in a “learn support” activity. A “learn planning” activity is illustrated using “fundamental” type of information fragments, and a “learn support” activity is illustrated using “specification” type of information fragments. Both activities are further illustrated using “clarification” or “advice” types of fragments. Figure 3.4 illustrates these semantic constraints and rules. The full benefits of these rules and constraints can be appreciated at the structure building phase of the hypermedia authoring process, where a special tool is developed to apply these rules and to guide the author towards constructing a semantically valid and credible hypermedia-based technical documentation structure.

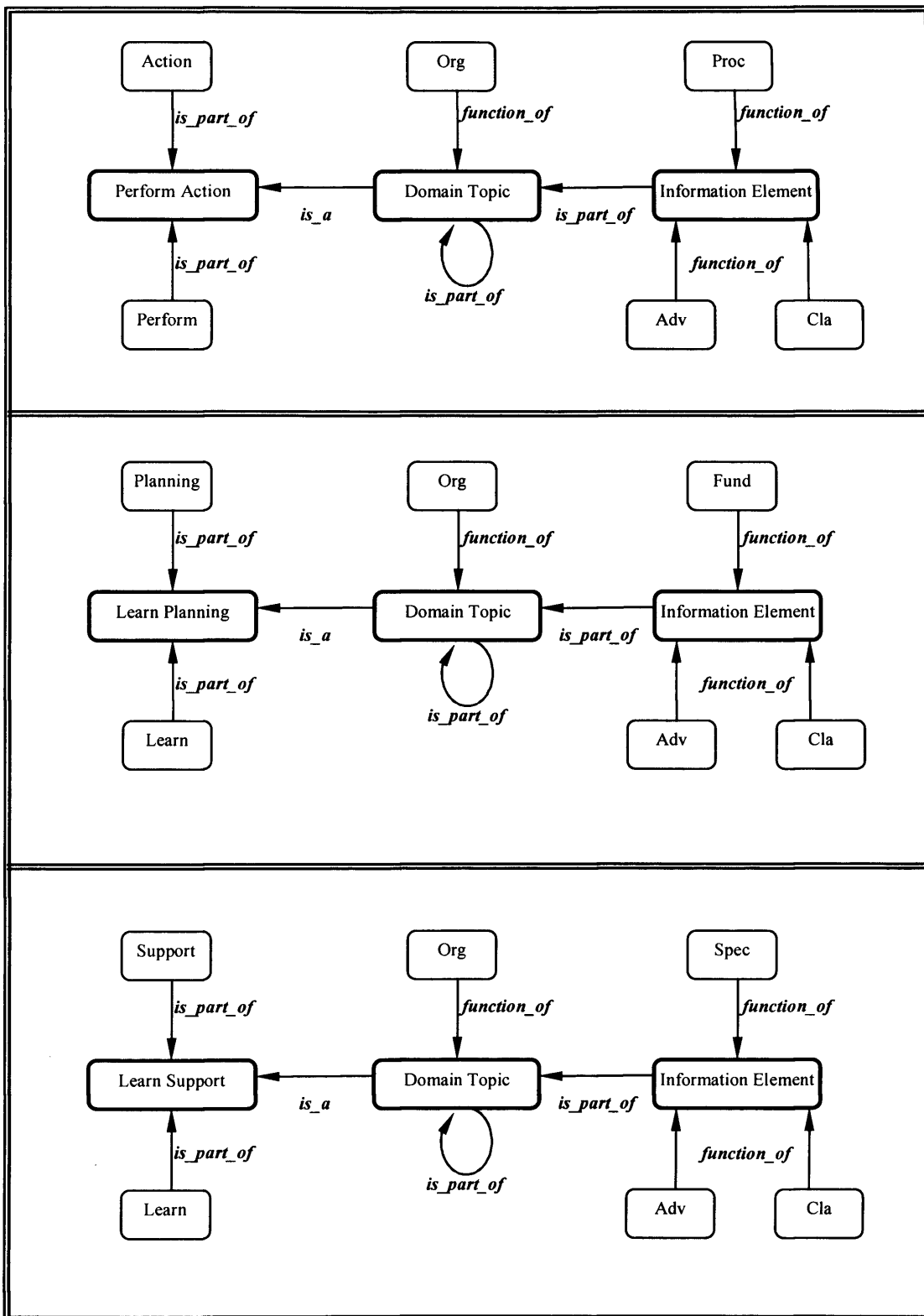


Figure 3.4 Semantic Rules and Valid Relationships

3.1.3 Mapping from the Semantic Data Model to a Database Schema

Support for rich semantics will transform data management into knowledge management and databases into knowledge bases. The mapping of the semantic network into a database schema results in a domain-independent knowledge base that can preserve data semantics and constraints. This knowledge base can then be processed by the author(s) of the technical documentation. The entities abstracted in the semantic data model represent “data about data” and are often referred to as *metadata*. They provide important semantic knowledge about the data that will be processed at a later stage. The metadata will be used not only during the design of the technical documentation, but also all the way through the system’s development process which include, authoring, documents generation, and maintenance.

An important feature of semantic networks is their natural ability to be converted into database schemas. Indeed, they were initially introduced in the early 1970s to facilitate the design of these database schemas [Schnase et al., 1993]. Nodes and links in a semantic network can be converted into entities and relationships, respectively, in an Entity-Relationship (E-R) schema. The derived schema is generic and independent of any specific Database Management System (DBMS) constructs, rules, and limitations. According to Oxborrow [1989], a semantic data model, independent of any DBMS has the following advantages:

- ♦ **Generality**: due to its independence, the best methodology could become a standard for data modelling.

- ◆ **Flexibility:** an independent semantic data model can be mapped into a number of commercially available DBMSs data models and hence would be suitable for use as a global model in a distributed database system.
- ◆ **Integrity:** the semantic rules and constraints could be used as the basis for developing application programs which maintain data integrity, by insuring that these rules and constraints are not violated.

E-R diagrams are considered to be the most commonly used techniques for graphically modelling database schemas. Figure 3.5 shows an abstract E-R diagram representing the semantic network described in the last section. This schema constitutes the knowledge base or *metadatabase*. This is an abstract E-R diagram because some details such as the attribute(s) of each entity have been deliberately omitted. Moreover, category, task, activity, topic, and information element are domain-related entities, i.e. their data differ with regard to the domain. As described earlier, information purpose types are *planning*, *support*, and *action*, task types are *learn* and *perform*, and abstract functions are *Org*, *Fund*, *Spec*, *Proc*, *Cla*, and *Adv*.

At this stage, the knowledge base of the technical documentation is ready for the introduction of domain-related information. These include the main categories of information, the primitive tasks, activities, domain topics, and information elements. The combination of information categories and primitive tasks will provide a set of domain related activities. Domain topics are then associated with the appropriate activity. Information elements are granted an abstract function or role and are associated with the appropriate domain topic. This process will be described in more detail in a case study, which is outlined in the next section.

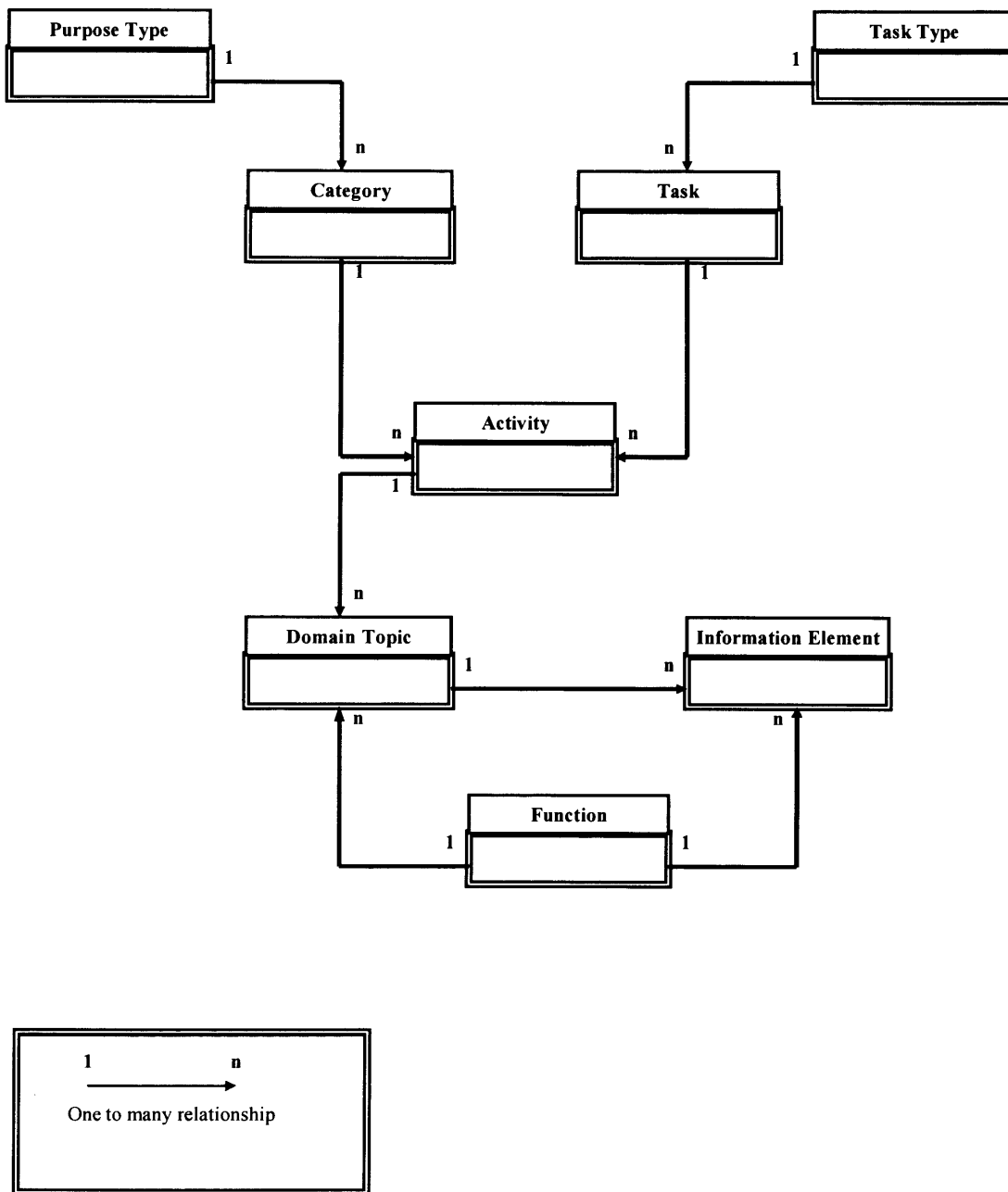


Figure 3.5 Abstract E-R Diagram Representing the Semantic Database Schema

3.2 CASE STUDY: USAGE-BASED ANALYSIS AND DATA MODEL FOR TECHNICAL MANUALS

The primary objective of this case study is practically to demonstrate the semantic modelling of technical documentation. This section briefly introduces technical documentation through technical manuals. Next, it demonstrates the usage analysis of product-related information. Results from this analysis are used to build a semantic database schema for the development of technical manuals. Finally, the user interface which is built on top of the database tables in order to assist in editing the schema is introduced.

3.2.1 Product-Based Technical Manuals

Technical manuals are special types of PSSs. The British Standards Institute [BS 4884, 1992] defines them as:

“A generic term for any document that explains how to use, maintain and handle a product from its delivery to its disposal, and in addition gives any technical information that a user is likely to need during the life of the product.”

This generic definition can be applied to any form of technical manuals, including paper-based, electronic, Web-based, etc. The main purpose of product documentation as defined by Graham [1997] is to communicate specific information to users in such a way to enable them to demonstrate the acquisition of specific skills and knowledge.

The users of technical manuals may include product purchasers, installers, trainers, operators, maintainers, product developers, etc.

Product-based technical manuals were chosen to demonstrate the semantic data modelling approach of technical documentation because of the special characteristics that exist within their information space, which can comprehensively illustrate this approach. These include:

- The diversity of their purpose, i.e. they contain planning, support, and action type of information
- They support both learning and performing types of tasks
- They, therefore, support a variety of distinct activities
- Their information base contains information elements that perform many types of roles and functions
- Most technical manuals are data-intensive, i.e. they contain large amount of data.

3.2.2 Information Analysis of Technical Manuals

British standards for technical manuals [BS 4884, Part1, 1992; BS 4884, Part2, 1993] are the main source of knowledge in the domain for technical manuals. They specify the full range of abstract information categories that a user of a product manual might need, including:

1. ***Purpose and Planning (PP)***. What is the product for?
2. ***Operating Instructions (OI)***. How to use the product?
3. ***Technical Description (TD)***. How does the product work?

4. ***Handling, Installation, Storage, and Transit (HIST)***. How to prepare the product for use?
5. ***Technical Maintenance (TM)***. How to keep the product working?
6. ***Technical Maintenance Schedules (TMS)***. What is done when?
7. ***Parts Lists (PL)***. What does the product consist of?
8. ***Modification Instructions (MI)***. How to change the product?
9. ***Disposal Instructions (DI)***. How to dispose of the product?

The main purposes of use of any technical manual are highlighted by these categories. Depending on the knowledge that they provide these categories can be classified into three abstract types: planning, support, and action. Table 3.2 outlines the main information categories supported by technical manuals classified by their purpose. The product purpose, performance, and technical maintenance schedules are planning information. Technical description and parts lists are support information. Operation, maintenance, handling, installation, storage, transit, and disposal instructions are action information. It is important to determine the association between the information of the supported product and the abstract purpose of information. Table 3.3 depicts the abstract purpose of usage categories in support of the product information. Moreover, it outlines the kind of product information that assists in product planning, support, and action.

The role of the information fragments is revealed through analysing their abstract functional characteristics. The type of product information supported by each function type is abstracted in Table 3.4. The table shows that the purpose of the product, general rules, maintenance plans, performance and capabilities data, etc. are

Table 3.2 Classification of Information Categories of Technical Manuals According to
their Purpose

Information Category	Purpose		
	Planning	Support	Action
Purpose and Planning (PP)	X		
Operating Instructions (OI)			X
Technical Description (TD)		X	
Handling, Installation, Storage, and Transit (HIST)			X
Technical Maintenance (TM)			X
Technical Maintenance Schedules (TMS)	X		
Parts Lists (PL)		X	
Modifications Instructions (MI)			X
Disposal Instructions (DI)			X

Table 3.3 Abstract Purpose of Product Information

Abstract Purpose	Product Information
Planning	<ul style="list-style-type: none"> • Purpose of the product • Performance data • Product suitability, capabilities and reliability for particular application or environment • Company's regulations, policies and standards. • General health and safety information • Maintenance strategy and schedules • Information about product or parts from manufacturer, supplier, vendor, etc
Support	<ul style="list-style-type: none"> • Technical concepts • Technical specifications • Product structure and parts lists (systems, assemblies, and parts)
Action	<ul style="list-style-type: none"> • Health and safety procedures • Starting, operating, and shutting down procedures • Maintenance procedures • Training and testing procedures • Fault diagnosis and correction procedures • Handling, installation, storage, and transit procedures • Modification procedures • Disposal procedures

Table 3.4 Functional Characteristics of Product Information

Function Type	Product Information
Fundamental (Fund)	<ul style="list-style-type: none"> • Purpose of the product • Performance data and product capabilities • Maintenance strategy and plan • General rules
Procedure (Proc)	<ul style="list-style-type: none"> • Sequential events regarding installation, operation, maintenance, diagnostic, testing, measuring, and hazards.
Specification (Spec)	<ul style="list-style-type: none"> • Product technical description • Information about systems, assemblies, and parts of the product
Advice (Adv)	<ul style="list-style-type: none"> • Specific requirements and recommendations • Warnings and cautions • Required test equipment and parts
Clarification (Cla)	<ul style="list-style-type: none"> • Assemblies and parts illustrations • Auxiliary and additional explanations
Organisation (Org)	<ul style="list-style-type: none"> • Types of maintenance • Product structure and parts lists

fundamental information. Sequential events with regards to installation, operation, maintenance, diagnostics, etc., are procedural information. Information about the product's systems, assemblies, parts and their technical descriptions are all specification information. Extra requirements and recommendations, warnings, and cautions are advice information. Illustrations of assemblies and parts and additional explanations are clarifying information. Finally, complex classifications and structures such as maintenance plans and product structure are organisational information.

3.2.3 Abstract Tasks and Activities in Technical Manuals

A large and diverse set of primitive tasks for technical manuals can be identified. This set might include tasks such as *introduce*, *plan*, *assess*, *launch*, *install*, *handle*, *check*, *assemble*, *test*, *replace*, etc. However, all of these tasks belong to two main abstract categories, namely “learn” and “perform”. For example, introduce, plan, and assess are tasks that assist the user in acquiring product-related knowledge, i.e. they have a learning objective. On the other hand, install, check, and assemble are tasks that enhance the user's skill in acting on the product, i.e. they have a performing objective.

People access technical manuals when they have an informational purpose in mind, and an associated task to accomplish. This goal is perceived as the activity of the user, which can be classified and categorised in advance during the technical manual's design. For instance, learning general information about the product is an activity which has a “planning” purpose, a category “PP”, and a primitive task “introduce”. Learning about a part of the product is an activity of purpose “support”, category

“PL”, and primitive task “assess”. Checking a fault in the product is an activity of purpose “action”, category “TM”, and primitive task “check”. Table 3.5 outlines the role of information categories and primitive tasks in supporting the abstract activities of technical manuals. It may be noticed that a single information category can be associated with more than one primitive task, e.g. OI is associated with “launch”, “handle”, and “manoeuvre”. In addition, a single task can be associated with more than one information category, e.g. “assess” is associated with PP, PL, and TD.

3.2.4 Semantic Database Schema for a Technical Manual

Figure 3.6 outlines an activity-based database schema for technical manuals. This schema is an instantiation of the generic schema shown in Figure 3.5. Entities are converted into physical database tables, and data categories are inserted. This knowledge base is used as a framework to organise and classify the topics in the technical manual. The figure also shows an example of a technical manual topic, which is a procedure that shows how to dismount the wheel of the forklift truck. This topic is described using three separate information elements, namely, steps of wheel dismount, unscrew nuts, and caution, which have function type procedure, clarification, and advice, respectively. Entities “topic” and “information element” will be discussed in detail in the next chapter, as they will be used to organise the data of the technical manual.

At this stage, it is suggested that the technical manual domain expert or designer should:

Table 3.5 Information Categories and Primitive Tasks in Support of the Abstract

Activities of Technical Manuals

Information Category	Primitive Task	Abstract Activity
PP	Introduce	Product Introduction
PP	Assess	Performance Assessment
PL	Assess	Parts Assessment
TD	Assess	Technical Data Assessment
OI	Launch	Launch Product
OI	Handle	Handle Product
OI	Manoeuvre	Product Manoeuvre
TM	Check	Check Product for Maintenance
TMS	Plan	Planning Maintenance Schedules

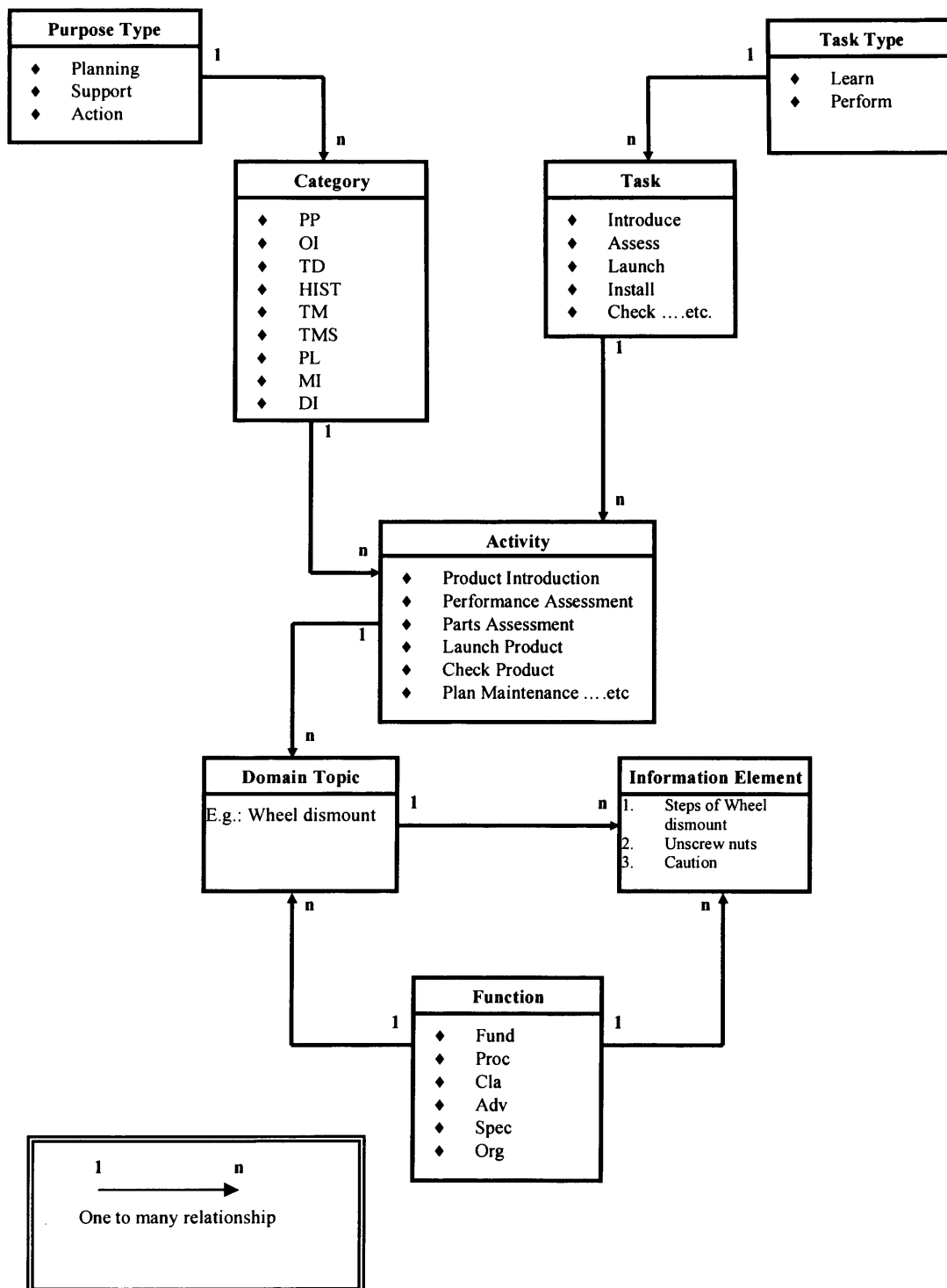


Figure 3.6 Activity-Based Database Schema for Technical Manuals

1. Identify the categories of information of the technical manual and associate each category with its usage purpose selected from the already existing abstract set. For example, “PP” is of type “Planning”, “PL” is “Support” and “TM” is “Action”.
2. Identify the primitive tasks supported by the technical manual and associate each task with a task type selected from the already existing abstract set. For example, “introduce” is of type “Learn” and “check” is of type “perform”.
3. Identify the abstract activities supported by the technical manual by associating information categories with primitive tasks. For example, “Product Introduction” belongs to the category “PP” and involves the task “introduce”, “Check Product” belongs to the category “TM” and involves the task “check”.
4. Identify the set of functions that will be used to describe the role of each information element within the technical manual. These can be chosen from the extended set outlined in Table 3.1 or, for simplicity, from the abstract types presented in Figure 3.6.

Figure 3.7 shows the schema editor organised into a set of data-entry forms. The schema editor was created on top of the database tables to facilitate appending and updating of the semantic categories. These forms facilitate the user interaction with the database tables. A MS-Access™ relational database management system [MS-Access, 1996] was used to implement this schema. The schema editor was developed using the MS-Access™ form generation wizard.

This approach enables the realisation of the purpose and the primitive task for every information fragment used in the documentation. Therefore, the purpose,

The figure displays five separate windows from a Schema Editor, each showing a record from a different database table. Each window has a title bar, a list of fields with their values, and a record navigation bar at the bottom.

- Information Purpose**: ID: 1, Name: Planning. Record: 1 of 3.
- Task Type**: ID: 1, Name: Learn. Record: 1 of 2.
- Information Category**: ID: 1, Name: PP, Description: Purpose and Performance, Type: 1. Record: 1 of 6.
- Task**: ID: 1, Name: Introduction, Type: 1. Record: 1 of 8.
- Activity**: Name: PP_Inf, ICat: 1, Task: 1, Description: Truck Purpose Introduction, ID: 1,1. Record: 1 of 9.
- Function**: ID: 1, Name: Fund, Description: Principle, Definition, Concept. Record: 1 of 6.

Figure 3.7 Schema Editor

category, task, task type, and function of every documented topic or even information element can be retrieved when needed. As it will be demonstrated in the next chapters, this feature facilitates and enables the automation of some of the technical manual development tasks. These include the automatic generation of hypermedia documents, and the automatic association of semantically relevant topics. It will also assist in the implementation of the adaptive information retrieval system and the diagnostic expert system.

3.3 SUMMARY

This chapter has presented a usage-based data model for the design of technical documentation. The proposed model abstracts the intended purpose of the documentation, the documentation-supported tasks, and the functional characteristics of documents. These abstractions have been combined in a usage-based semantic network that acts as a framework for structuring the documentation. In addition, semantic rules and valid relationships have been identified, and the semantic data model is mapped into a database schema. This mapping creates a knowledge base that preserves the data semantics, which can then be used by authors in order to organise, generate, and maintain the technical documentation. Furthermore, a case study has been conducted on a product-related technical manual in order to demonstrate the validity of this design approach.

As will be shown in subsequent chapters, the usage-based data model has been employed to support a structured authoring methodology for developing hypermedia-based technical documentation. In addition, this model will also be used to support the implementation of the adaptive hypermedia component of an intelligent PSS.

CHAPTER 4

MODEL-DRIVEN HYPERMEDIA AUTHORIZING APPROACH FOR WEB-BASED TECHNICAL DOCUMENTATION

Authoring is the art of collecting, structuring, and presenting information. This chapter presents a model-driven approach for Web-based hypermedia authoring. A brief conceptual review of the traditional hypermedia authoring process is presented. Then a methodology for authoring Web-based technical documentation is introduced, which utilises the usage-based data model described in Chapter 3. An approach for classifying information objects (IOs) and building the hypermedia structure to support the authoring methodology is outlined. In addition, a technique for generating identification codes for IOs and a tool for building the hypermedia structure are presented. These techniques are used automatically to generate the hypermedia pages in two different mark-up languages, HTML and XML. A technical manual for a fork-lift truck is used throughout this chapter to demonstrate the authoring methodology. The chapter also introduces a navigational model based on information semantics. It includes navigational access methods and an approach to generate two alternative types of navigational relationships. These are context-driven relationships and purpose-driven relationships. Then, a presentation technique using frame-based templates, icons, and colours is introduced. Finally, the system architecture that is used to demonstrate the authoring methodology and to integrate all these approaches, techniques, and tools, is illustrated.

4.1 HYPERMEDIA AUTHORIZING FOR THE WEB

4.1.1 Traditional Process to Web-Based Hypermedia Authoring

Figure 4.1 shows a conceptual model for the traditional Web-based hypermedia authoring process, outlining the main weaknesses of this authoring approach. This model exhibits an implementation language dependent authoring process that has tightly-coupled and inseparable structure, navigation, and presentation frameworks. It supports a single author environment and delivers applications of the “one size fits all” type i.e. published once and for all, which are then difficult to update. In addition, user-tailored presentations are difficult to implement, and the authoring process cannot support real-time user demands. Moreover, the model also shows that the authoring process, which is represented by a linear time line, is mainly an implementation-driven activity that is terminated after a certain time. The following is a brief description of some of the problems that hypermedia authors encounter when authoring for large data intensive Web-based applications using the traditional process:

- *Broken links* are hyperlinks with unavailable or not valid target documents. This problem is most likely to occur when an author deletes a document, and the hyperlinks in other documents pointing to that document are not updated.
- *Orphan documents* are documents with no incoming hyperlinks pointing to them, so they become unreachable. This occurs when the last link pointing to them is removed.

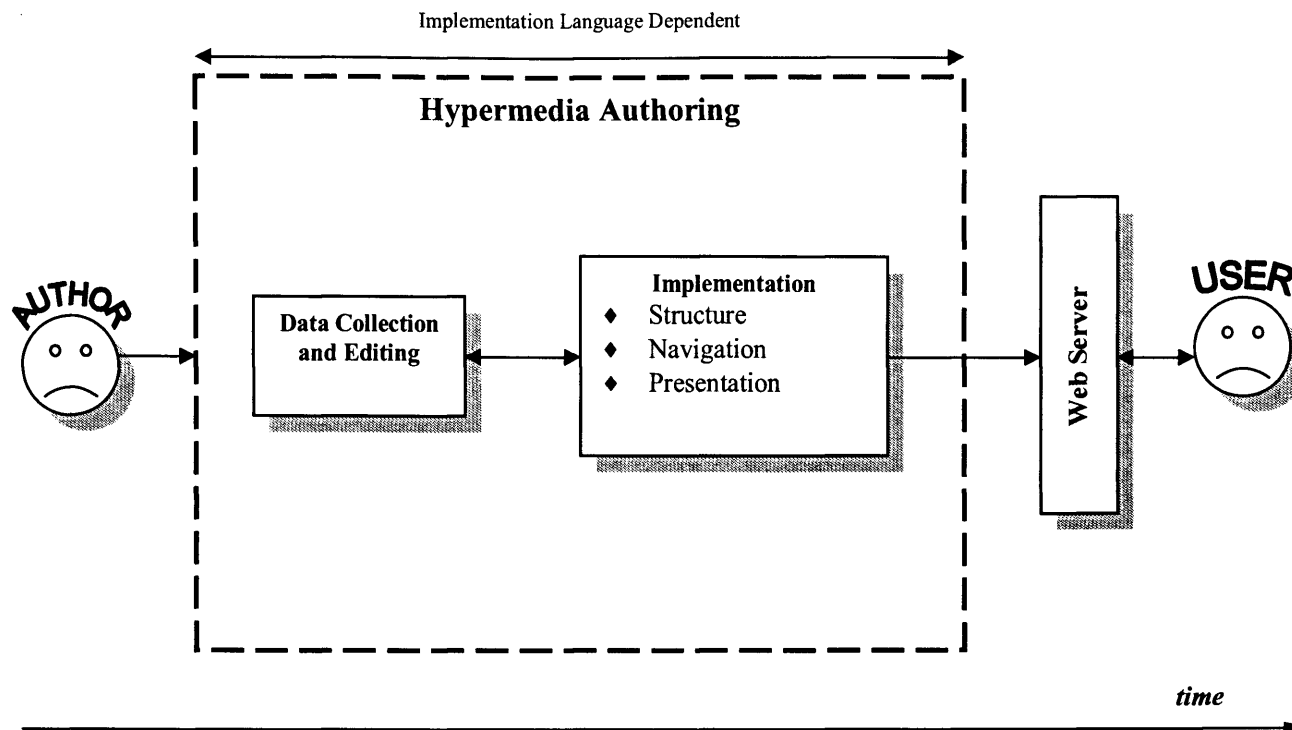


Figure 4.1 Conceptual Model - The Traditional Web-Based Hypermedia Authoring Process

- *Support for collaborative authoring* is difficult because hyperlinks are embedded within documents. Such hyperlinks are difficult to identify, especially when created and processed by more than one author.
- *Losing control of the available information.* Authors find it difficult to acquire an overview of the available information, to find or re-find the information they are looking for, and to update or append to existing information.
- *Customisation, personalisation, and adaptive user support services* are very difficult to accomplish because the structure, navigation, and presentation of information are tightly coupled and cannot be easily separated in order to be tailored to a particular type of user. In this case, the only way of providing user-tailored support is by, manually, creating a static set of pages for every user or group of users. This is a complex and time-consuming effort, which demands a huge amount of resources.
- *The final rendered application is statically tailored for a unique type of browser and/or display device.* This problem occurs because the structure, navigation, and presentation of the application are fully implementation language dependent rather than being declaratively and explicitly modelled.

This static and rigid approach to hypermedia authoring will eventually result in inconsistencies and chaos, and will create a significant complication regarding the maintenance of the application. In addition, since authoring is mostly conducted manually and with minimal support, the level of customisation and the resulting visual qualities of the final application are highly dependent on the effort spent by authors, which can be very high. In brief, this traditional approach to Web-based hypermedia

authoring may be efficient for small sized applications, but when the application grows overtime, problems and complications evolve at an alarming rate.

4.1.2 Model-Driven Approach to Web-Based Hypermedia Authoring

When organisations start moving their information onto the Web, authors need to consider new attitudes, concepts, and work strategies that coincide with the special requirements of the new publishing medium. Structured authoring approaches are used to tackle shortcomings of the traditional hypermedia authoring process. Figure 4.2 depicts a model-driven approach to Web-based hypermedia authoring, outlining the main features of this authoring approach. This approach is based on different interacting data models that are built on top of each other. These are the domain data model, and the hypermedia structure, navigation, and presentation models. These models represent the agreed framework for the domain and hypermedia, which are used and shared between authors and tools. They also provide formal guidelines for all authors to comply with, supported by applied constraints which insure that the resulting application, fully, complies with the requirements that these models represent. The separation of the hypermedia structure, navigation, and presentation models is a very important feature of this structured authoring approach. This separation provides authors with a high level of flexibility where, for instance, navigation hyperlinks are updated and edited without affecting the structure and vice versa. Franternali [1999] declares that the independent specification of presentation, separate from structure and navigation, is particularly relevant in the Web context where the final rendering on the interface depends on the browser and the display device. Thus it may be necessary to map the same design to different presentation

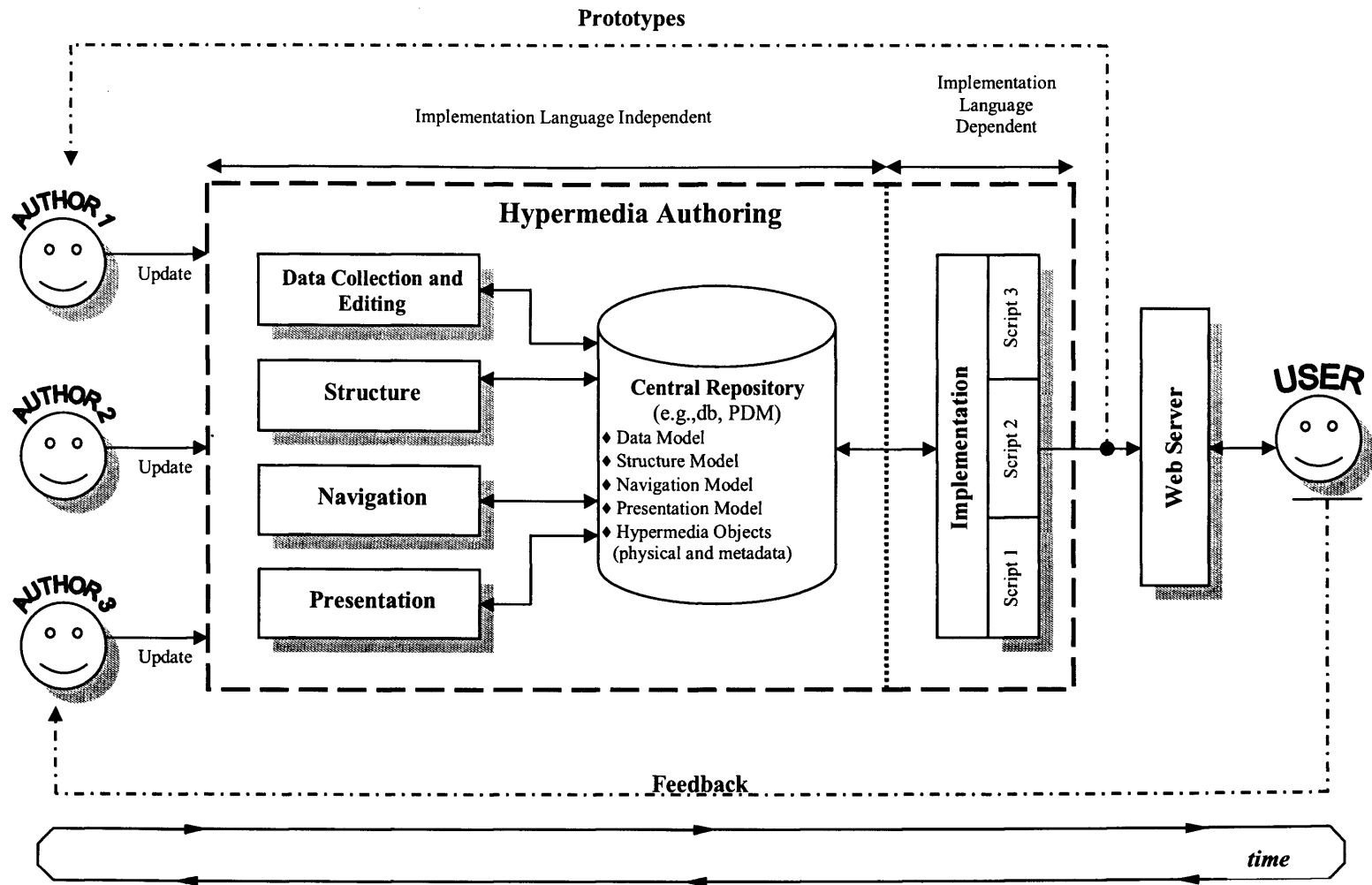


Figure 4.2 Model-Driven Approach to Web-Based Hypermedia Authoring

schemes associated with different user interfaces and/or display devices. In addition, the separation of these models allows the adaptive support of users' informational needs to be applied separately to each model, e.g.: adaptive navigation, adaptive presentation, etc.

This model-driven authoring approach involves a “central repository” that stores a representation of the domain data model, as well as abstract model-driven structural and navigational elements and presentation templates. These standard elements, together with a set of rules and constraints, are used to assist in the building of valid hypermedia structure, navigation, and presentation instantiations, and to provide consistency and efficiency to the delivered application. The central repository services an implementation language independent and multi-author environment. The clear separation of implementation issues from other authoring activities provides authors with an implementation language independent environment that enables them to render the same material in more than one Web-based mark-up language, e.g. HTML, XML, etc. In addition, this separation of activities enables the automatic generation of the application code using language-dependent rendering utilities, sparing the authors the burden of learning complex programming languages and providing for rapid prototypes and consistent applications.

In contrast with the static traditional authoring process that, directly or indirectly, terminates as soon as the material is published, this enhanced authoring process is most likely to continue after the application is deployed for use. For instance, tasks such as rapid and continuous information updates, information re-structuring, adaptive support and real-time user modelling, are executed while the system is deployed for

use. Moreover, the continuous authoring process provides the means for authors to dynamically process real user demands and feedback. The circular time line represents the continuation in the authoring process.

In brief, this structured approach to hypermedia authoring tackles and overcomes all the problems associated with the traditional approach that were mentioned earlier. Thus, there can be no “broken” links or “orphan” documents, the available information is fully controlled, stored, and contained, collaborative authoring is efficiently supported, and the requirements for the adaptive user support are considered at the authoring phase.

4.1.3 Application Example

Throughout this chapter, a technical manual for a forklift truck will be used in order to demonstrate the applicability and validity of the proposed model-driven approach for authoring Web-based technical documentation. This approach is demonstrated using example data, based on a prototype technical manual developed for a manufacturer of all-terrain forklift trucks as part of a project undertaken by the author’s laboratory within an EC-funded collaborative research program [Pham and Setchi 2003; Pham and Setchi, 2001, Setchi, 2000, and Pham et al., 1999]. This Web-based prototype is implemented in two different mark-up languages, HTML and XML. A set of CASE tools have been created in order to assist in the authoring and implementation processes. These tools have been developed under the MS-Windows NT™ operating system environment using the Java™ programming language, utilising the Java Development Kit JDK™ 1.3 [JDK, 1.3.0] and the MS-ACCESS 97™ [MS-Access,

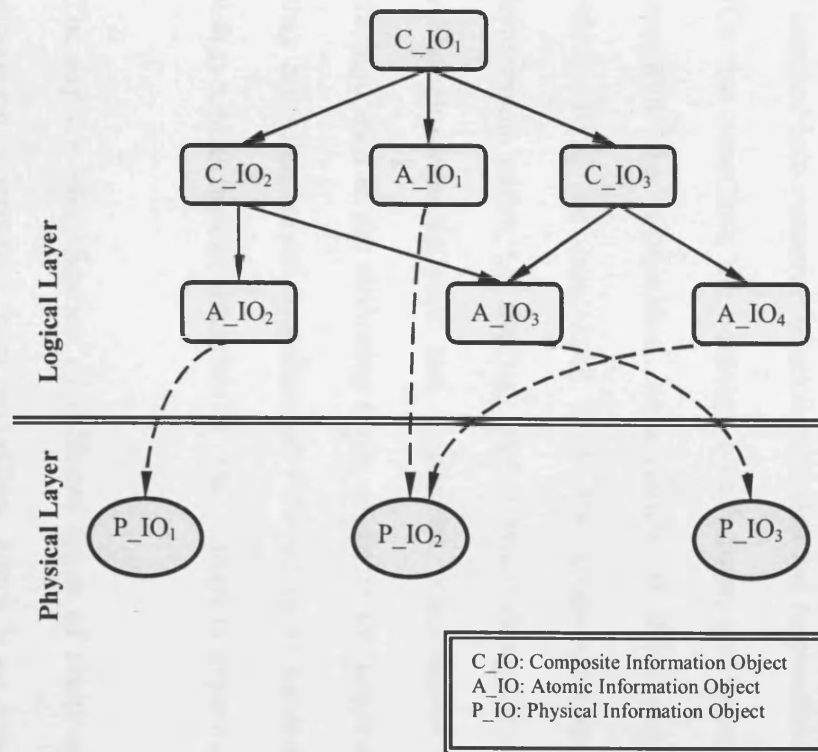
1996] relational database management system (RDBMS). In addition, the RDBMS was used as a central repository that is manipulated and accessed by the Java-based tools using embedded Structured Query Language (SQL) statements and a JDBC-ODBC Bridge.

4.2 EDITING AND MANAGEMENT OF MULTIMEDIA DATA ELEMENTS

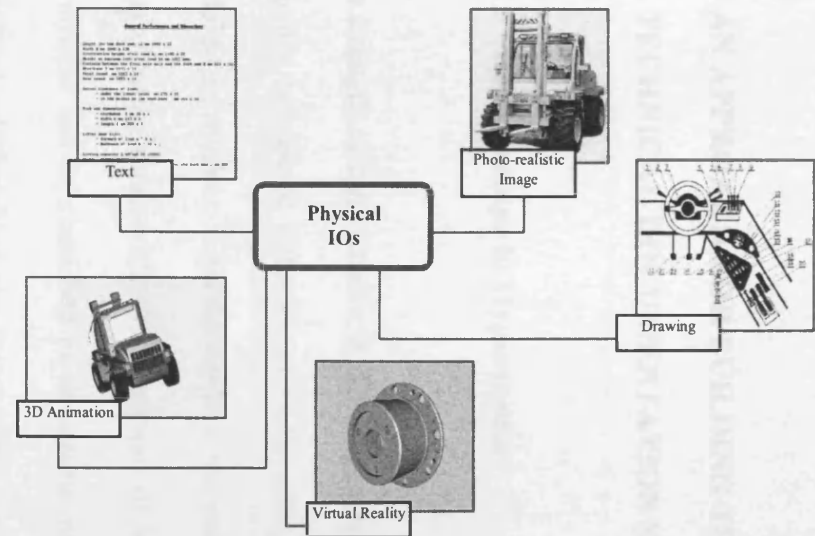
This phase of the authoring process is concerned with identifying sources of information and collecting, editing, and managing the multimedia data elements. In order to utilise these primitive and elementary multimedia elements, the author(s) should edit these elements and create publishable multimedia information objects (IOs), which are then stored in a repository. IOs are defined in [Tucker et al., 1997] as “a locution (set of words, phrases, sentences, etc.) of product documentation that describes one idea, concept, function, etc.”. IOs are the smallest units of publishable information that are created and modified by authors using widely available commercial tools, such as graphics packages, audio and video editors, animation software, etc. As described in [Langer et al., 1994], IOs are built by combining media objects along local or time dimensions, where a distinct media type is lost, e.g.: labelling graphics with text and including speech annotations. These IOs should be as small as possible, in order to ensure high flexibility, and at the same time they must be large enough to stand alone as part of a topic and to be reused in another, e.g. a step in a procedure, warning message, etc. As shown in Figure 4.3 (a), the IOs of the application domain span two layers, *physical* and *logical*. IOs are defined in the logical layer (metadata) while their actual content resides in the physical layer (data files). These two layers constitute the multimedia database, which is used to store and

organise the logical definitions and physical content of IOs. Within the logical layer, IOs are of two types, *atomic* and *composite*. The former are IOs that are associated with exactly one physical IO which holds its logical definition. The only way of accessing physical IOs is through atomic IOs. The latter are aggregations of atomic IOs, or other composite IOs, or a combination of both. The aggregation of atomic and composite IOs supports the creation of more complex and sophisticated hierarchical structures of domain documents/topics, where normally more than one IO contributes to the description of a single topic. The main concern of authors at this phase is preparing the physical layer by collecting and creating appropriate IOs with minimal redundancy, providing for a high-level of reusability.

In technical manuals, primitive multimedia elements are produced during the product design and production, and include 3D models, assembly trees, drawings, reports, virtual prototypes, assembly instruction sheets, bill of materials, etc. On the other hand, animations, video clips, annotated images, formatted text, and audio streams are specifically developed for documentation purposes. During the authoring process, the technical manual author(s) needs to access these IOs in order to prepare them for publishing. This includes activities aimed at classifying, updating, improving existing IOs, or creating new ones. The product and documentation data of the forklift truck include a heterogeneous collection of multimedia IOs, as shown in Figure 4.3(b). The data used in this work was collected and extracted from the structured data stored in Pro/INTRALINK™, a Product Data Management (PDM) system and in the Web-based product manual developed by Setchi [2000].



(a) Logical and Physical Information Objects



(b) Sample of the Collected Multimedia Information Objects for the Technical Manual of the Forklift Truck

Figure 4.3 Information Objects

4.3 AN APPROACH FOR BUILDING THE HYPERMEDIA-BASED TECHNICAL DOCUMENTATION STRUCTURE

4.3.1 Relationships in Hypermedia

The strength of hypermedia is demonstrated through its ability to provide users with logically connected network of documents, which is accessed and navigated in an exploratory manner. This network is the result of connecting data elements together by associating them with different types of relationships. Relationships in hypermedia documents can be classified as *structural* relationships and *referential* relationships. Structural relationships are responsible for holding the hierarchical structure of the hypermedia and the construction of the application's backbone. They are further classified into *resource* relationships that are responsible for the identification of the IOs that constitute the content of each page, and *organisational* relationships that constitute the hierarchical organisation of these pages. In contrast, referential relationships are concerned with the cross-reference linkage criteria between hypermedia pages, i.e. the way they interact with each other in the final application. Although organisational and referential relationships are clearly separated and distinguished at the authoring stage, as far as the application end user is concerned, they are often mixed together and referred to as *navigational* relationships. Figure 4.4(a) depicts the classification of relationships in hypermedia.

The explicit identification of different types of relationships enables the complete separation of structure from navigation, which is an essential authoring requirement

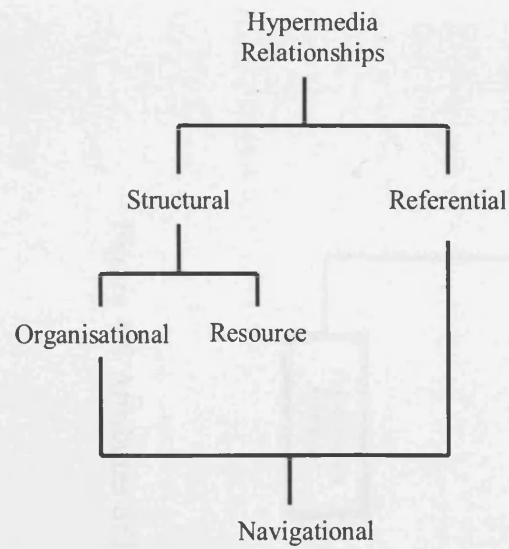
(see 4.1.2), and the automatic generation of these links based on their semantics.

Figure 4.4(b) shows different types of relationships in a hypermedia document.

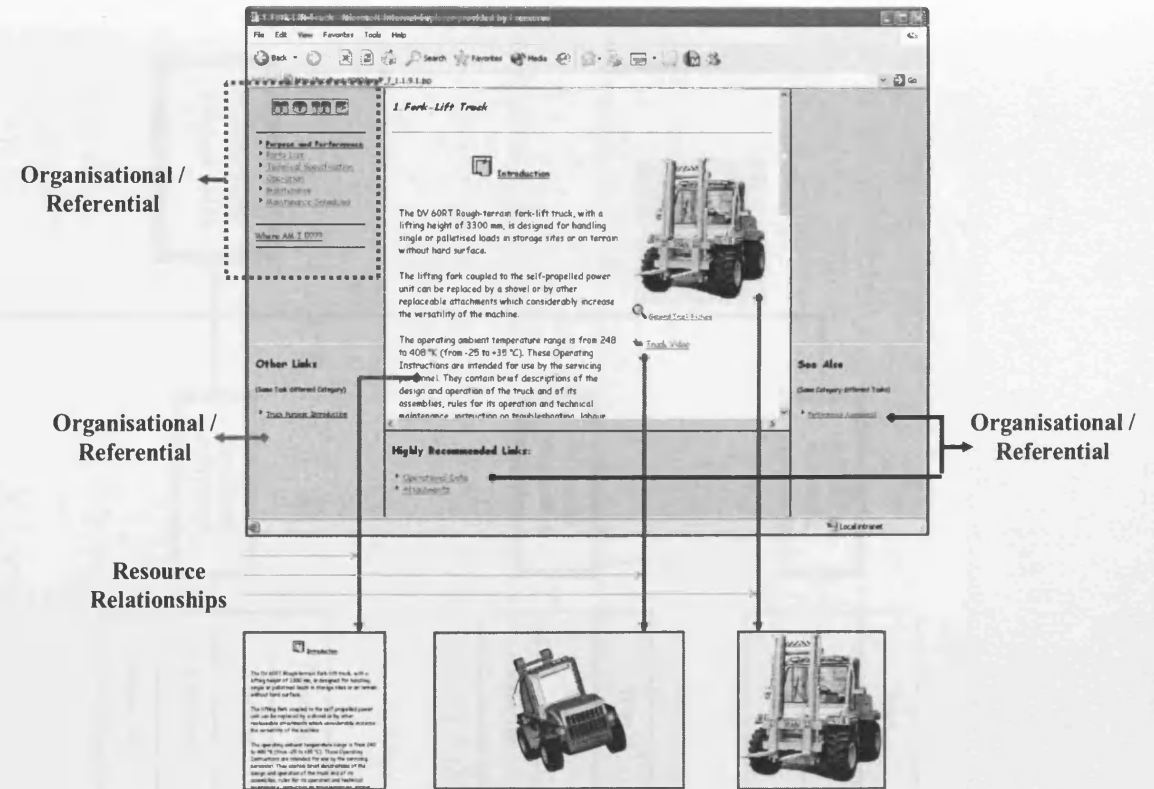
4.3.2 Attributes for Indexing the IOs of the Technical Documentation

Classification of IOs requires a great deal of understanding of their intended use and their semantic properties. A semantically rich usage-based data model for developing technical documentation and its associated rules has been suggested in Chapter 3. This model is used here to guide authors in the semantic description of IOs. Recall that information categories, tasks, activities, and functions were categorised and identified during the analysis and design stage of the technical documentation. In addition to other types of metadata i.e. attribute names and values, these abstract elements will be used to build a uniform and semantically valid hypermedia structure. Figure 4.5 presents the attributes used in the classification and indexing of the technical documentation IOs. These attributes are categorised into five distinct groups of categories:

- *Definition attributes*: These attributes provide a generic description of the IOs, and are used in conjunction with all other types of attributes. They include *name*, *description*, and *Identification Code (ID)*. The “name” and “ID” attributes are automatically set by the system, whereas “description” is set by the author.



(a) Classification of Relationships



(b) Different Types of Relationships in a Hypermedia Document

Figure 4.4 Relationships in Hypermedia

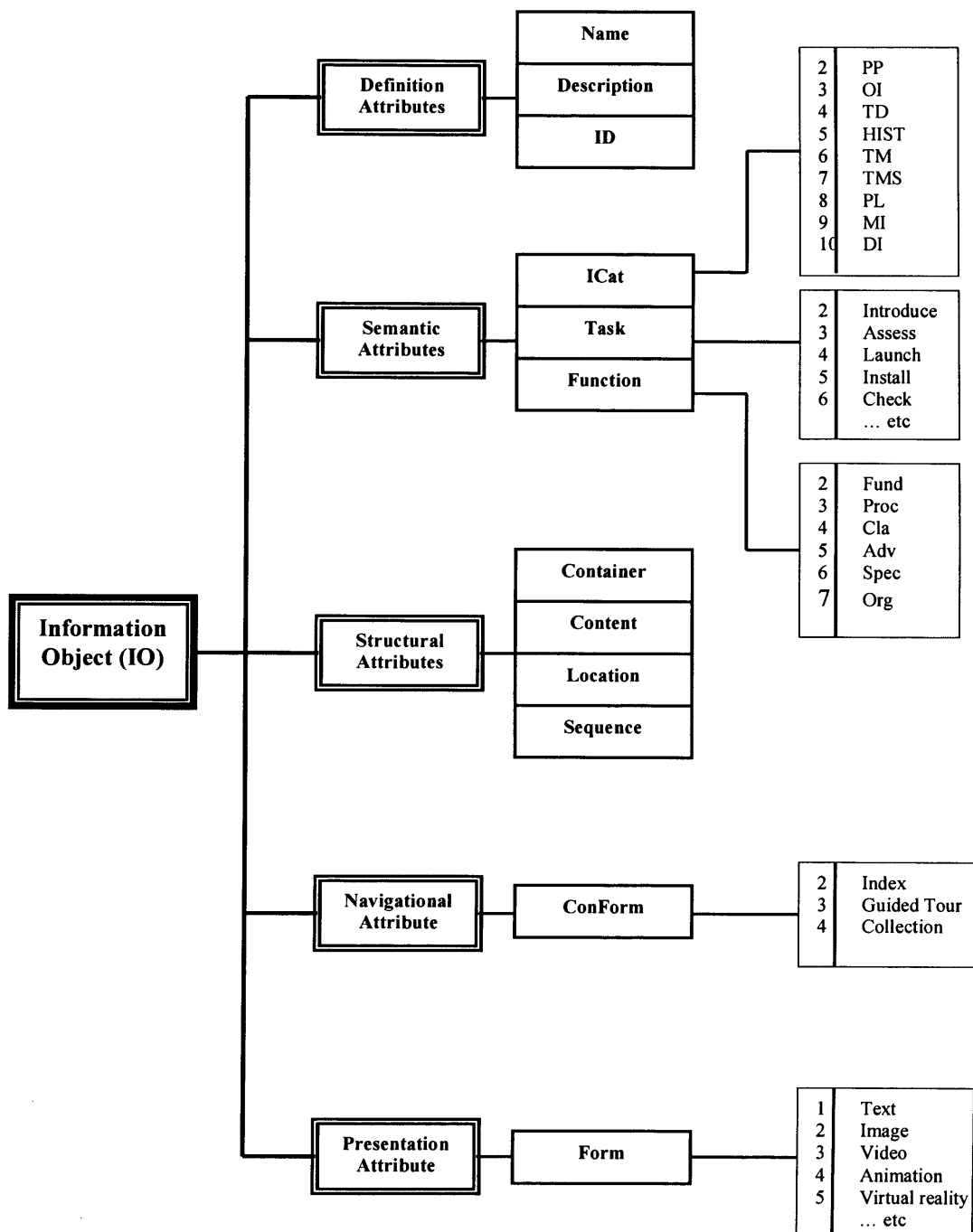


Figure 4.5 Attributes of Information Objects

- *Semantic attributes*: These attributes identify the semantic properties of every IO within the framework of the semantic data model and they are controlled by the associated rules. The attributes of this group include the *information category (ICat)*, *task*, and *function*. The values of these attributes are selected from pre-set known values that are identified by the domain expert during the analysis and design phase. The author identifies the semantic properties of the current IO by selecting a valid combination for the values of these attributes.
- *Structural attributes*: These attributes are concerned with building the hierarchical structure of the hypermedia application by identifying “organisational” and “resource” relationships. This group includes the *container* attribute which specifies the parent IO of the current IO, where the value of this attribute is chosen from a set of existing composite IOs. Containers are similar to the “folders” metaphor in the MS-Windows™ directory structure. The group also includes the *content* and *location* attributes, which specify a reference to the physical content of the current IO, and the *sequence* attribute which specifies the order of the current IO within its current container. The values of the attributes of this group are set by the author.
- *Navigational attribute - Container Form (ConForm)*: The value of this attribute identifies the access method for the current IO. The system supports three types of access methods, namely *index*, *guided tour*, and *collection*. The author specifies the appropriate access method for the current IO, e.g. maintenance procedures are best accessed via a guided tour, and parts lists are best accessed via an index.
- *Presentation attribute - Form*: This attribute identifies the media type of the current IO, which may include text, image, video, animation, virtual reality, etc.

4.3.3 Hypermedia Structure for the Technical Manual

In general, building a hypermedia structure requires the identification of the main topics supported by the documentation, which will be transformed into hypermedia pages. In addition, it requires the identification of the documents that constitute each topic and the organisational hierarchy (tree structure) of these topics and documents. Referential relationships between topics/documents can then be inserted on top of the structure to form the hypermedia network.

The author starts to build the hypermedia structure, after data is collected and edited, by associating related IOs together using “organisational” and “resource” relationships. The hypermedia structure is built by associating every IO with the appropriate values for each attribute (metadata), which were identified and categorised earlier. The tabular documents that identify the type of product information supporting different semantic elements, which were produced in the analysis and design stage (see Tables 3.3 and 3.4), provide good assistance for building the structure and associating every IO with a semantically valid attribute value.

British Standard 4884 [BS 4884, 1992] provides many guidelines for building the abstract structure of technical manuals. Two of the most common structuring methods in this field are the *system-based* and the *function-based* methods. The system-based method arranges the product into systems, sub-systems, assemblies, sub-assemblies, parts, etc. Similarly, the function-based method arranges the product into functions

and sub-functions, etc. Both are useful and valid structuring methods which consider the product from different perspectives, but for demonstration purposes, a semantically enhanced version of the system-based method is adopted in order to structure the technical manual's documents. This method relies heavily on the semantic data model, which is at this stage is preserved in a database ready to be used for the coding and classification of IOs (see Figures 3.3 and 3.6). To demonstrate this structuring method, a logical structure for a forklift truck technical manual is built. All data elements will be referred to and treated as IOs, which are either logical (atomic or composite) or physical as described in 4.2.

The author identifies the main systems that constitute the product, and that will be used in the technical manual. For instance, the systems of the forklift truck may include the braking system, lifting system, engine, electrical system, etc. These systems are coded as composite IOs with an organisational role of holding either the hierarchical structure of the system's assemblies and parts (parts list), the system's performance and specification data (performance data, technical description), or the sequential steps within a procedure or process applied to this system (technical maintenance). The hierarchy expands until terminated by one or more atomic IO, which points to an existing physical IO.

Figure 4.6 depicts the logical structure of the "braking system" within different semantic domains. These semantic domains represent the activities that support the

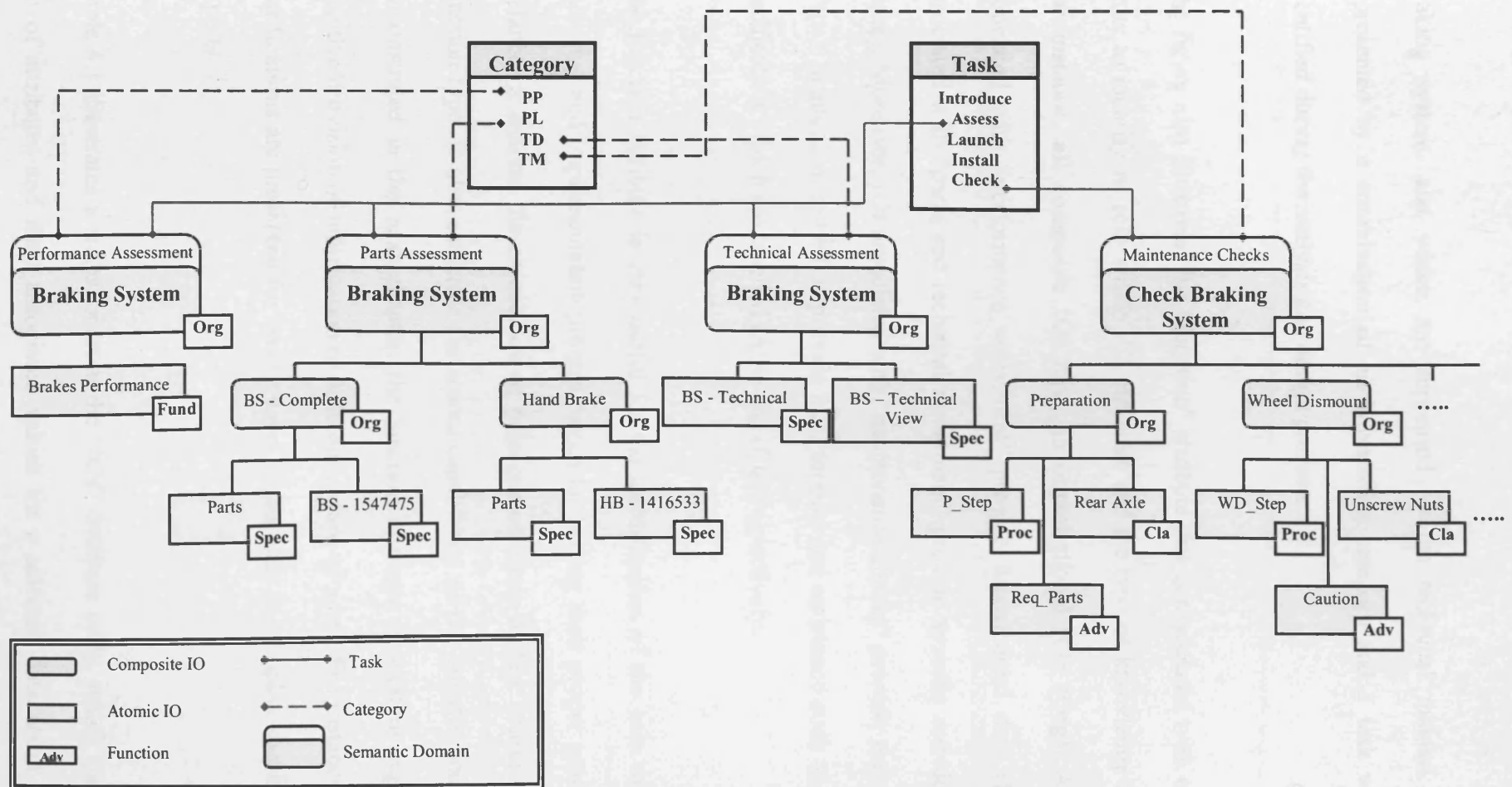


Figure 4.6 Braking System - Logical Structure with Semantic Domains

braking system, and which are supported by the technical manual. They are represented by a combination of an information category and a task which were identified during the analysis and design process.

The figure also illustrates the “function” attribute that is associated with every IO in order to identify its role within this structure and the type of knowledge it conveys. For instance, all composite IOs have an organisational role (Org). Atomic IOs associated with “performance assessment” provide fundamental data (Fund). IOs associated with “parts and technical assessment” provide specific and detailed data (Spec). Moreover, IOs associated with “maintenance checks” provide procedural data (Proc). In addition, some IOs provide local support and assistance such as advice and clarification, which are labelled (Adv) and (Cla) respectively.

The function attribute is very useful for the identification of the role of IOs in the authoring and implementation process, hence facilitating their proper processing. For instance, it enables the association of presentation icons and/or colours with every function type. It also enhances the search capabilities of the authors and, as will be demonstrated in the next chapter, the function attribute is vital in supporting the adaptive provision of information to different groups of users. For instance, additional clarifications are presented for novice users, in contrast to complex specifications for experts.

Table 4.1 illustrates a snapshot from the “IOs” database table, which shows the full set of attributes and their associated values for a selected data about the braking system of the forklift truck. In addition to the classification of attributes into

definition, semantic, structural, presentation, or navigation, attributes are also classified as system-set or author-set. The attributes *name* and *ID* are system-set i.e. their values are automatically set by the system. The remainder are author-set attributes, i.e. their values are manually set by the author. The fully processed set of IOs and their metadata is shown in Appendix A.

4.3.3.1 Dynamic Identification Codes for Maintaining the Structural and Semantic Properties of IOs

The Identification Code (ID) is an important definition attribute that holds the semantic and structural properties of every IO. It is an ordered pattern of digits that uniquely distinguishes and identifies IOs in a declarative manner. Theoretically, the number of levels in a dynamic structure hierarchy is not fixed and can grow to an infinite number of levels. Therefore, the ID of IOs must also be of a dynamic nature, reflecting the depth in the hierarchy and the set of predecessors for any IO. Internally, each attribute value of type “semantic” is associated with a unique identification number and these are stored in system database tables. Figure 4.7 presents these system tables and illustrates the mechanism of creating IDs for IOs. With respect to the logical structure, IOs are classified in two types, *Root*, and *non-Root* IOs. Root IOs are those residing on the top of the tree-like structure and do not have a parent, e.g.: the IOs that represent the “Braking System” in Figure 4.6; the remainder are

Table 4.1 Information Objects Metadata – Attributes and Values

Definition			Semantic			Structural		Presentation		Navigation	
Name	Description	ID	ICat	Task	Function	Container	Seq	Location	Content	Form	ConForm
f_1.3.9.2	Braking System	1.3.9.2	1	3	9	1.3	2	/ipm/	Empty	0	3
f_1.3.9.2.1.1	Brakes Performance	1.3.9.2.1.1	1	3	1	1.3.9.2	1	/ipm/	<PRE>W	1	0
f_2.3.9.2	Braking System	2.3.9.2	2	3	9	2.3	2	/ipm/	Empty	0	2
f_2.3.9.2.9.1	Braking System - Complete	2.3.9.2.9.1	2	3	9	2.3.9.2	1	/ipm/	Empty	0	3
f_2.3.9.2.9.1.5.1	Braking System 1547475	2.3.9.2.9.1.5.1	2	3	5	2.3.9.2.9.1	1	/ipm/pictures/P	Empty	2	0
f_2.3.9.2.9.1.5.2	Braking System Parts	2.3.9.2.9.1.5.2	2	3	5	2.3.9.2.9.1	2	/ipm/	<pre>No	1	0
f_2.3.9.2.9.2	Hand Brake	2.3.9.2.9.2	2	3	9	2.3.9.2	2	/ipm/	Empty	0	3
f_2.3.9.2.9.2.5.1	Hand Brake 1416533	2.3.9.2.9.2.5.1	2	3	5	2.3.9.2.9.2	1	/ipm/pictures/H	Empty	2	0
f_2.3.9.2.9.2.5.2	Hand Brake Parts	2.3.9.2.9.2.5.2	2	3	5	2.3.9.2.9.2	2	/ipm/	<pre>No	1	0
f_3.3.9.2	Braking System	3.3.9.2	3	3	9	3.3	2	/ipm/	Empty	0	3
f_3.3.9.2.5.1	Braking System - Technical Perspective	3.3.9.2.5.1	3	3	5	3.3.9.2	1	/ipm/	<PRE>Th	1	0
f_3.3.9.2.5.2	Braking System - Technical View	3.3.9.2.5.2	3	3	5	3.3.9.2	2	/ipm/pictures/T	Empty	2	0
f_5.9.9.1	Check Braking System	5.9.9.1	5	9	9	5.9	1	None	Empty	0	2
f_5.9.9.1.9.1	Preparation Process	5.9.9.1.9.1	5	9	9	5.9.9.1	1	None	Empty	0	3
f_5.9.9.1.9.1.2.1	Preparation Process	5.9.9.1.9.1.2.1	5	9	2	5.9.9.1.9.1	1	None	DESCRIP	1	0
f_5.9.9.1.9.1.3.2	Rear Axle	5.9.9.1.9.1.3.2	5	9	3	5.9.9.1.9.1	2	/ipm/pictures/s	Empty	2	0
f_5.9.9.1.9.1.4.3	Preparation Parts Requirements	5.9.9.1.9.1.4.3	5	9	4	5.9.9.1.9.1	3	None	<table bor	1	0
f_5.9.9.1.9.10	Reassemble of the Gear	5.9.9.1.9.10	5	9	9	5.9.9.1	10	None	Empty	0	3
f_5.9.9.1.9.10.2.1	Gear Reassembly	5.9.9.1.9.10.2.1	5	9	2	5.9.9.1.9.10	1	/ipm/	1 Put the	1	0
f_5.9.9.1.9.10.3.2	Gear Reassembly	5.9.9.1.9.10.3.2	5	9	3	5.9.9.1.9.10	2	/ipm/pictures/s	Empty	2	0
f_5.9.9.1.9.10.3.3	Gear reassemble on video	5.9.9.1.9.10.3.3	5	9	3	5.9.9.1.9.10	3	/ipm/videos/sta	Empty	3	0
f_5.9.9.1.9.11	Clearing Adjustment For The Hub Bearings	5.9.9.1.9.11	5	9	9	5.9.9.1	11	None	Empty	0	3
f_5.9.9.1.9.11.2.1	Clearing adjustment for the hub bearings	5.9.9.1.9.11.2.1	5	9	2	5.9.9.1.9.11	1	/ipm/	DESCRIP	1	0
f_5.9.9.1.9.11.3.2	Hub Bearings	5.9.9.1.9.11.3.2	5	9	3	5.9.9.1.9.11	2	/ipm/pictures/s	Empty	2	0
f_5.9.9.1.9.11.3.3	Clear and Adjust Hub Bearings	5.9.9.1.9.11.3.3	5	9	3	5.9.9.1.9.11	3	/ipm/videos/sta	Empty	3	0
f_5.9.9.1.9.12	Remounting of the Carrier	5.9.9.1.9.12	5	9	9	5.9.9.1	12	None	Empty	0	3
f_5.9.9.1.9.12.2.1	Carrier Remount	5.9.9.1.9.12.2.1	5	9	2	5.9.9.1.9.12	1	/ipm/	DESCRIP	1	0
f_5.9.9.1.9.12.3.2	Place the Carrier	5.9.9.1.9.12.3.2	5	9	3	5.9.9.1.9.12	2	/ipm/pictures/s	Empty	2	0
f_5.9.9.1.9.12.3.3	Replace screws, spring washers and bolts	5.9.9.1.9.12.3.3	5	9	3	5.9.9.1.9.12	3	/ipm/pictures/s	Empty	2	0
f_5.9.9.1.9.12.3.4	Place the Carrier	5.9.9.1.9.12.3.4	5	9	3	5.9.9.1.9.12	4	/ipm/videos/sta	Empty	3	0
f_5.9.9.1.9.12.3.5	Replace screws, spring washers and bolts	5.9.9.1.9.12.3.5	5	9	3	5.9.9.1.9.12	5	/ipm/videos/sta	Empty	3	0
f_5.9.9.1.9.13	Remounting of the Brake Drum	5.9.9.1.9.13	5	9	9	5.9.9.1	13	/ipm/	Empty	0	3
f_5.9.9.1.9.13.2.1	Remount the Brake Drum	5.9.9.1.9.13.2.1	5	9	2	5.9.9.1.9.13	1	/ipm/	DESCRIP	1	0
f_5.9.9.1.9.13.3.2	Replace the Brake Drum	5.9.9.1.9.13.3.2	5	9	3	5.9.9.1.9.13	2	/ipm/pictures/s	Empty	2	0
f_5.9.9.1.9.13.3.3	Replace the Brake Drum	5.9.9.1.9.13.3.3	5	9	3	5.9.9.1.9.13	3	/ipm/videos/sta	Empty	3	0

“non-Root” IOs. The ID of a “Root” IO is generated as the *dot* concatenation of its information category, task, function, and sequence.

$$ID\ of\ Root\ IO = ICat . Task . Function . Seq$$

$$E.g.: ID\ of\ IO_1 = 2.3.9.1$$

The ID of a “non-Root” IO is generated as the *dot* concatenation of parent ID, function, and sequence, which enables the inheritance of the semantic properties of information category and task within the same hierarchy.

$$ID\ of\ non-Root\ IO = ID\ of\ Parent\ IO . Function . Seq$$

$$E.g.: ID\ of\ IO_2 = \mathbf{2.3.9.1.9.1}$$

Embedding semantic and structural properties within the IDs of IOs simplifies the retrieval of these valuable properties for generating hypermedia pages during implementation and run-time. These IDs are also used in the generation of the navigational hyperlinks and the presentation templates.

The example shown in Figure 4.7 illustrates a hierarchical structure of five IOs. IO₁ has an ID value of (2.3.9.1) which when verified with the accompanied system tables provides the following knowledge about this IO:

- Information Category = 2 (Parts List)
- Task = 3 (Assessment)
- Function = 9 (Organisation)
- Sequence = 1 (First within the activity)

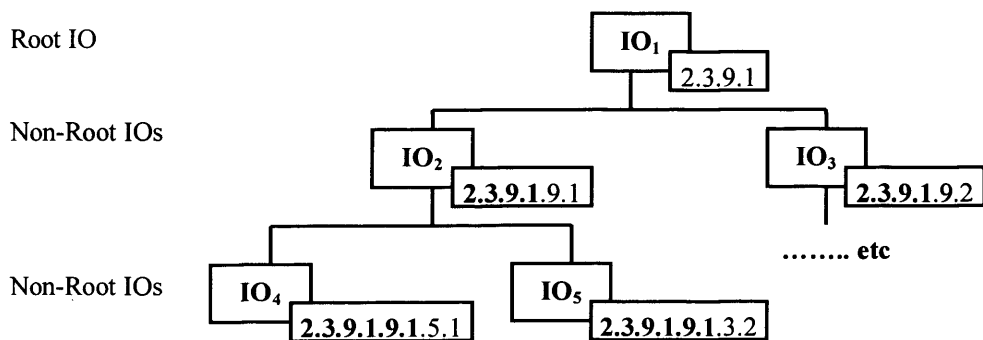
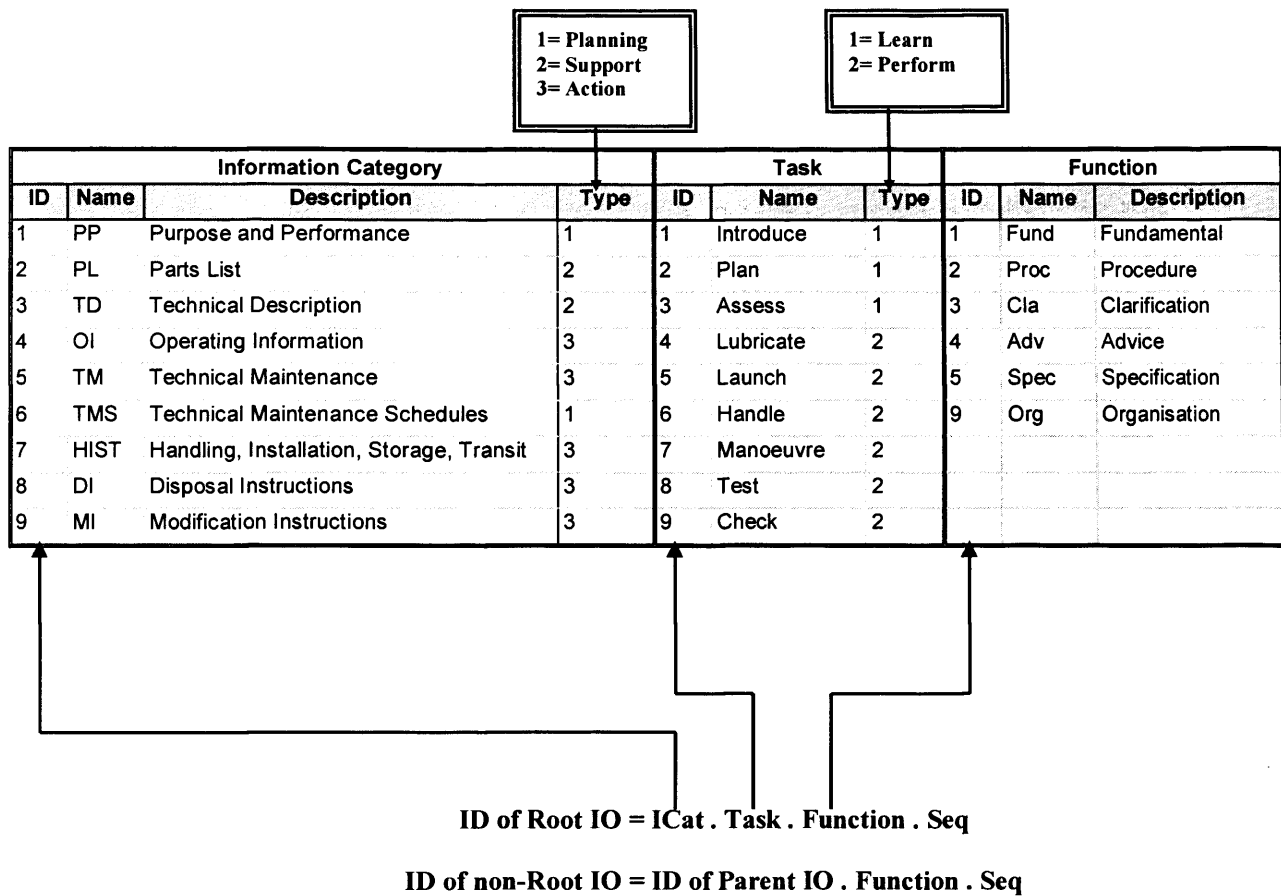


Figure 4.7 Identification Codes for Information Objects

Thus, it can be inferred that this IO concerns a product-related system, because it provides a first-level organisational role (and it is a composite IO). It is also related to the assessment of the parts of the product. Furthermore, this IO provides “support” type information and it is used with product-related “learning” tasks. IO₂, IO₃, IO₄, and IO₅ inherit the same information category and task from their parent IO₁. IO₂ and IO₃ are composite IOs that descend from a product system; hence they are either product assemblies or parts. IO₄, and IO₅ are atomic IOs performing a specification role (function 5) and clarification role (function 3), respectively. This implies that IO₄ is a specification of the product assembly/part represented by IO₂, and IO₅ is an additional clarification of this assembly/part.

4.3.3.2 Structure Builder – An Authoring Tool

Structure Builder is an authoring tool which has been created in order to assist authors in building a semantically valid and implementation language-independent structure. Figure 4.8 shows the main interface of the Structure Builder, depicting the possible values for coding an example IO. The example shown in the figure is concerned with coding a composite IO that is used in assessing the performance of the braking system of the forklift truck. The author inserts the description of the IO (Braking System) and then selects the appropriate information category from the list (PP). The tool then retrieves the set of tasks associated with this category from the “activity” table (Introduction and Assessment). The author selects “Assessment”, and then selects the function type i.e. role of the current IO, which is an organisational role (Org). In addition, the system also retrieves all existing composite IOs (Containers) within the selected activity (Performance Assessment), for the author to select the

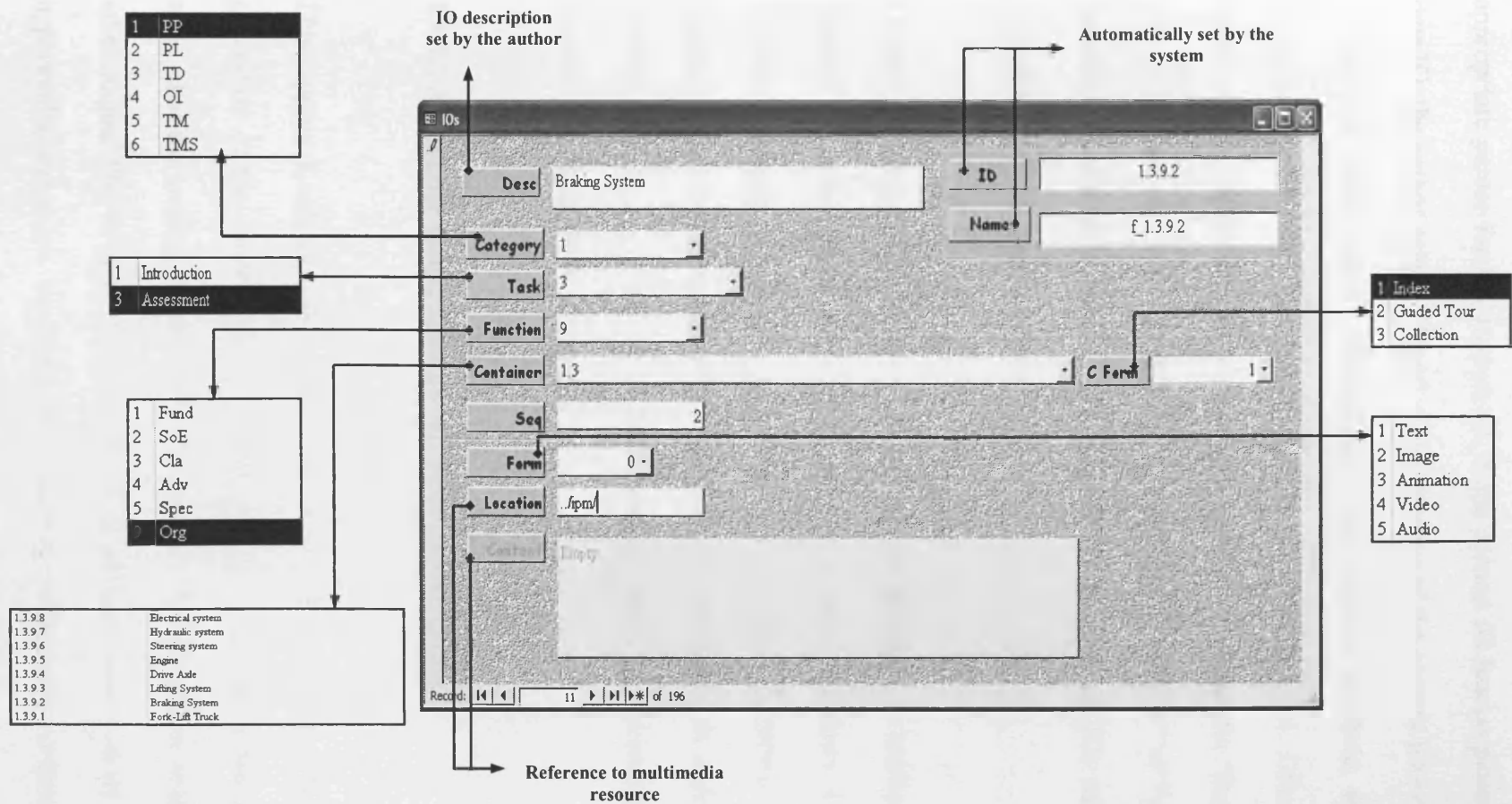


Figure 4.8 Structure Builder – An Authoring Tool

appropriate parent for the current IO. If the current IO has no parent, i.e. it is a root type IO, the author selects “Root”. The sequence of the current IO within the specified container is, then, entered. Depending on the function attribute, the author will be prompted to select the IO’s access method (Container Form). Otherwise, i.e. for an atomic IO, the author will be prompted to select the multimedia “form” of the current IO, and depending on this selection the appropriate “location” or “content” has to be specified. Extended examples of coding different types of IOs using the Structure Builder tool are shown in Appendix B.

The main strength of this tool is demonstrated through its ability to preserve and validate the semantic rules and relationships depicted in Figure 4.4. In addition, it provides an easy to use and author-friendly graphical interface, which hides the complexities associated with identifying IDs for IOs and with code generation. This tool supports the author through a set of lists, invoked upon demand, of logically and semantically valid attribute values.

4.3.3.3 Automatic Generation of the Hypermedia Pages

This approach of classifying and structuring IOs enables the automatic code generation of the Web-based hypermedia pages. It is highly recommended, at this stage, that the author should verify the application’s structure before proceeding to other stages. This is done by automatically generating one or more prototypes of the hypermedia structure, rendered in a chosen Web-based scripting language. To demonstrate that the built structure is implementation language independent, two rendering tools were developed using the Java™ programming language with

embedded SQL statements, which generate hypermedia pages in HTML and XML mark-up languages.

Figure 4.9 shows the automatically generated HTML pages of the “Check the braking system” procedure using the HTML rendering tool. These HTML pages have been generated by scanning the database tables using a breadth-first tree traversing technique, and translating the “resource” and “organisational” relationships into valid HTML pages and hyperlinks, respectively. It is important to realise that this structure is only a prototype that is generated to be tested and verified against the technical manual’s requirements, and is not used for publication. In addition, this structure constitutes the backbone of the final application, where navigation elements such as access methods and referential hyperlinks, and presentation elements such as templates, colours, and icons, are inserted, as will be shown later.

XML is a new emerging mark-up language whose strength is exhibited through its ability to generate self-describing data that explicitly emphasis the structure and semantics of the data separate from its presentation. Thus, XML can be used to easily interchange data among applications and/or organisations, to preserve data in machine-readable fashion, or to publish on the Web using separate style sheets for presentation [Harold, 1999]. An XML version of the technical manual is automatically generated using the XML rendering tool. Similar to the HTML version, the XML version has been generated by scanning the database tables. However, the technique used in this case is a depth-first tree traversing of the structured data. Figure 4.10 shows a sample from the automatically generated XML file and two different renderings resulting from applying Cascading Style Sheet (CSS) and eXtensible Style

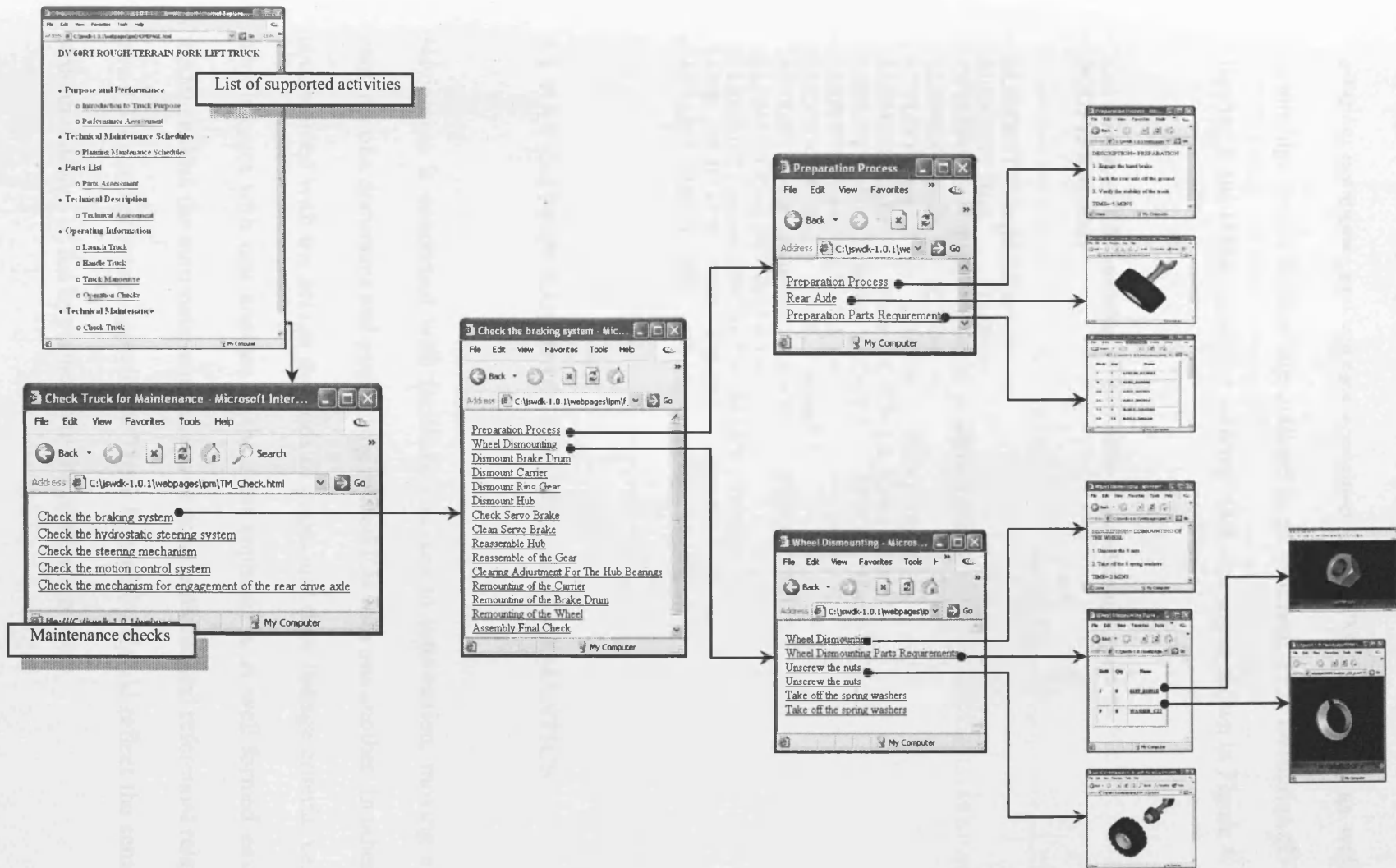


Figure 4.9 Structural Outline of Automatically Generated HTML Pages

Language (XSL) styling methods to the same XML file. The list of the elements, attributes, notations, and entities contained in an XML document, as well as their relationships to one another are outlined in a Document Type Definition (DTD). The following is the DTD associated with the XML document shown in Figure 4.10:

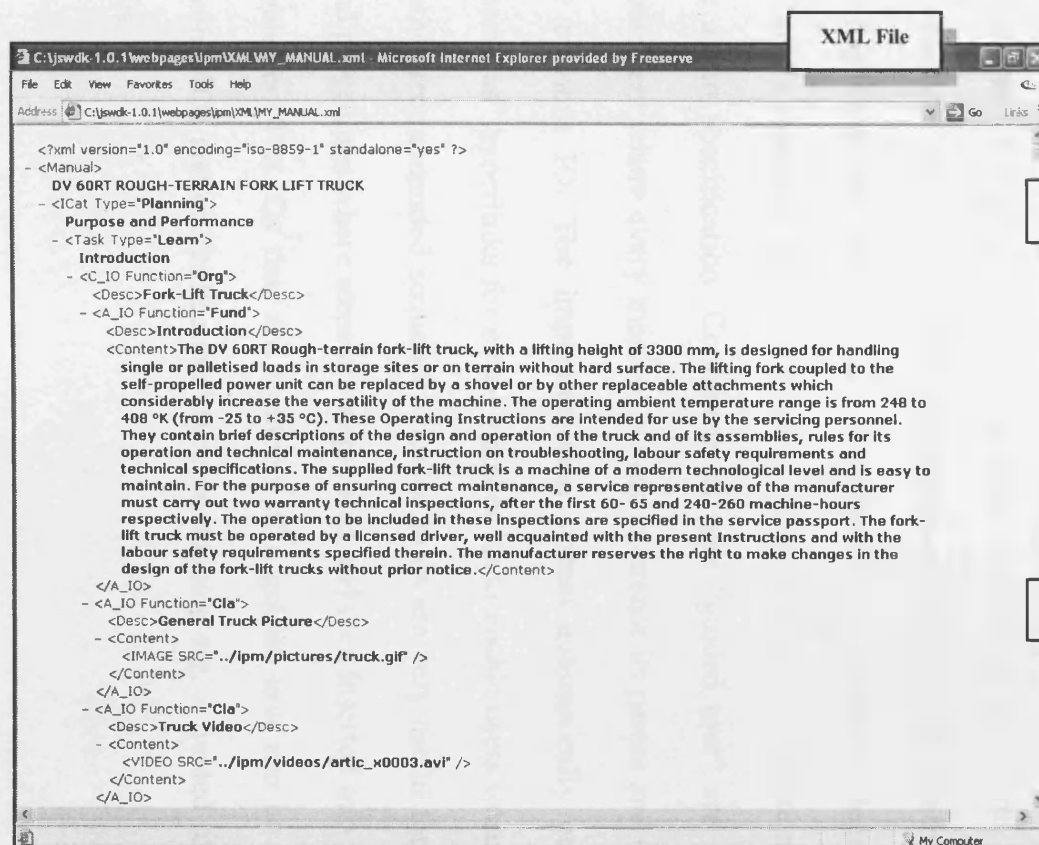
```
<?xml version = "1.0" encoding = "ISO-8859-1" standalone = "yes" ?>
<!DOCTYPE Manual
[

<!ELEMENT Manual (ICat+)>
<!ELEMENT ICat      (Task+)>
<!ATTLIST ICat  Type  CDATA      #REQUIRED  >
<!ELEMENT Task      (C_IO+)>
<!ATTLIST Task  Type  CDATA      #REQUIRED  >
<!ELEMENT C_IO      (Desc , (C_IO+ | A_IO+))>
<!ATTLIST C_IO  Function CDATA    #FIXED "Org"  >
<!ELEMENT Desc (#PCDATA)>
<!ELEMENT A_IO      (Desc , Content)>
<!ATTLIST A_IO  Function CDATA    #REQUIRED  >
<!ELEMENT Desc (#PCDATA)>
<!ELEMENT Contnet (#PCDATA| IMAGE | VIDEO)*>
<!ATTLIST IMAGE SRC  CDATA      #REQUIRED  >
<!ATTLIST VIDEO SRC  CDATA      #REQUIRED  >

]>
```

4.4 NAVIGATION BASED ON INFORMATION SEMANTICS

Navigation is concerned with facilitating access to information, moving across the contents of a document and associating related IOs with one another. In other terms, it is concerned with the access methods of pages and their linkage criteria, i.e. the way they interact with one another in the final application. A well-formed navigational model enables the automatic conversion of organisational and referential relationships (see 4.3.1) into physical hyperlinks. These hyperlinks should reflect the semantic and structural design of the hypermedia to the application users.



CSS Style Sheet

XSL Style Sheet

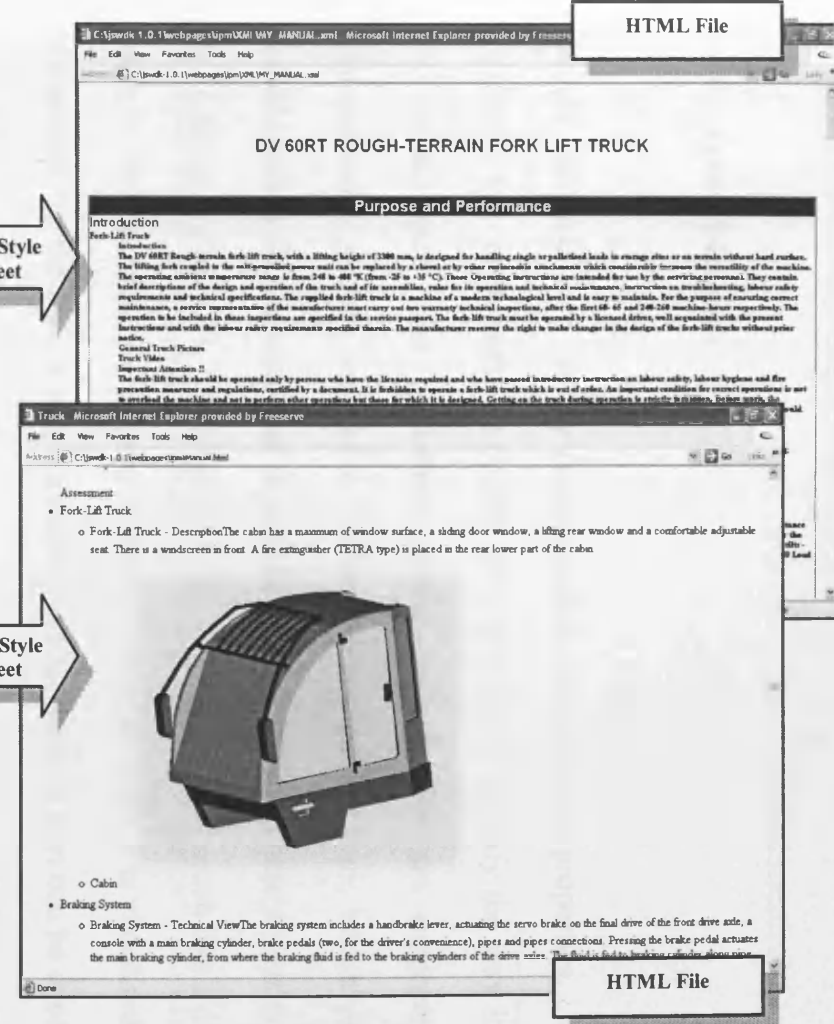


Figure 4.10 XML-Based Technical Manual Rendered Using CSS and XSL Style Sheets

4.4.1 Access Methods

Access methods are author-recommended navigational properties, which are associated with containers (composite IOs) in order to identify the way its member IOs are accessed. The predefined access methods that are supported by the implementation system are *index*, *collection*, and *guided tour*, which provide the possible values for the container form (ConForm) attribute. These access methods are illustrated in Figure 4.11. Containers of type “index” are very much like a directory or folder in a hierarchical file system. When accessing an index it is visualised as a table of contents, i.e. the user is given a choice of IOs; these in turn can be another index, collection, or guided tour. Index containers are useful in building hierarchical structures of related IOs, e.g.: product assemblies and parts. Containers of type “collection” are useful when combining and integrating relatively small pieces of atomic IOs to describe a larger composite IO. For instance, integrating different types of media elements, e.g. text, images, animations, etc., to create a step in a procedure or a part specification. Containers of type “guided tour” are sequentially-indexed containers where every member IO can reference its parent and the sequentially next or previous IO. The implementation system automatically generates “next” and “previous” hyperlinks for every member IO to enable users to access these IOs in an author recommended sequence. Guided tours are very useful for presenting tutorials and procedures where steps are sequenced and then inserted into a “guided tour” type container. The IOs that are members of a guided tour can be any combination of atomic IOs and/or “collection” type containers, e.g. a guided tour of collections of IOs.

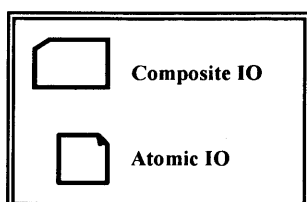
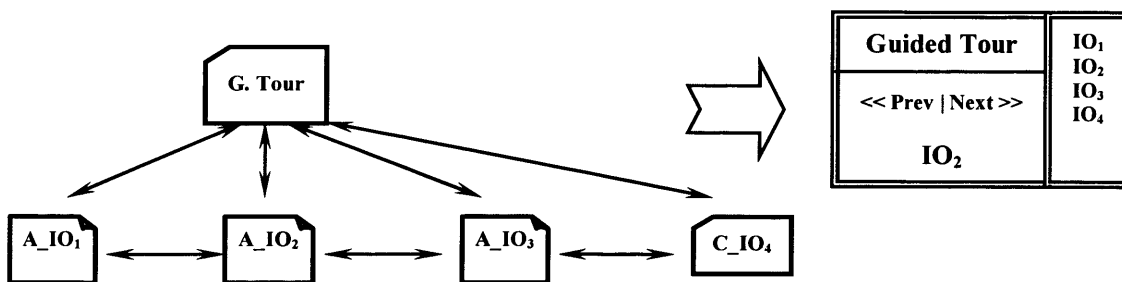
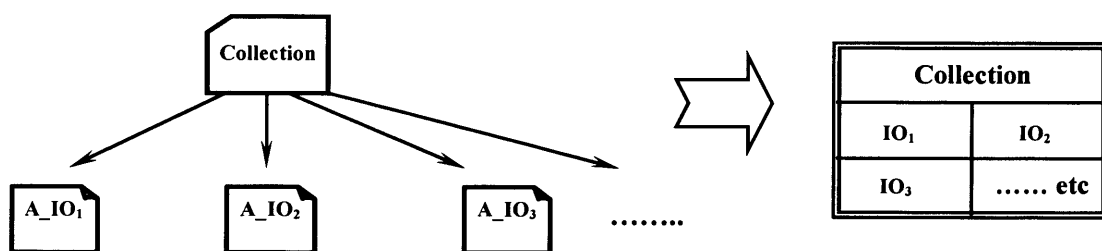
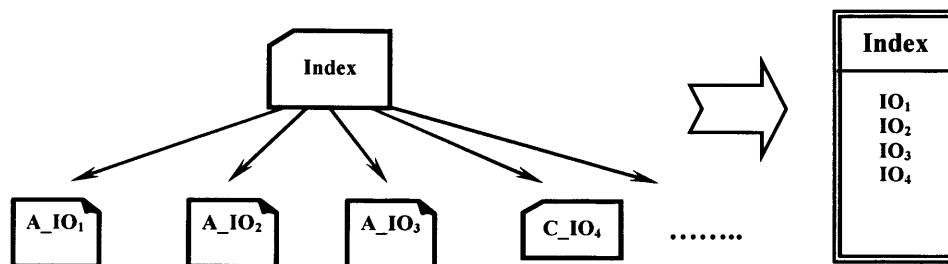


Figure 4.11 Access Methods

Access methods reduce the depth of the implemented hierarchical structure and shrink the number of structural levels by integrating them into collections and guided tours. Every IO can be a member of more than one index, collection, or guided tour. If not entered by the author, the default value for the access method (“ConForm” attribute) is “index”. A presentation template has been created for every access method, to support its individual requirements and enable its automatic code generation.

Among those structured IOs, the main entry point, which is the home page of the application, is a unique IO that is generated automatically by the implementation system and accessed via an index. This index constitutes a list of all the main high-level activities supported by the technical manual, which were acquired from the database representation of the semantic data model. The flexibility of the structuring mechanism described earlier allows different views (semantic domains) to be created on top of the same structure by switching between many groupings of abstract activities. As illustrated in Figure 4.12, the main activities supported by the technical manual can be categorised by their information category, purpose, supported tasks, or cognitive demand (abstract task types). Furthermore, the author can choose which activities to include and which to discard depending on the targeted end users and the type of the generated material, e.g.: operator manual, installation manual, planning and support material, etc. The final usage of the material, i.e. whether it is for publishing on the Web or for data exchange (XML format), is another important factor which determines the type of material to include.

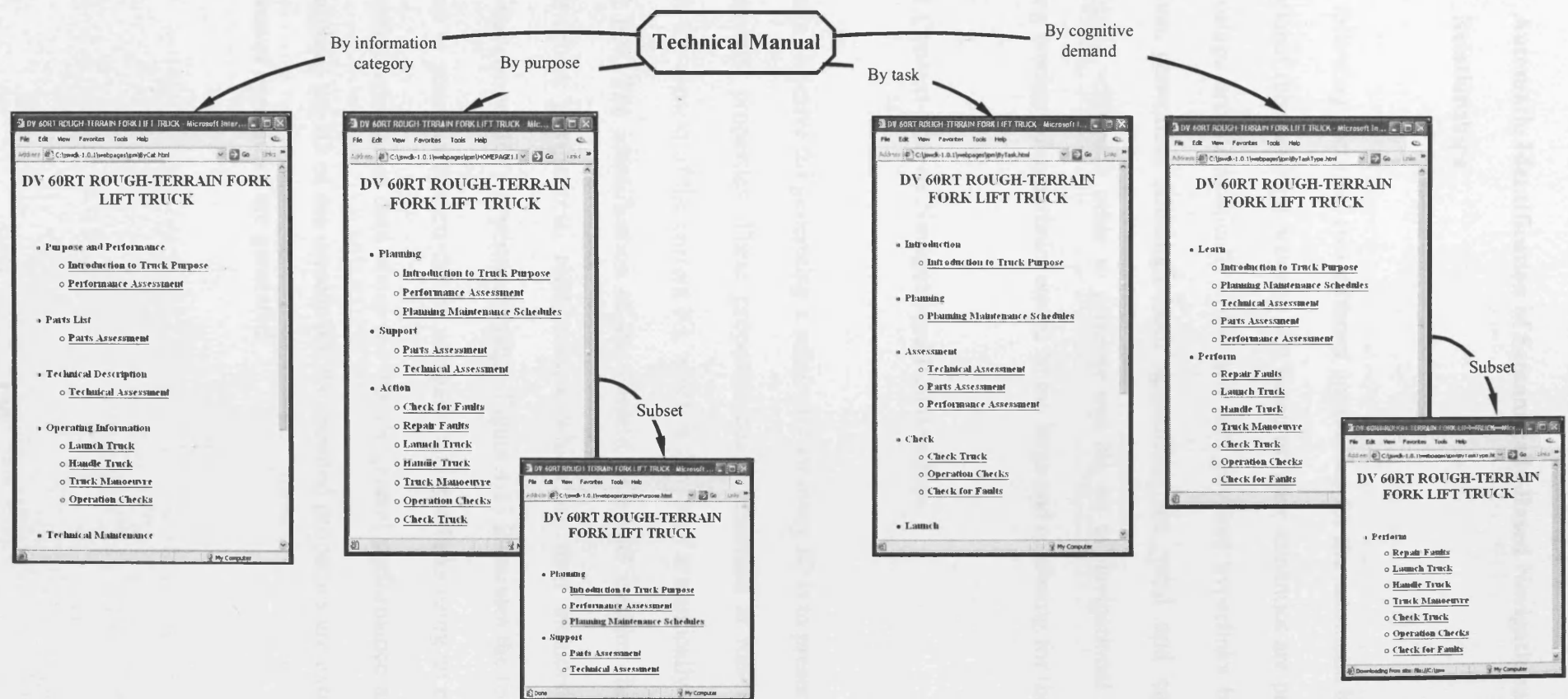


Figure 4.12 Semantic Domains - Alternative Views for the Technical Manual's Home Page

4.4.2 Automatic Identification of Semantically-Based Navigational Relationships

In the following sections, two different approaches for the automatic identification of navigational relationships between IOs based on their semantics are presented. These relationships are automatically converted into physical hyperlinks by the run-time hyperlink generator. Although both approaches are valid and can be used in conjunction with each other to generate one big set of navigational hyperlinks, the resulting number of hyperlinks would be too large and confusing for the user.

4.4.2.1 Context-Driven Navigational Relationships

The main objective for generating a unique ID for every IO is to preserve its structural and semantic properties. These properties can be accessed at run-time in order to identify the context of the current IO, which is the set of semantically and structurally related IOs. The identification of the context of an IO enables the generation of context-driven navigational relationships, which are then included as hyperlinks within the current IO's hypermedia page. Figure 4.13 illustrates the technique used at run-time to generate context-driven navigational hyperlinks using an example IO that represents fundamental data about the truck's general performance and dimensions. By analysing the ID of the current IO, the required properties are extracted, and five main lists of hyperlinks are generated:

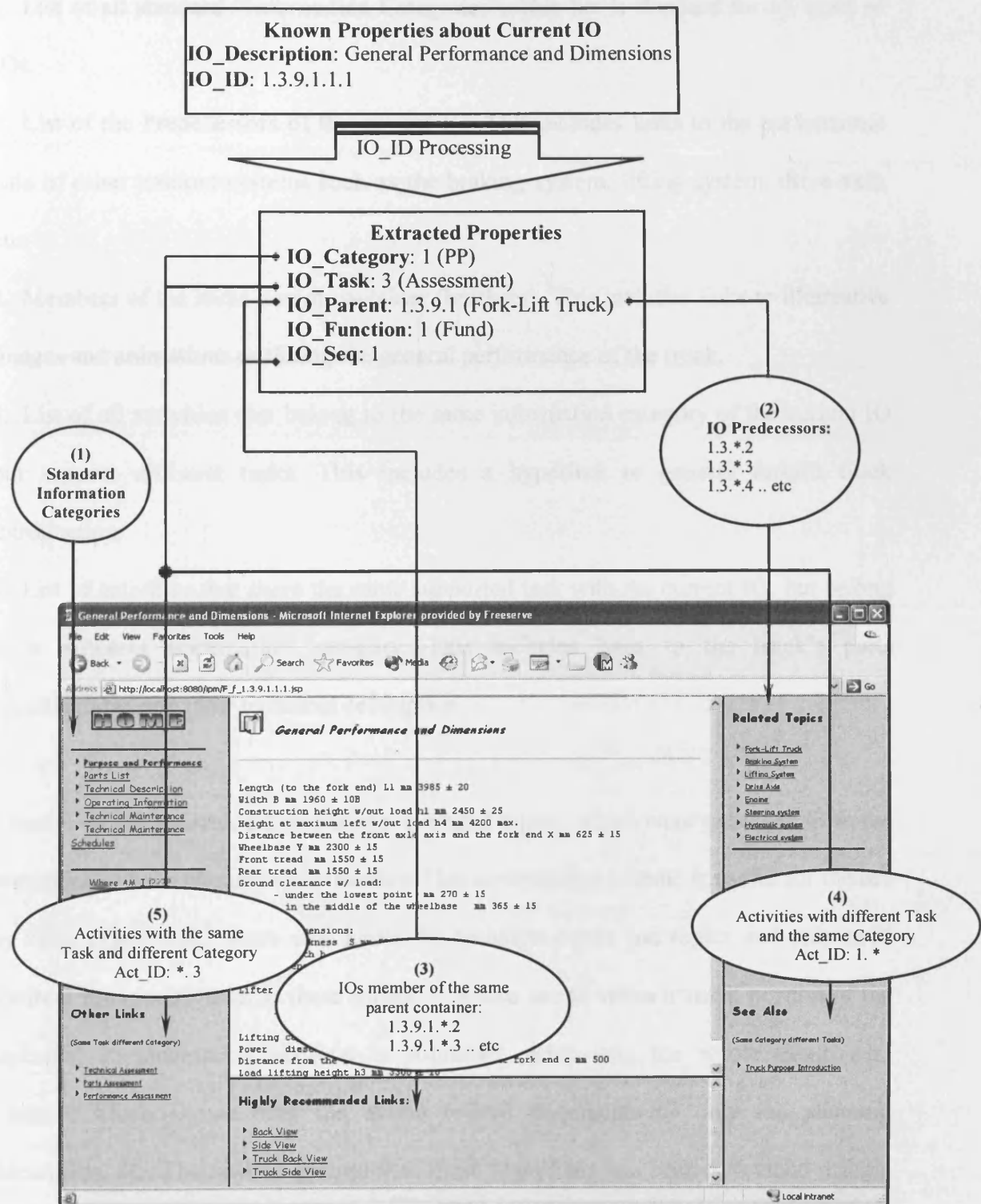


Figure 4.13 Generation of Context-Driven Navigational Hyperlinks

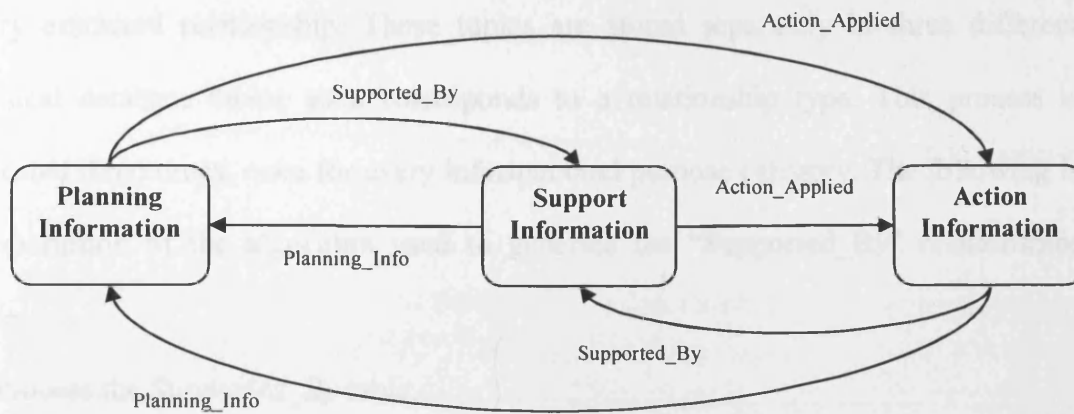
1. List of all standard “Information Categories”. This list is standard for all types of IOs.
2. List of the Predecessors of the current IO. This includes links to the performance data of other product systems such as the braking system, lifting system, drive axle, etc.
3. Members of the same parent container (brothers). This includes links to illustrative images and animations outlining the general performance of the truck.
4. List of all activities that belong to the same information category of the current IO but support different tasks. This includes a hyperlink to general forklift truck introduction.
5. List of activities that share the same supported task with the current IO, but belong to a different information category. This includes links to the truck’s parts specifications and their technical description.

Clearly, all the generated hyperlinks are related topics, which were extracted from the semantically built hierarchical structure. This navigational scheme is useful for the use by more experienced users who know the technical terms and topics and only need shortcut referential links to these topics. It is also useful when a small portion of the technical documentation material is published rather than the whole thing, e.g.: material which shows only the action related documents or only the planning documents, etc. The on-line generator of these hyperlinks has been developed using a JavaTM ServletTM with embedded SQL statements.

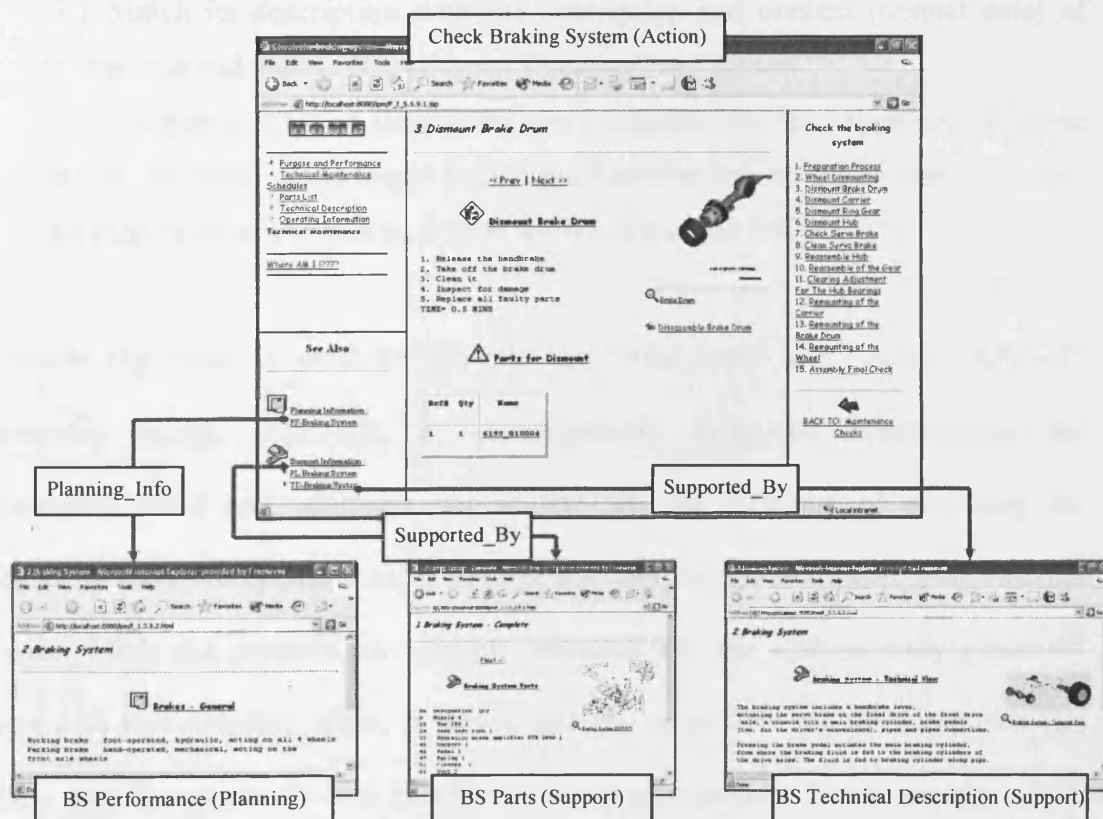
4.4.2.2 Purpose-Driven Navigational Relationships

This novel technique for automatically identifying navigational relationships and generating physical hyperlinks depends heavily on the semantic data model and in particular, the classification of information categories by their “purpose”. Every composite IO is associated with an information category that has a distinctive informational purpose, i.e. planning, support, or action. In most cases there exists a close complementary relationship between topics in these three separate but yet tightly coupled domains. This technique aims at identifying and extracting these relationships and converting them into physical hyperlinks. Figure 4.14(a) illustrates the relationships between planning, support and action information and the type of these relationships, namely, “Planning_Info”, “Supported_By”, and “Action_Applied”. For instance, action-related information about a certain product’s system “S1” is closely associated with the planning and support information about “S1” and vice versa. Figure 4.14(b) shows the automatically generated purpose-driven navigational hyperlinks for the “check the braking system” procedure, which semantically belongs to the “action” information domain. Ideally, a comprehensive presentation of this procedure should include hyperlinks pointing to fundamental information about the performance of the braking system (Planning_Info relationship), and to the specification of the braking system’s parts and its technical description (Supported_By relationship).

The technique used to generate these relationships depends on matching the textual description and content of IOs with one another, taking into consideration their purpose (planning, support, action) and their structural type (composite, atomic). The



(a) Relationships between Planning, Support, and Action Domains



(b) Automatically Generated Purpose-Driven Hyperlinks

Figure 4.14 Purpose-Driven Navigational Relationships

navigational links generator generates a two-tuple $\langle source, target \rangle$ representation for every extracted relationship. These tuples are stored separately in three different physical database tables each corresponds to a relationship type. This process is executed three times, once for every informational purpose category. The following is a description of the algorithm used to generate the “Supported_By” relationships table:

1. Process the *Supported_By* table.
 - 1.1 Delete the existing *Supported_By* table (if it exists).
 - 1.2 Create a new *Supported_By* table.
2. Extract the description of all composite IOs of type “Support”.
3. For every IO extracted in step 2, do the following:
 - 3.1 Match its description with the description and content (textual data) of composite and atomic IOs of type “Planning” or “Action”.
 - 3.2 If a match is found then insert a new relation into the *Supported_By* table where the value of the target field is the ID of the current IO of type “Support” and the value of the source field is the ID of the matched IO.

A similar algorithm is used to generate the “Plan_Info” and “Action_Applied” relationship tables. Although the automatically generated relationships are semantically valid and adequate, the author has the privilege of accessing the relationship tables to update them if needed, e.g.: adding a relationship to an external resource. Table 4.2 presents an example snapshot into the automatically generated “Supported_By” database table. It shows that the target IOs are all composite IOs (topics), and the source IOs can be either composite or atomic. A relationship, which associates a composite IO with another composite IO, will generate a referential hyperlink between the two corresponding hypermedia pages, e.g.: the highlighted relationship $\langle \text{Braking System, TD-Braking System} \rangle$ relates the

Table 4.2 Automatically Generated “Supported_By” Relationships

Source ID	Source Description	Target ID	Target Description
1.1.9.1	Fork-Lift Truck	3.3.9.1	TD-Fork-Lift Truck
1.1.9.1	Fork-Lift Truck	2.3.9.1	PL-Fork-Lift Truck
1.1.9.1.1.1	Introduction	2.3.9.1	PL-Fork-Lift Truck
1.1.9.1.1.1	Introduction	3.3.9.1	TD-Fork-Lift Truck
1.3.9.1	Fork-Lift Truck	3.3.9.1	TD-Fork-Lift Truck
1.3.9.1	Fork-Lift Truck	2.3.9.1	PL-Fork-Lift Truck
1.3.9.1.1.1	General Performance and Dimensions	2.3.9.4.9.3	PL-Wheel
1.3.9.1.1.1	General Performance and Dimensions	3.3.9.5	TD-Engine
1.3.9.1.1.1	General Performance and Dimensions	2.3.9.3.9.3	PL-Fork Arm
1.3.9.1.1.1	General Performance and Dimensions	2.3.9.5	PL-Engine
1.3.9.2	Braking System	3.3.9.2	TD-Braking System
1.3.9.2	Braking System	2.3.9.2	PL-Braking System
1.3.9.2.1.1	Brakes - General	2.3.9.4.9.3	PL-Wheel
1.3.9.3	Lifting System	2.3.9.3	PL-Lifting System
1.3.9.3	Lifting System	3.3.9.3	TD-Lifting System
1.3.9.4	Drive Axle	2.3.9.4	PL-Drive Axle
1.3.9.4	Drive Axle	3.3.9.4	TD-Drive Axle
1.3.9.5	Engine	2.3.9.5	PL-Engine
1.3.9.5	Engine	3.3.9.5	TD-Engine
1.3.9.5.1.1	Engine - General	2.3.9.5	PL-Engine
1.3.9.5.1.1	Engine - General	3.3.9.5	TD-Engine
1.3.9.6	Steering system	3.3.9.6	TD-Steering system
1.3.9.6	Steering system	2.3.9.6	PL-Steering system
1.3.9.7	Hydraulic system	2.3.9.7	PL-Hydraulic system
1.3.9.7	Hydraulic system	3.3.9.7	TD-Hydraulic system
1.3.9.8	Electrical system	2.3.9.8	PL-Electrical system
1.3.9.8	Electrical system	3.3.9.8	TD-Electrical system
1.3.9.8.1.1	Electrical system - General	2.3.9.8	PL-Electrical system
1.3.9.8.1.1	Electrical system - General	3.3.9.8	TD-Electrical system
4.5.2.2	Running the New Truck	3.3.9.1	TD-Fork-Lift Truck
4.5.2.2	Running the New Truck	2.3.9.5	PL-Engine
4.5.2.2	Running the New Truck	3.3.9.4	TD-Drive Axle
4.5.2.2	Running the New Truck	2.3.9.7	PL-Hydraulic system
4.5.2.2	Running the New Truck	2.3.9.1	PL-Fork-Lift Truck
4.7.2.5	Picking Loads	2.3.9.3.9.3	PL-Fork Arm
4.7.2.5	Picking Loads	2.3.9.2.9.2	PL-Hand Brake
.... etc

performance data of the braking system with its technical description. However, a relationship which associates an atomic IO with a composite IO corresponds to the automatic identification of the “hot” keywords that exist in the source IO’s textual content and will generate the appropriate referential hyperlink to a more detailed elaboration of these keywords. For example, the text describing the IO titled “Picking Loads” contains references to the IOs titled “Fork Arm” and “Hand Brake”. These textual references are identified and the corresponding relationships are generated, which are converted into referential hyperlinks as depicted in Figure 4.15.

The on-line generator of these hyperlinks was developed using a Java™ Servlet™ with embedded SQL statements. The full set of the automatically-generated purpose-driven navigational relationships, “Action_Applied”, “Plan_Info”, and “Supported_By” are depicted in Tables C.1, C.2, and C.3 of Appendix C, respectively.

4.5 FRAME-BASED PRESENTATION TEMPLATES

The presentation templates used in the user interface are based on HTML frames, which divide the interface window into a number of navigation-able areas. Every composite IO is associated with a presentation template which is based on its access method. Thus, there are three types of templates namely index, collection, and guided tour. Figure 4.16 illustrates the physical organisation of different types of presentation templates based on the access method. The “Categories”, “Content”, “See Also”, and “Where Am I?” frames are standard in all templates. Other frames are exclusive to a specific access method, e.g.: list of IOs that are members of a guided tour. The

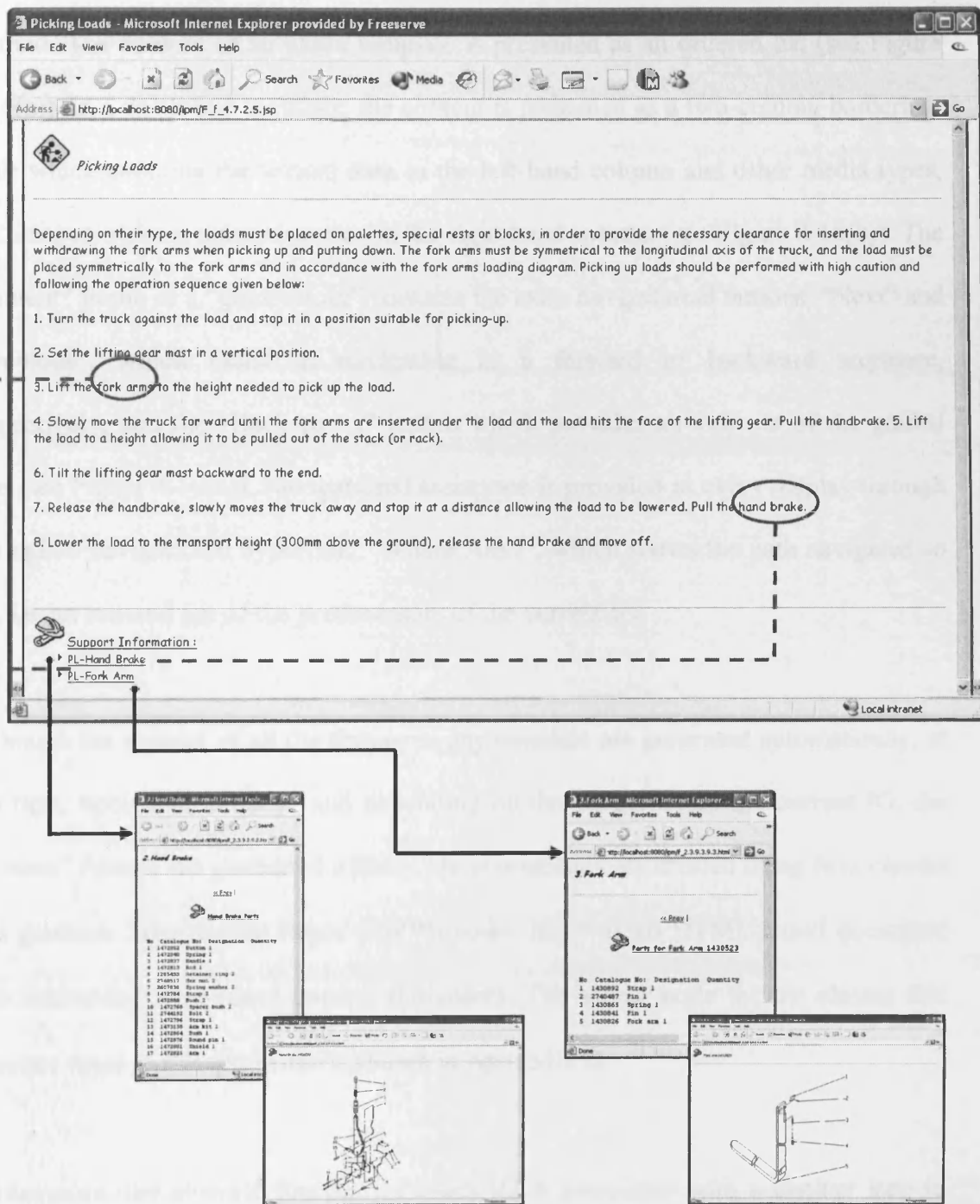


Figure 4.15 Keyword-Based Referential Hyperlinks

presentation of the IOs inside the “content” frame depends, again, on the access method. The content of an index template is presented as an ordered list (see Figure 4.16(a)). In a collection template, the content is presented as a two-column borderless table which contains the textual data in the left-hand column and other media types, e.g. images, videos, animation etc., in the right-hand column (see Figure 4.16(b)). The “content” frame of a “guided tour” contains the extra navigational buttons, “Next” and “Previous”, which assist in navigating in a forward or backward sequence, respectively, and the “Back to ...” button which provides an exit out of the guided tour (see Figure 4.16(c)). Navigational assistance is provided in every display through a standard navigational hyperlink, “Where Am I”, which shows the path navigated so far, i.e. an ordered list of the predecessors of the current IO.

Although the content of all the frames in any template are generated automatically, at run time, upon user request, and depending on the properties of the current IO, the “content” frames are generated offline. These templates are created using Java classes that generate Java Server Pages (JSP™) code. JSP™ is an HTML-based document with embedded Java-based control statements. The source code for the classes that generate these JSP™ templates is shown in Appendix D.

Furthermore, the abstract function of every IO is associated with a distinct icon to represent its role in a visual manner. Table 4.3 outlines the abstract functions of IOs and their associated visual icons. The use of some of these icons is shown in the user interface windows in Figure 4.16. Different background colours can be easily associated with different classifications of categories, purposes, tasks, or task types (cognitive demand). Every classification can be associated with a distinctive

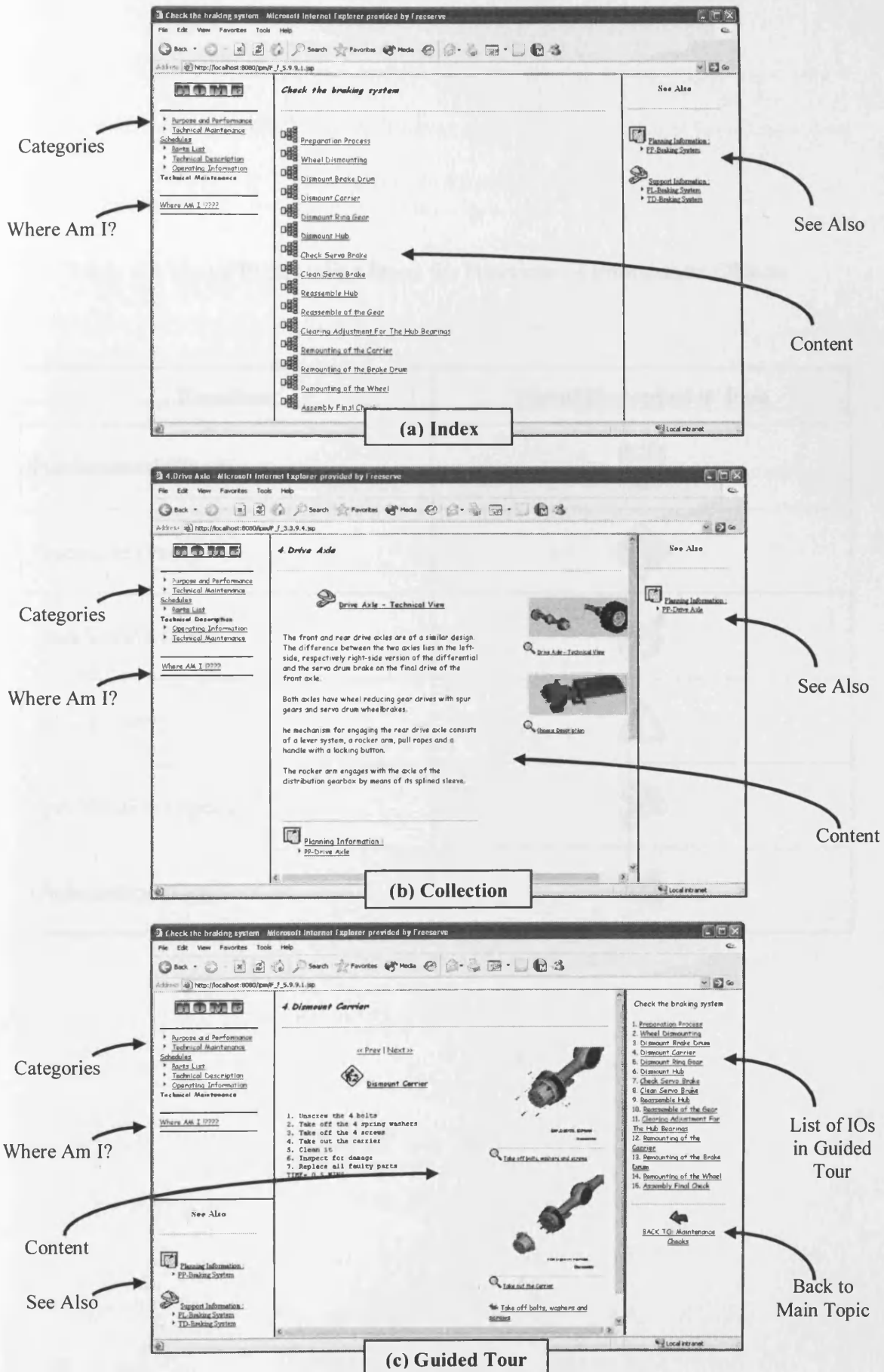








Figure 4.16 Physical Organisation of Presentation Templates of Access Methods

Table 4.3 Visual Presentation Icons for Functions of Information Objects

Function	Visual Presentation Icon
Fundamental (Fund)	
Procedure (Proc)	
Clarification (Cla)	
Advice (Adv)	
Specification (Spec)	
Organisation (Org)	

background colour to enable users to determine the type of the delivered information in a visual manner. For example, “blue” background can be associated with planning information, “yellow” with action information, and so forth.

4.6 SYSTEM ARCHITECTURE

The system architecture shown in Figure 4.17 has been created to assist in the development i.e. author and implement, of the technical manual using the model-driven authoring process. It distinguishes two types of environments namely, *Off-Line Environment* and *On-Line Environment* (run-time).

The off-line environment comprises specially developed tools for authoring and implementing the technical documentation. These tools sit on top of a database containing the technical documentation metadata. All access to the database is facilitated through a JDBC/ODBC bridge. This database is updated by the author(s) using the *Structure Builder* and processed by the *Navigational Relationships Generator*. The domain’s physical data files are stored in a file system that is edited using different types of commercially available multimedia editors.

The main implementation tool is the *Web Pages Generator*, which processes the design and metadata database of the technical documentation, and automatically generates HTML and XML documents using the *HTML Render* and *XML Render* tools respectively. The HTML render tool uses the *Presentation Templates Generator* in order to generate JSP templates which embed Java-based control statements that are invoked at run-time (on-line). In addition, the HTML render tool renders the

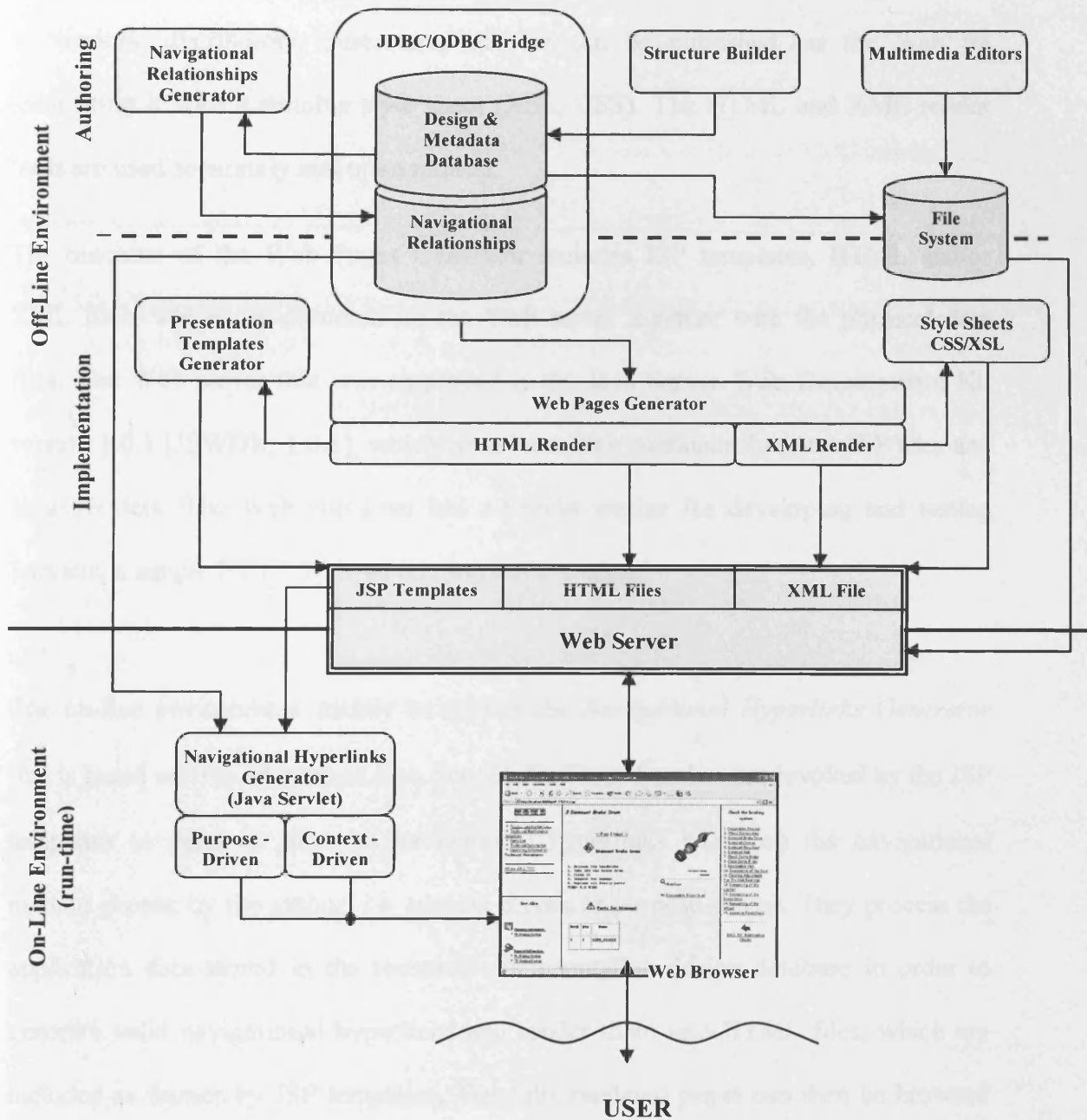


Figure 4.17 System Architecture

content of every IO in an HTML file, which is requested, at run-time, by JSP templates. Similarly, the XML render tool generates the XML file, which contains the structured application data. This file can either be used for data exchange e.g. sent to vendors, distributors, customers, etc., or can be published on the Web by associating it with a suitable style sheet (XSL, CSS). The HTML and XML render tools are used separately and upon request.

The outcome of the Web Pages Generator includes JSP templates, HTML and/or XML files, which are mounted on the Web server together with the physical data files. The Web server that was employed is the Java Server Web Development Kit version 1.0.1 [JSWDK, 1.0.1], which contains a Web container that runs JSP files and Java Servlets. The Web container has a Servlet engine for developing and testing Servlets, a simple HTTP Web server, and a JSP engine.

The on-line environment mainly comprises the *Navigational Hyperlinks Generator* that is based on two alternative Java Servlets™. These Servlets are invoked by the JSP templates to generate physical navigational hyperlinks based on the navigational method chosen by the author, i.e. context-driven or purpose-driven. They process the application data stored in the technical documentation design database in order to generate valid navigational hyperlinks and render them into HTML files, which are included as frames by JSP templates. The fully rendered pages can then be browsed using any standard Web browser.

4.7 SUMMARY

Performance support through hypermedia-based technical documentation was the central focus of this chapter. The main contribution of this chapter is a model-driven authoring methodology for Web-based technical documentation. This methodology utilises the usage-based data model for the design of technical documentation and adheres to the guidelines of structured authoring described in Chapter 3. Novel approaches, methods, and techniques have been introduced to assist in the practical implementation of this authoring methodology. These include an approach for building the hypermedia-based technical documentation structure which is based on a semantically enhanced version of the system-based structuring method. This approach includes a criterion for indexing the IOs of the technical documentation, a technique for generating dynamic identification codes for maintaining the structural and semantic properties of IOs, and a technique for the automatic code generation of hypermedia documents in two different mark-up languages, HTML and XML. A navigational model based on information semantics also has been proposed which includes an approach to generate context-driven navigational relationships, and an approach to generate purpose-driven navigational relationships. Finally, a presentation approach using frame-based templates, icons, and colours has been introduced and the system architecture used for implementing the authoring methodology has been presented.

CHAPTER 5

INTELLIGENT PERFORMANCE SUPPORT THROUGH INTEGRATED KNOWLEDGE-BASED ADAPTIVE HYPERMEDIA

This chapter is organised in three main sections. The first section introduces a methodology for providing intelligent diagnosis support through knowledge-based Expert Systems (ES). At the core of this methodology is an integrated (shallow and deep) knowledge engineering process for diagnostic ESs, which includes an integrated knowledge model. Implementation techniques concerned with automatically building a Knowledge Base (KB) for locating and correcting braking system faults in a forklift truck are presented. In addition, a general architecture for the diagnostic ES is outlined. In the second section a methodology for the “intelligent” retrieval of diagnosis information through adaptive hypermedia is described. A strategy for providing adaptive support based on a stereotype model of the knowledge of the users is discussed. This strategy is implemented on top of the technical documentation data model discussed in chapters 3 and 4. In addition, a general architecture for the adaptive hypermedia system is outlined. Finally, the third section presents a general architecture for the integration of both systems in one adaptive hypermedia diagnostic ES.

5.1 INTELLIGENT PERFORMANCE SUPPORT THROUGH KNOWLEDGE-BASED DIAGNOSTIC SYSTEM

The main focus of this section is on providing intelligent performance support through a knowledge-based diagnostic ES. It introduces an integrated approach for knowledge engineering. At the core of this approach is an integrated shallow and deep knowledge model which is also introduced. Next, in order to demonstrate the applicability and validity of this original approach, a rule-based KB for diagnosing and correcting braking faults in a forklift truck is automatically generated in a specific ES shell format. Finally, a general architecture for the diagnostic ES is presented.

5.1.1 Encapsulating Diagnosis Knowledge of Experts in an Expert System

In general, an “expert” is a person who possesses an extensive theoretical and/or practical understanding of a subject or domain, i.e. knowledge. According to Rolston [1988], although expert knowledge can be secured from a variety of sources, including documentation and existing computerised information systems, most of it must be elicited from human experts. Expert knowledge is crucial for supporting the performance of less experienced users, where human experts are often either too busy to consult or are not available at all. The importance of expert knowledge for the success of today’s organisations and businesses is indicated by Negnevitsky [2002], who states that “any successful company has at least a few first-class experts and it cannot remain in business without them”.

The word “diagnosis” is defined in [Oxford Dictionary, 1998] as the identification of a mechanical fault or a disease from its symptoms. Moreover, in [Patel et. al., 1996] diagnosis is defined as the process of locating the exact cause(s) of an error or a failure. Diagnosis assistance is a very important part of the overall assistance provided by a performance support system. In addition to identifying and locating a certain fault or error, an extended diagnosis process can be utilised to include the fault correction¹. As current products, equipment, and systems increase in size and complexity, the difficulty of diagnosing their faults increases, hence the use of expert diagnosis knowledge to support user performance is increasingly necessary. Capturing and encapsulating expert diagnostic knowledge enables permanent preservation of, and access to, this knowledge through special purpose ESs. These are computer applications which solve complicated problems that would otherwise require extensive human expertise, by simulating the human reasoning process [Rolston, 1988].

Long lists of successful knowledge-based ESs developed over the last thirty years are available in [Coffey et al., 2003; Negnevitsky, 2002; Patel et. al., 1996; and Rolston, 1988]. ESs have distinctive and unique characteristics which distinguish them from conventional computer programs, namely, they perform at a human expert level in a narrow and specialised domain, they have explanation capability, they apply heuristics (rules of thumb) to guide the reasoning, and they are able to employ symbolic reasoning when solving a problem. Application categories of ESs that are suitable for Web-based implementation may include product selection/configuration

¹ Throughout this chapter, diagnosis will be used to refer to all activities concerned with identifying, locating, and correcting faults.

advisors, job aids/performance support tools, diagnostic assistants, tutorials, etc. The three main components at the core of an ES are the Knowledge Base (KB) that contains the domain expert knowledge in a format useful for problem solving, the Inference Engine (IE) which carries out the reasoning in order to yield a solution, and the User Interface (UI), which is the means of communication between the user and the ES. Additional utilities that accompany an ES might include explanation facilities, developer interface, rules editor, debugging aids, and run-time knowledge acquisition tool.

5.1.2 Integrated Knowledge Engineering Process for Diagnostic ESs

Knowledge engineering (KE) is the process of acquiring specific-domain knowledge and building it into the KB to be used within an ES [Rolston, 1988]. This process is executed by a “knowledge engineer”, and it is considered the most important and difficult aspect of the ES development process. A general outline of a “typical” KE process is depicted in Figure 5.1. This shows the main stages in the process, namely knowledge acquisition, knowledge representation, and knowledge transformation. The figure also shows the main sources of knowledge, namely human experts, hardcopy documentation, and existing computerised information systems. The following brief description of the knowledge engineer’s role taken from Negnevitsky [2002], illustrates the typical KE process:

“The knowledge engineer is responsible for selecting an appropriate task for the ES. S/he interviews the domain expert to find out how a particular problem is solved (*knowledge acquisition*). Through interaction with the expert, the knowledge engineer

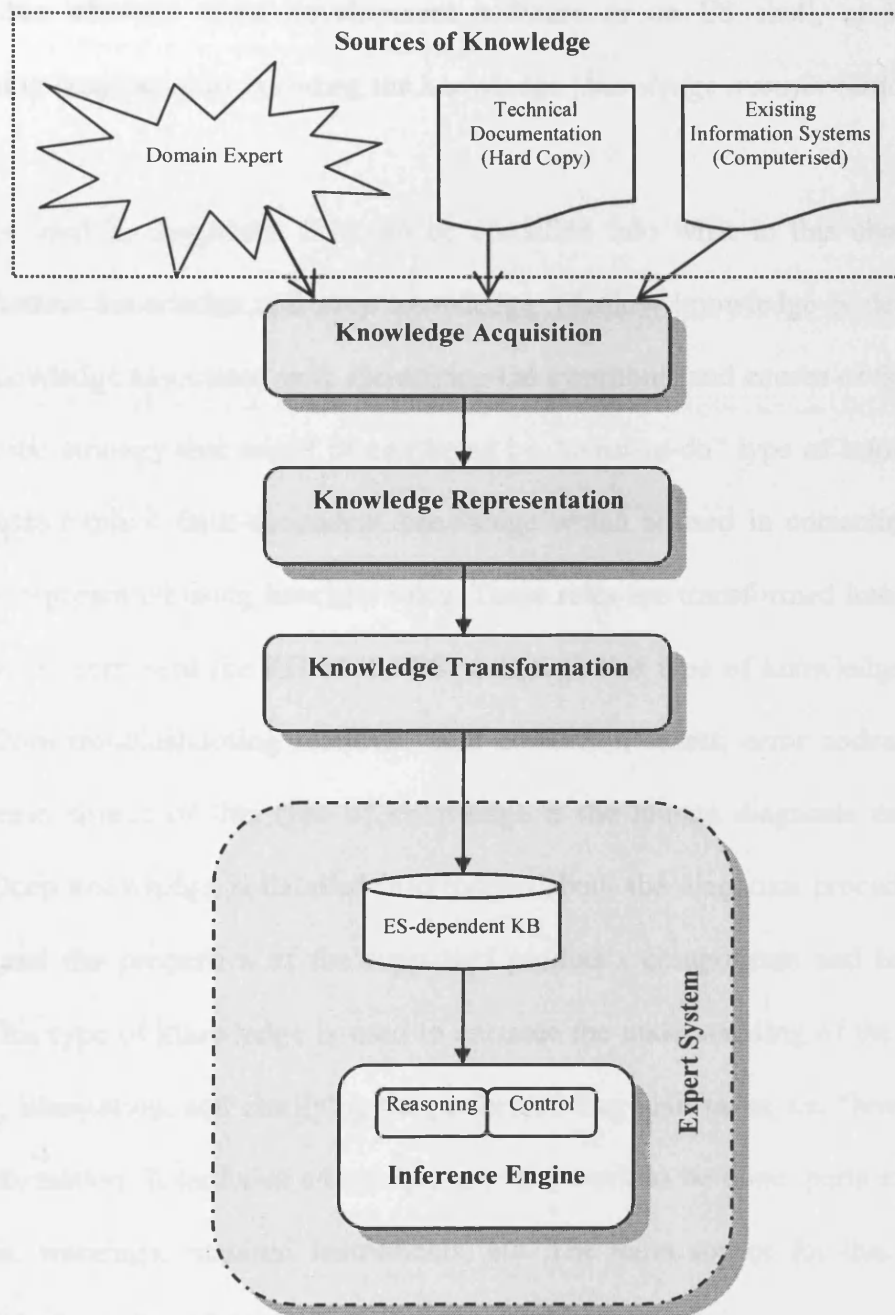


Figure 5.1 General Outline of a Typical Knowledge Engineering Process

establishes what reasoning methods the expert uses to handle facts and rules and decides how to represent them in the ES (*knowledge representation*). The knowledge engineer then chooses some development software or an ES shell, or looks at programming languages for encoding the knowledge (*knowledge transformation*).”

Knowledge used in diagnostic ESs can be classified into what in this chapter are termed, *shallow* knowledge and *deep* knowledge. Shallow knowledge is defined as specific knowledge associated with identifying the symptoms and causes of faults and the diagnostic strategy that might be employed i.e. “what-to-do” type of information. This provides explicit fault-dependent knowledge which is used in correcting faults and can be represented using heuristic rules. These rules are transformed into a set of formal rules to represent the KB of the ES. Although this type of knowledge can be acquired from troubleshooting manuals, fault correction sheets, error codes, charts, etc., the main source of this type of knowledge is the human diagnosis expert. In contrast, Deep knowledge is detailed information about the diagnosis procedure, the structure, and the properties of the supported product’s components and how they interact. This type of knowledge is used to enhance the understanding of the user by explaining, illustrating, and clarifying the performed diagnosis tasks, i.e. “how-to-do” type of information. It includes a description of the work to be done, parts involved, precautions, warnings, required instruments, etc. The main source for this type of knowledge is the technical documentation of the product itself.

Clearly, shallow and deep knowledge types complement each other, and their integration provides effective knowledge support for user performance. This hybrid approach to knowledge representation takes advantage of both types of knowledge

and minimises the limitations associated with the use of one type on its own. In addition, it provides the means to transform or upgrade existing computerised technical documentation into fully knowledge-based performance support systems.

The key issue in achieving the above is the integration of both knowledge types within the KE process at the knowledge representation stage. Figure 5.2 depicts an integrated (shallow and deep) KE process for developing diagnostic ESs. The Figure clearly indicates the integration of both knowledge types at the knowledge representation stage in an ES shell-independent manner. This integration is implemented through the inclusion of “deep knowledge” references in the KB of the ES. These references are used to retrieve a detailed description of the diagnosis procedures that exists in the hypermedia-based technical manual. In addition, they provide the means to enhance the content of the technical documentation by automatically updating its information base with useful diagnosis and correction procedures. These features will be fully described and demonstrated later. Issues related to the generation of Web-based technical documentation were discussed in the previous chapters.

5.1.3 Integrated Shallow and Deep Knowledge Model

Shallow knowledge is mainly fault-oriented and can be modelled with regard to a specific set of faults. Figure 5.3 outlines an abstract integrated shallow and deep knowledge model for diagnostic ESs. The model is based on the following hypotheses, where words in *italic* constitute the main entities of the model:

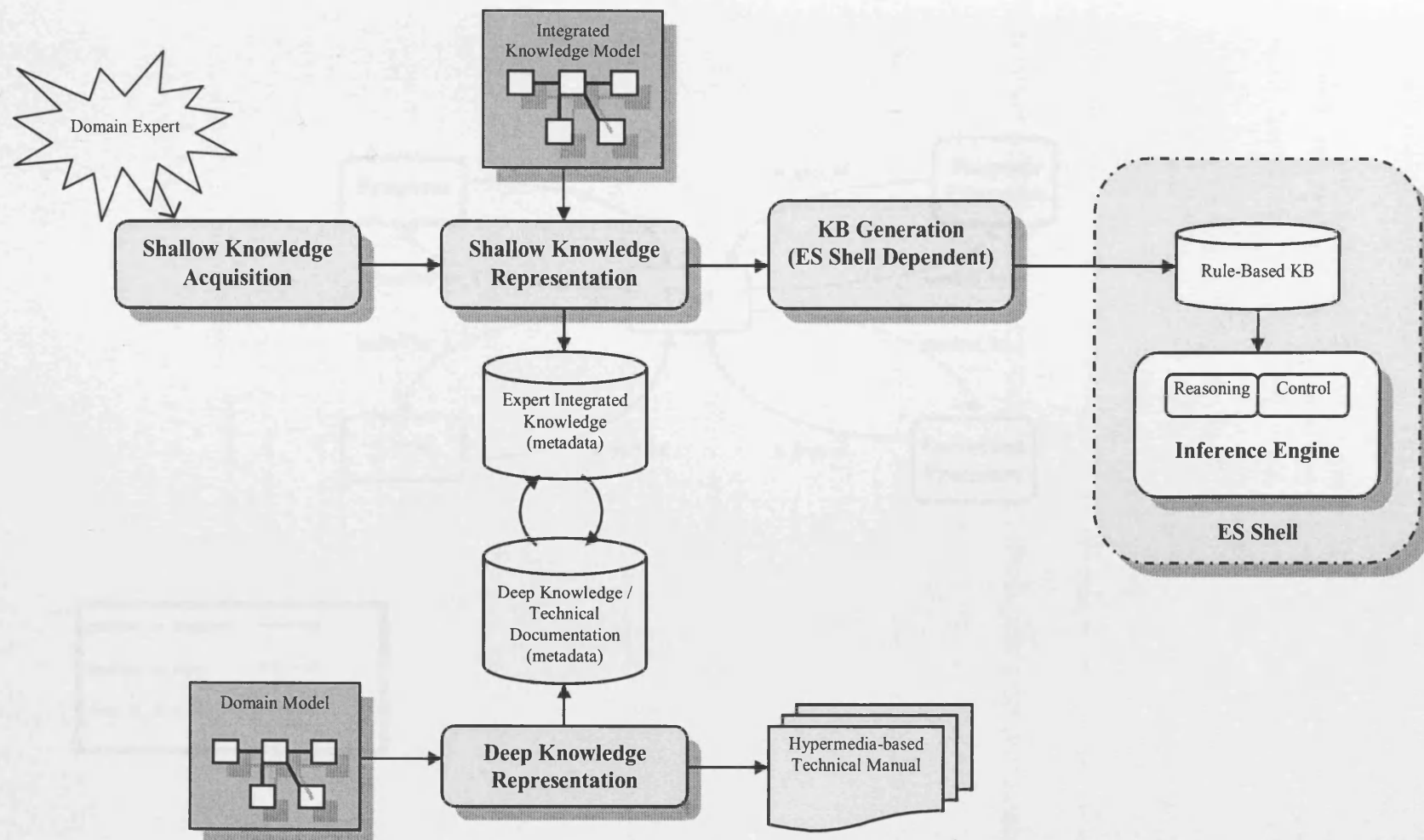


Figure 5.2 Integrated Knowledge Engineering Process for Developing Diagnostic Expert Systems

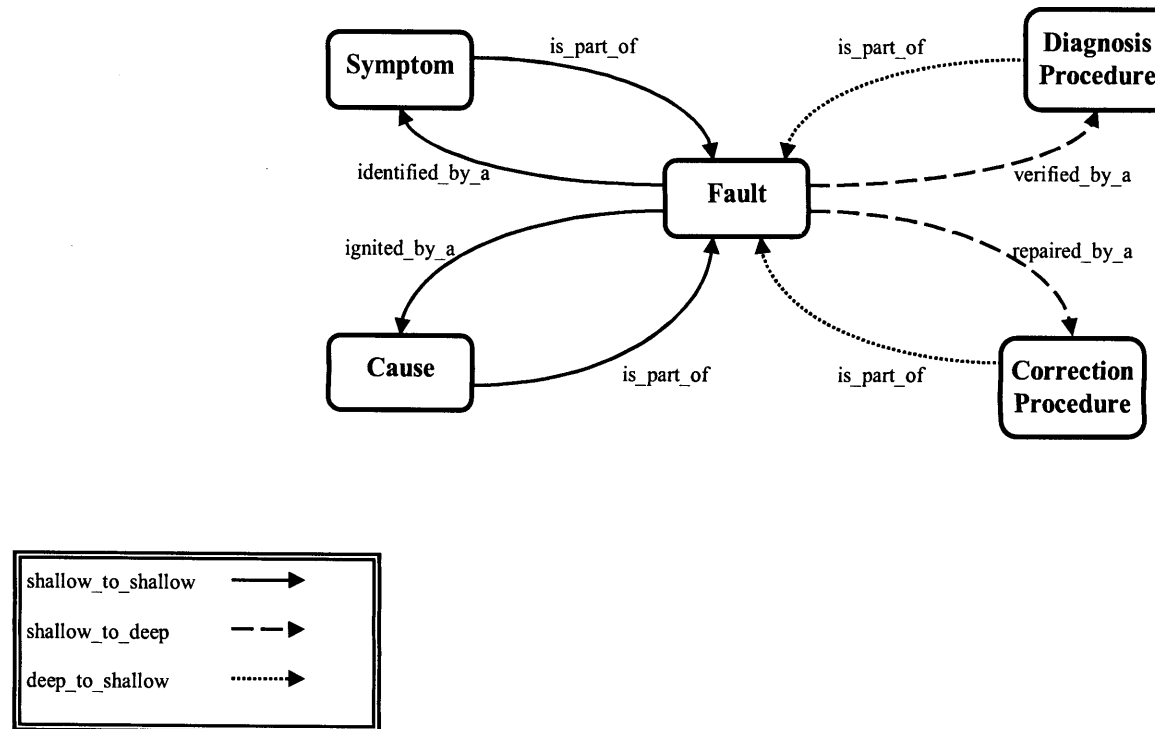


Figure 5.3 Integrated Shallow and Deep Knowledge Model for Diagnostic Expert Systems

- Every *fault* has one or more *symptoms* and *causes*,
- A *symptom* is an identifiable sign of existence of a *fault*,
- A *cause* is something that ignites a *fault*,
- Every *fault* has a *diagnosis* and a *correction* procedure (deep knowledge),
- A *diagnosis procedure* verifies a *fault*, and
- A *correction procedure* repairs a *fault*.

It is worth noting that the information associated with diagnosis and correction procedures are of deep knowledge type. Through the association of every fault (shallow knowledge) with the corresponding diagnosis and correction procedure, the relation between the shallow and deep knowledge can be established.

This integrated knowledge model is implemented within a database schema for use along with the required expert knowledge data. Figure 5.4 shows an E-R diagram representing the corresponding database schema of the integrated knowledge model. It outlines the database tables constituting the physical repository of the shallow knowledge namely, symptoms class *ES_S_Class*, symptoms *ES_Symptoms*, causes class *ES_C_Class*, causes *ES_Causes*, fault symptoms *ES_FSs*, fault causes *ES_FCs*, and faults *ES_Faults*. The interaction between the shallow knowledge and deep knowledge is indicated through the relation between the faults table *ES_Faults* and the technical documentation metadata table *IOs* (information objects).

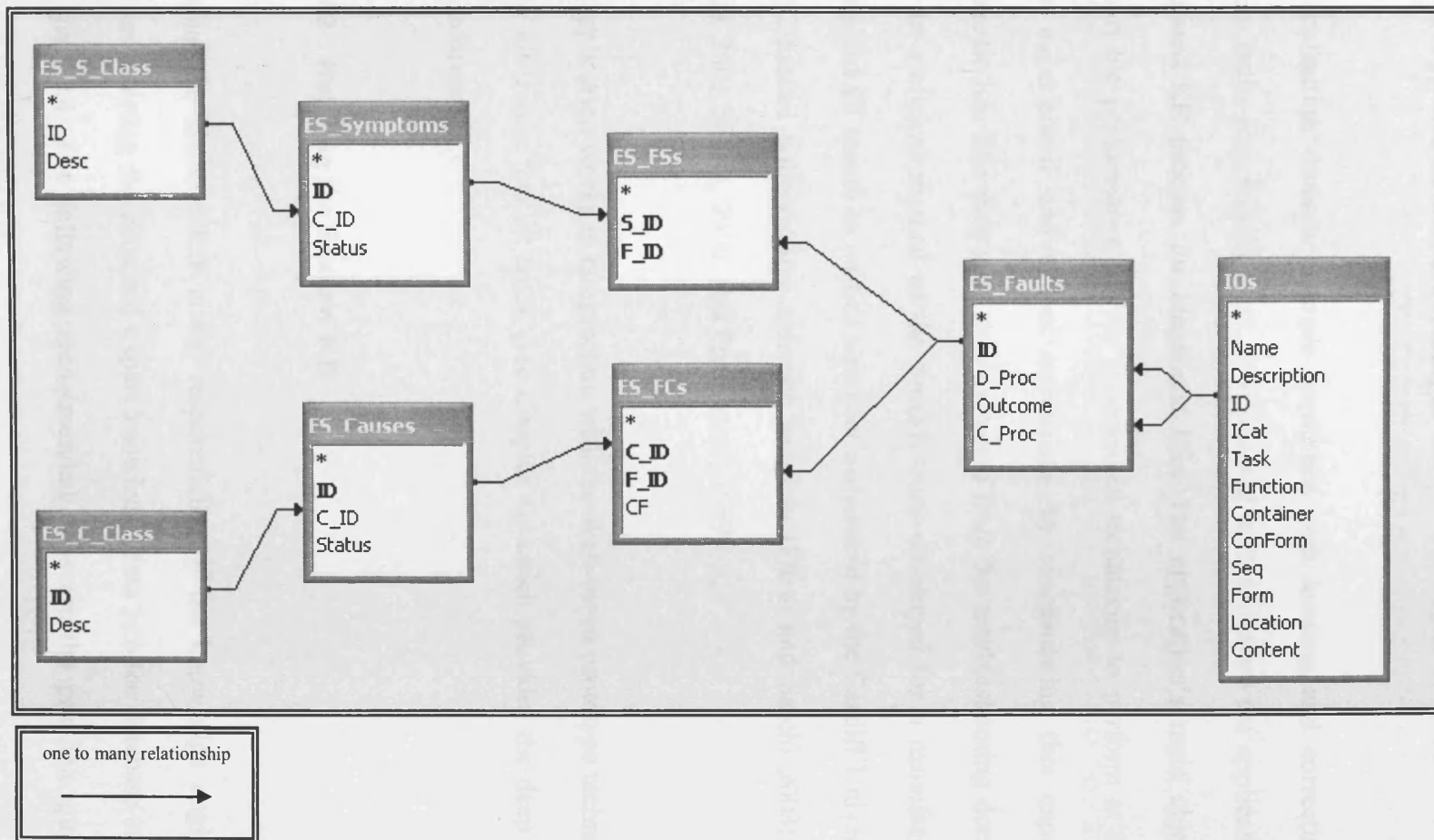


Figure 5.4 E-R Diagram Representing the Database Schema of the Integrated Knowledge Model

5.1.4 Rule-Based KB for Fault Diagnosis

5.1.4.1 Application Domain Example

An application domain example concerned with locating and correcting braking system faults in a forklift truck, was selected to demonstrate the applicability of the integrated KE process for diagnostic ESs. The application's main objective is to support the performance of less experienced technicians to perform at the level of more experienced and skilled technicians by encapsulating this expertise in a diagnostic ES. The data used was extracted from the troubleshooting documentation and the technical manual of the forklift truck developed for a manufacturer of all terrain forklift trucks as part of a project undertaken by the Cardiff University within an EC-funded collaborative research program [Pham and Setchi 2003; Pham and Setchi, 2001, Setchi, 2000, and Pham et al., 1999].

The application works in conjunction with the Web-based prototype technical manual of the all-terrain forklift truck (see Chapter 4), which provides the deep knowledge type information.

5.1.4.2 Building the Shallow KB

Building the shallow KB is the responsibility of the knowledge engineer which requires inserting the acquired expert knowledge data into the database tables shown in Figure 5.4. The following recommended steps are the principle guidelines for building the shallow KB:

1. Identify the set of faults to be tackled. Faults are given a unique identification code (ID), and are associated with a diagnosis procedure, an abnormal diagnosis outcome, and a correction procedure. Within this context, a fault is perceived as an abnormal (faulty) outcome of a diagnosis procedure. The description of the diagnosis and correction procedures may already be included in the technical documentation, and in this case only a reference to their ID is needed.
2. Identify the set of symptoms and causes that will be used in the faults' determination. Symptoms and causes are identified by a unique ID, class description and status, e.g.: the symptom "*pedal is soft*" is coded as (*[ID: 1]*, *[description: pedal feel]*, *[status: soft]*).
3. Associate every fault with one or more symptoms selected from the full set of symptoms identified in step 2. Update existing symptoms or add new symptoms if needed. Similarly, associate faults with fault causes and attach a certainty factor CF with every "fault-cause" combination. CF represents a measure of the expert confidence that a cause *C* is most likely to ignite a fault *F*. It is used in the ES to direct the dialogue with the user by presenting the causes with higher CFs first, which helps in converging more quickly to a solution. This approach introduces the expert knowledge into the user dialogue sequencing process implemented by the ES.

Figure 5.5 shows the structured shallow knowledge data used for diagnosing the braking system faults in the forklift truck. The diagnosis data is structured in tables of possible causes, symptoms, and faults. In addition, the figure shows the tables which associate faults with their symptoms (*Fault Symptoms*) and their possible causes (*Fault Causes*). A fault symptom or cause class can have one or more user-defined

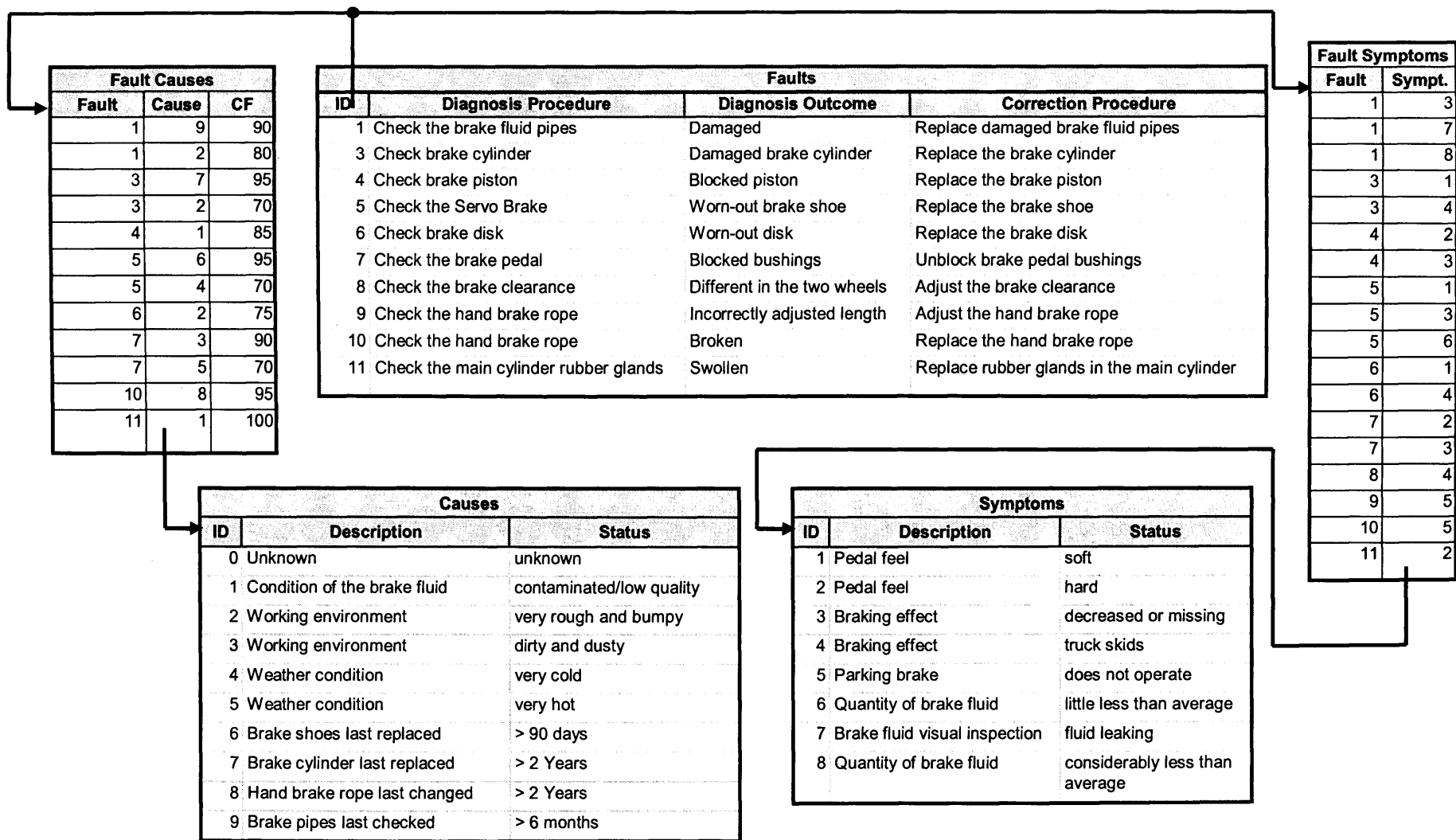


Figure 5.5 Structured Shallow Knowledge Data for Diagnosing Braking System Faults in the Forklift Truck

status values, e.g.: the fault symptom class “*pedal feel*” has two user-defined status values, “*soft*” and “*hard*”, and the fault cause class “*working environment*” has also two user-defined status values, “*very rough and bumpy*” and “*dirty and dusty*”. In addition, a default system-defined status value “*normal*” is automatically associated with every symptom and cause class. Moreover, every fault is associated with a diagnosis procedure, an abnormal diagnosis outcome and a correction procedure, e.g.: the diagnosis procedure for fault “*I*” is “*check the brake fluid pipes*”, the abnormal diagnosis outcome is “*damaged*”, and the recommended fault correction procedure is “*replace damaged brake fluid pipes*”. A fault diagnosis procedure can have one or more abnormal outcome values, e.g. the outcome of “*check the hand brake rope*” diagnosis procedure is either “*incorrectly adjusted*” or “*broken*”. Similarly a default system-defined outcome value “*normal*” is automatically associated with every fault diagnosis procedure.

A user interface has been constructed on top of the database tables in order to facilitate the updating process of the structured shallow knowledge data in a graphical and user-friendly way. This structured set of data has been used to automatically generate a prototype rule-based ES for diagnosing braking faults, and to demonstrate the user performance support provided by the adaptive hypermedia system. The quality of the data acquired from the diagnosis expert has a huge effect on the final performance of the ES.

5.1.4.3 Abstract Rule Format for Representing the Diagnostic Strategy

In a rule-based ES, knowledge is represented through a set of IF-THEN production rules. The inference engine compares each rule stored in the KB with available facts in order to reach a conclusion. The diagnostic strategy is represented through abstract IF-THEN rules which are realised from the integrated knowledge model discussed in 5.1.3. The following abstract rules are used to generate the rule-based KB:

Rule (I):

IF	[Symptom]	
AND	{ [Cause] }	,where {} means optional
AND	[Abnormal Diagnosis]	, requires deep knowledge
THEN	[Fault (x)]	

Rule (II):

IF	[Fault (x)]	
THEN	[Correct (x)]	, requires deep knowledge

These rules read as follows:

If a symptom is found and a cause is known and the diagnosis of a certain fault is abnormal then a fault is identified. If a fault is identified then the associated correction action is recommended.

Example:

IF	[Pedal is soft]	
AND	[The brake fluid is contaminated/low quality]	
AND	[Check brake piston for blockage is confirmed true]	
THEN	[Brake piston is blocked]	
IF	[Brake piston is blocked]	
THEN	[Unblock the brake piston]	

A fault symptom is the initial fact that can be determined because it is easily identifiable by the user, e.g.: soft pedal, decreased braking effect, etc. When one or more fault symptoms are identified, the associated fault cause(s) are checked for existence. Then the associated faults are diagnosed until a single fault is isolated. When a single fault is identified, the associated correction procedure is recommended. Fault causes are considered “optional” because, in certain situations, the fault cause(s) cannot be identified and fault symptoms are sufficient to lead directly to the identification of the fault.

5.1.4.4 Automatic Generation of the Rule-Based KB in e2gLite ES Shell

Format

Rule-based KB generation is the process of transforming the knowledge representation of the expert data into a specific rule-based ES shell format. An ES Shell is an ES without the KB, where all the knowledge engineer has to do is to supply the knowledge in a rule-based format in order to solve a problem. The ES shell that has been selected for demonstration purposes is the e2gLite ES shell [e2gLite, 2003], which is freely available on the Web. The e2gLite ES shell is a Java applet™ that is embedded in a Web page and downloaded from the Web server by the user's browser. The applet loads the KB from the server and then runs entirely on the browser. It uses a simple e2gLite language for encoding KBs, and it is fully Web enabled. In addition, the e2gLite ES shell comes in a package that includes an explanation facility and KB debugging services such as KB analysis.

A Java-based special purpose tool, *KB Generator*, has been created in order to generate the KB automatically by transforming the structured diagnosis data into rules and prompts in e2gLite ES shell format. The algorithm used to generate the KB is as follows:

Rules Generation:

1. for every fault do:
 - 1.1 for every associated fault cause(s), sorted (descending) by their CF, do:
 - 1.1.2 find all associated fault symptoms
 - 1.1.3 insert the corresponding abstract type (I) rule in the KB file
 - 1.2 insert an abstract type (I) rule in the KB file, discarding fault causes.
2. for every fault do:
 - 2.2 Retrieve the details of the fault correction procedure from the technical manual (deep knowledge)
 - 2.3 insert an abstract type (II) rule in the KB file

Prompts Generation:

1. for every fault symptom class do:
 - 1.1 find all possible status values
 - 1.2 insert a multiple choice type prompt in the KB file
 - 1.3 add the default status value “normal”
2. for every fault cause class do:
 - 2.1 find all possible status values
 - 2.2 insert a multiple choice type prompt in the KB file
 - 2.3 add the default status value “normal”
3. for every fault diagnosis procedure do:
 - 3.1 find all possible abnormal outcome values
 - 3.2 retrieve the required details of the fault diagnosis procedure from the technical manual (deep knowledge)
 - 3.3 insert a multiple choice type prompt in the KB file
 - 3.4 add the default outcome value “normal”

The KB rules that will be automatically generated using the above algorithm are an instantiation of the abstract rules defined earlier. The abstract rule (I) is instantiated as follows:

If	S ₁	If	S ₁	If	S ₁	
and	S ₂	and	S ₂	and	S ₂	
	.		.		.	
	.		.		.	
and	S _n	and	S _n	and	S _n	
and	C ₁	and	C ₂	and	C _n
and	D ₁	and	D ₁		and	D ₁
	,					... etc
Then	F ₁	Then	F ₁	Then	F ₁	

where, S_n is the nth identified symptom, C_n is the nth fault cause, and D₁ is the diagnosis procedure of fault F₁, i.e. D₁ confirms the existence of F₁. Moreover, the abstract rule (II) is instantiated as follows:

If	F ₁	If	F ₂	If	F _n	
Then	Cr ₁	Then	Cr ₂	Then	Cr _n
	,					... etc

where, F_n is the nth fault, and Cr_n is the correction procedure of F_n.

Figure 5.6 shows the automatically generated e2gLite rules and prompts for fault number “1” which is “*damaged brake fluid pipes*”. In the figure, Rule [1] prompts the user to determine the status of three symptom classes namely, *braking effect*, *brake fluid visual inspection*, and *quantity of brake fluid*. If these are confirmed by the user to be “*decreased or missing*”, “*fluid leaking*” and “*considerably less than average*”, respectively, then the first fault cause is displayed for determination, i.e. [*Brake pipes last checked*] = “> 6 months”, otherwise the second cause is displayed, i.e. [*Working environment*] = “very rough and bumpy”. If neither of these two causes are confirmed

REM Rules:

RULE [1]
If [Braking Effect] = "decreased or missing" and
[Quantity of brake fluid] = "considerably less than average" and
[Brake fluid visual inspection] = "fluid leaking" and
[Brake pipes last checked] = "> 6 months" and
[Check the brake fluid pipes] = "Damaged"
Then [Fault Code] = "1" and
[Cause] = "Brake pipes last checked is > 6 months"

Symptoms
Cause
Diagnosis
Identified fault
Cause description

RULE [2]
If [Braking Effect] = "decreased or missing" and
[Quantity of brake fluid] = "considerably less than average" and
[Brake fluid visual inspection] = "fluid leaking" and
[Working environment] = "very rough and bumpy" and
[Check the brake fluid pipes] = "Damaged"
Then [Fault Code] = "1" and
[Cause] = "Working environment is very rough and bumpy"

Fault description

RULE [15]
If [Fault Code] = "1"
Then [Fault] = "The outcome of Check the brake fluid pipes procedure is: Damaged" and
[Recommendation] = "Replace damaged brake fluid pipes (Man:10.10.9.1)"

Fault description
Recommended fault correction procedure
Reference to the technical documentation (How to do?)

REM Prompts:

PROMPT [Braking Effect] MultChoice
"Braking Effect? "
"decreased or missing"
"truck skids"
"Normal"

Symptom class
Symptom status
Default status

PROMPT [Brake fluid visual inspection] MultChoice
"Brake fluid visual inspection? "
"fluid leaking"
"Normal"

PROMPT [Quantity of brake fluid] MultChoice
"Quantity of brake fluid? "
"little less than average"
"considerably less than average"
"Normal"

PROMPT [Brake pipes last checked] MultChoice CF
"Brake pipes last checked? "
"> 6 months"
"Normal"

PROMPT [Working environment] MultChoice CF
"Working environment? "
"dirty / dusty"
"very rough and bumpy"
"Normal"

PROMPT [Check the brake fluid pipes] MultChoice
"You should Check the brake fluid pipes (Man:10.9.9.1), for the following fault(s): "
"Damaged"
"Normal"

Abnormal value
Fault diagnosis procedure
Reference to the technical documentation (How to do?)

REM Goals:

GOAL [Cause]
GOAL [Fault]
GOAL [Recommendation]

Goals of the ES

Figure 5.6 Automatically Generated e2gLite Rules and Prompts for Fault No. 1 (Damaged Brake Fluid Pipes)

and no other faults are associated with these symptoms, then fault “1” will be fired depending on the identified symptoms with unidentified cause. Three generic goals were set for the ES to determine namely, the fault, the fault cause, and the recommended correction action. These are shown in the clauses at the bottom of the figure starting with the word “GOAL”.

The inference engine of the e2gLite ES shell uses a combination of forward and backward chaining to determine the facts required in order to reach a valid conclusion. Figure 5.7 depicts the decision flow chart for determining fault “1”, which illustrates the diagnostic strategy of the ES as a whole. The inference engine tries to conclude its stated goals namely, cause, fault and recommendation. It tries to determine the existence of symptoms S_1 , S_2 , and S_3 , and if confirmed it tries to establish the causes C_1 or C_2 . If the cause is not confirmed and no other faults are determinable, it concludes the fault associated with the already confirmed symptoms, and it declares the cause “not found”.

Figure 5.8 illustrates the user interface of the diagnostic ES using the example scenario described above. It shows the dialogue between the system and the user, and the final system recommendation based on the user feedback to symptoms, causes, and fault diagnosis. The number of rules generated for the cases shown in Figure 5.5 is 32. These comprise 22 type (I) rules and 10 type (II) rules. The number of prompts generated is 21. These comprise a prompt for every distinct cause class (7), symptom class (5), and fault class (9). The source code for the KB Generator tool is shown in Appendix E.1, and the complete rule-based KB generated in e2gLite ES shell format is shown in Appendix E.2.

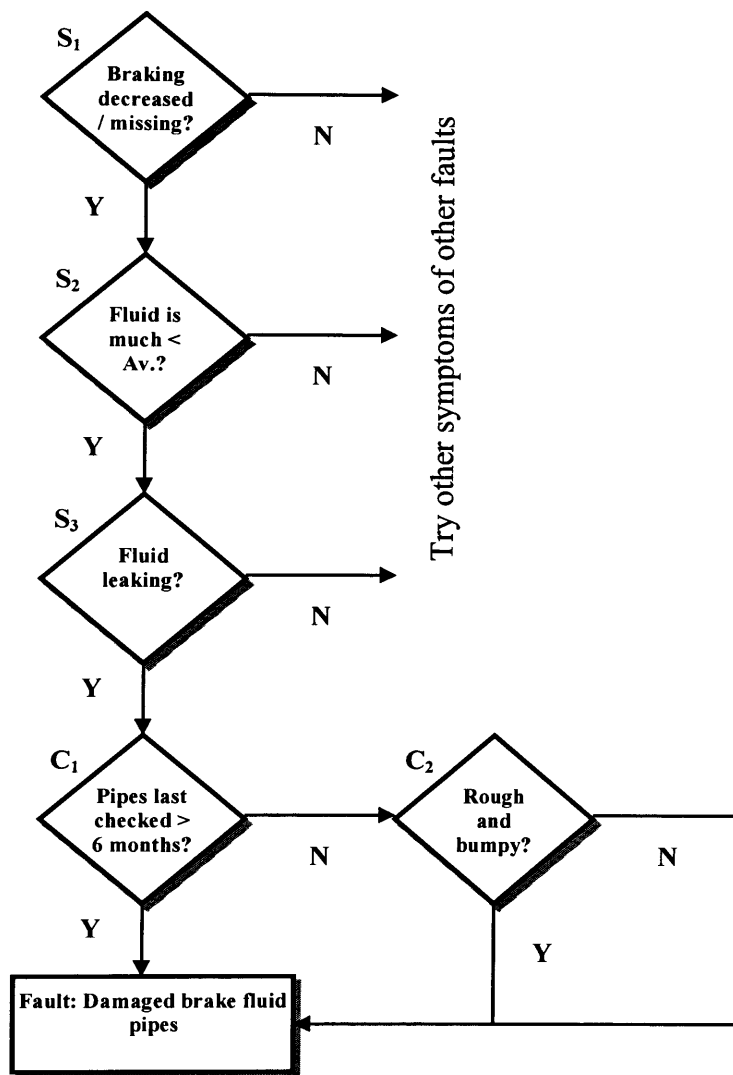


Figure 5.7 Decision Flow Chart for Determining Fault No. 1 (Damaged Brake Fluid Pipes)

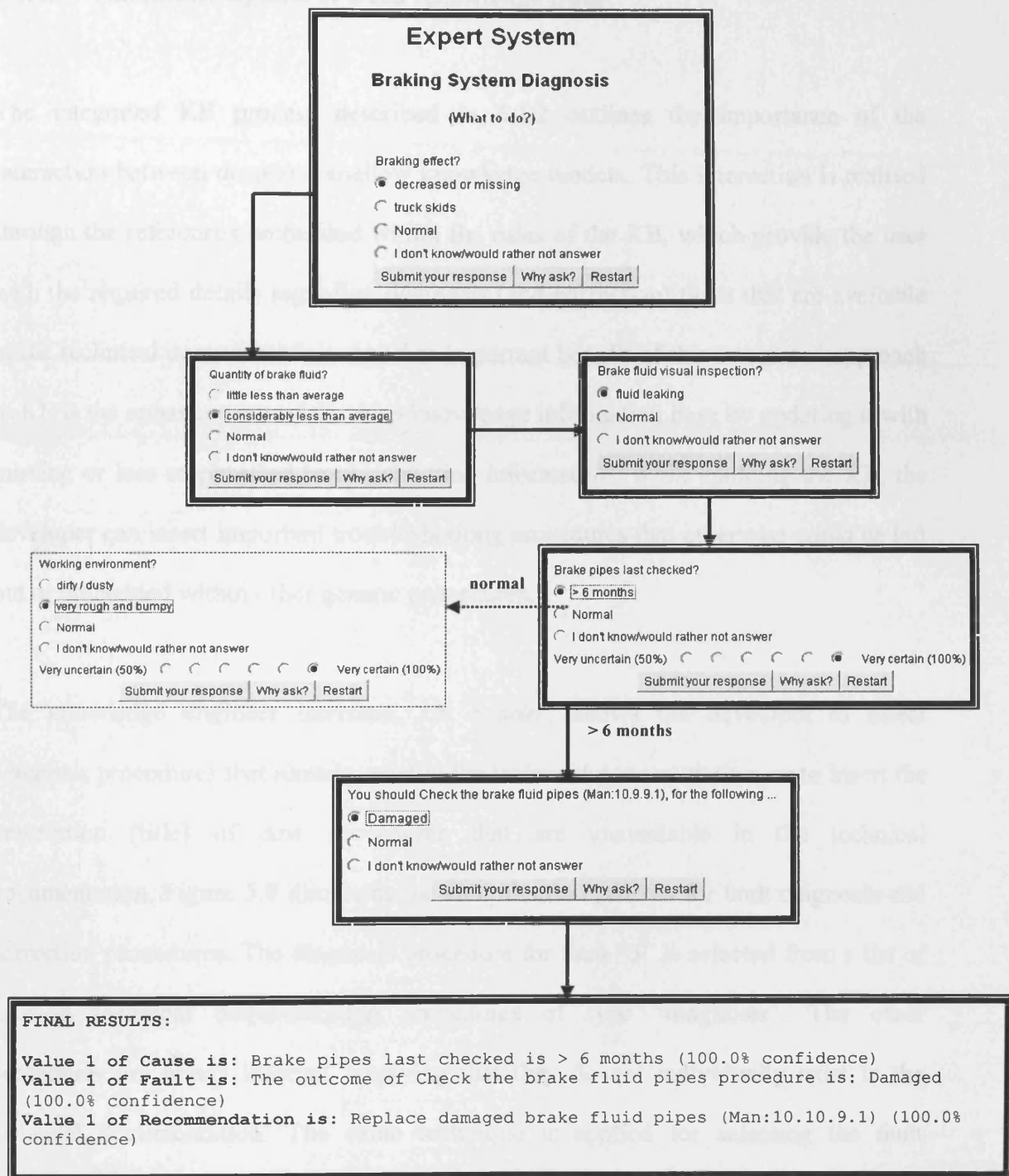


Figure 5.8 User Interface and Example Scenario of the Diagnostic ES

5.1.4.5 Automatic Update of Deep Knowledge Data

The integrated KE process described in 5.1.2 outlines the importance of the interaction between deep and shallow knowledge models. This interaction is realised through the references embedded within the rules of the KB, which provide the user with the required details regarding diagnosis (and correction) tasks that are available in the technical documentation. Another important benefit of this integrated approach to KE is the enhancement of the deep knowledge information base by updating it with missing or less emphasised troubleshooting information. While building the KB, the developer can insert important troubleshooting procedures that otherwise could be left out or embedded within other generic procedures.

The knowledge engineer interface, *KB builder*, allows the developer to select diagnosis procedures that already exist in the technical documentation, or to insert the description (title) of new procedures that are unavailable in the technical documentation. Figure 5.9 illustrates the identification process for fault diagnosis and correction procedures. The diagnosis procedure for fault “5” is selected from a list of existing technical documentation procedures of type “diagnosis”. The other procedures are newly inserted, implying that they do not individually exist in the technical documentation. The same technique is applied for selecting the fault correction procedures. During the automatic KB generation process, the *KB generator* realises the new procedures (diagnosis and correction) and inserts them in the technical documentation. New diagnosis and correction procedures are associated with the documentation category “troubleshooting”, and with the task type “diagnosis” and “correction”, respectively. In addition, the KB generator replaces the

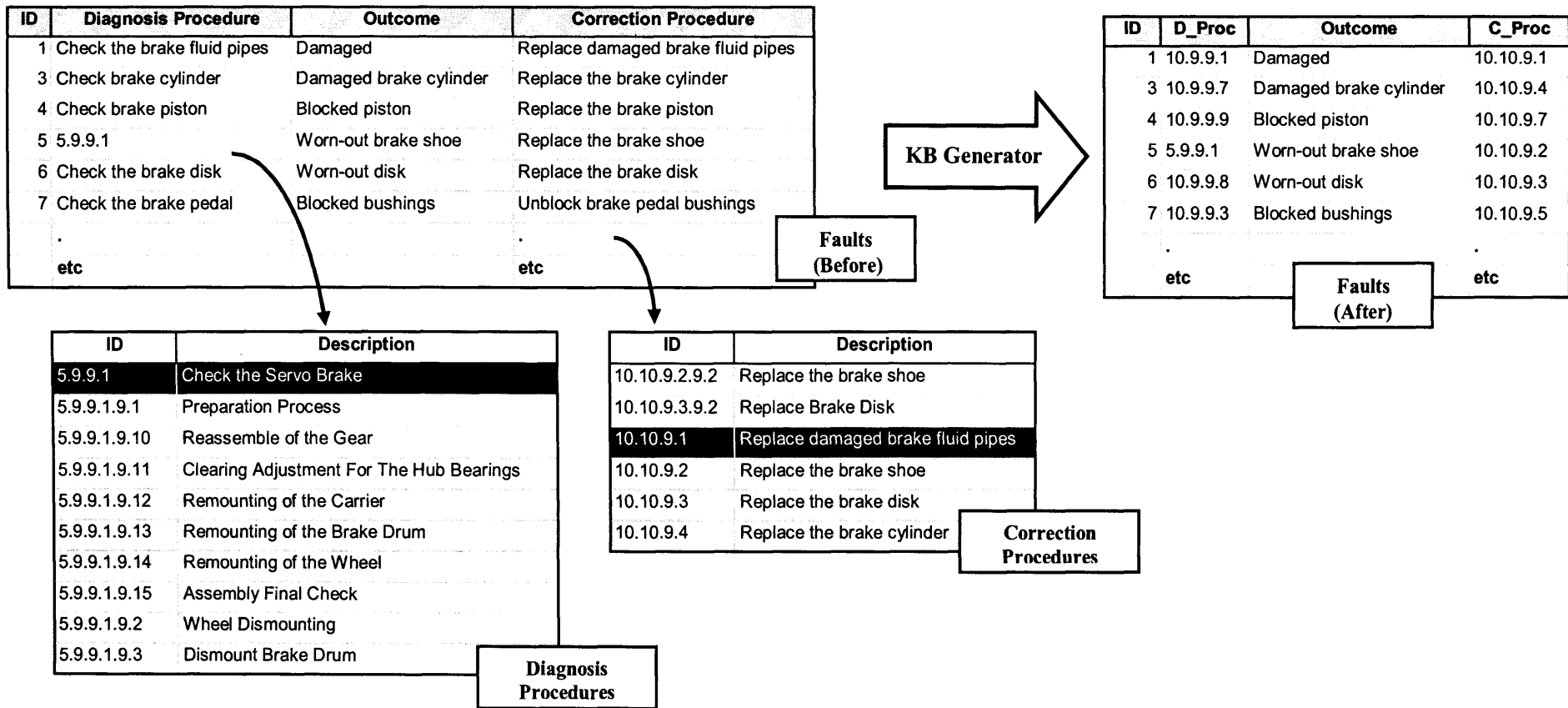


Figure 5.9 Identification of Faults Diagnosis and Correction Procedures

description of the new procedures with their newly assigned technical manual identification codes, as shown in Figure 5.9.

Although the actual description of the steps of these newly inserted troubleshooting procedures have to be composed manually, the connection with the rest of the product components is established automatically (see 4.4.2.2). This includes deeper and more detailed information such as: parts hierarchy, technical descriptions, specifications, performance data, planning information, etc.

5.1.5 General Architecture for the Diagnostic Expert System

The system architecture shown in Figure 5.10 illustrates the main components of the diagnostic expert system. These are the *KB Builder*, the *KB Generator*, and the *e2gLite ES Shell*.

The *KB Builder* is the main interaction of the domain expert with the KB. It facilitates the building and updating of the KB through a Graphical User Interface (GUI) created on top of the database tables. The KB Builder supports and sustains the validity and consistency of the knowledge by employing constraints through its GUI. It also supports the interaction interface with the technical documentation metadata (deep knowledge). In brief, the KB Builder maintains a structured data set of expert diagnosis knowledge which adheres to the integrated knowledge model.

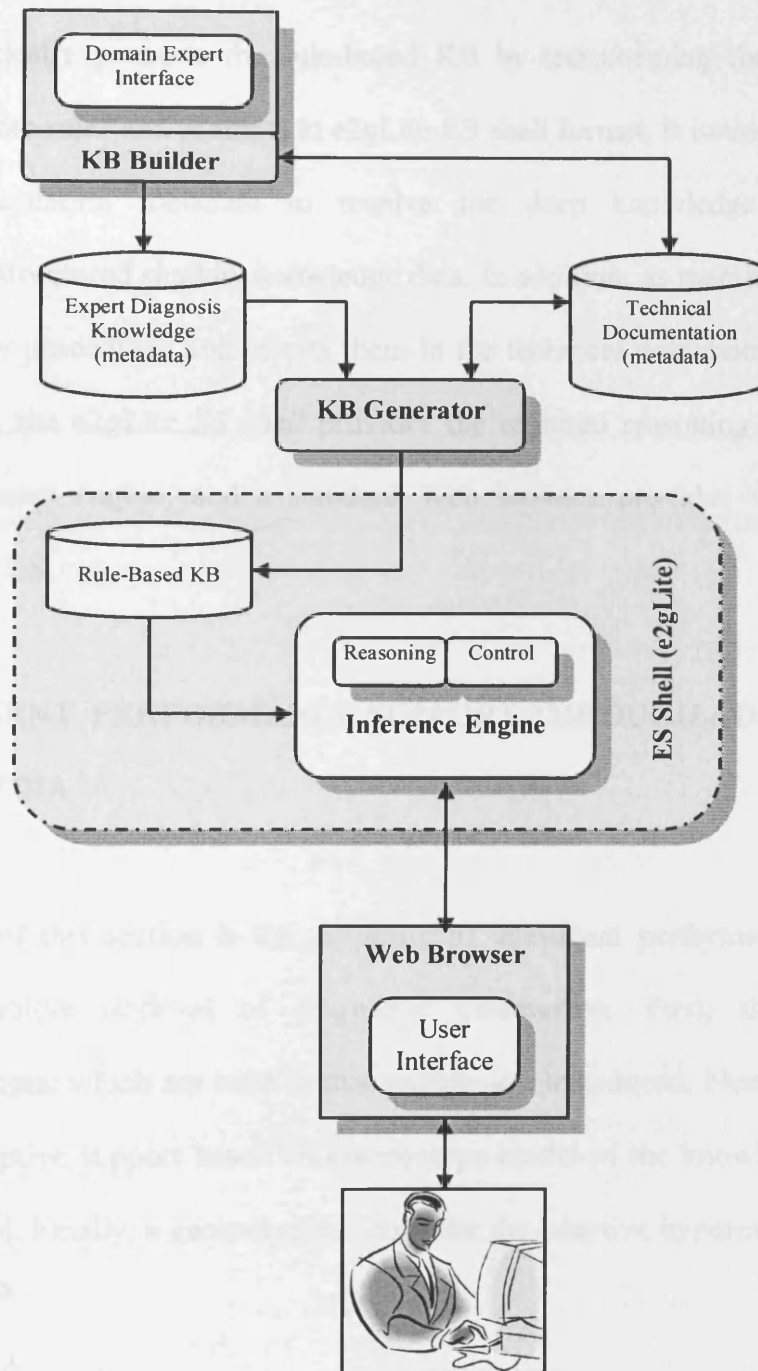


Figure 5.10 General Architecture of the Diagnostic Expert System

The *KB Generator* is a Java-based ES shell-dependent tool that has been created in order to automatically generate the rule-based KB by transforming the structured knowledge data into rules and prompts in e2gLite ES shell format. It interacts with the technical documentation metadata to resolve the deep knowledge references embedded in the structured shallow knowledge data. In addition, as mentioned earlier, it realises the new procedures and inserts them in the technical documentation meta-database. Finally, the e2gLite *ES shell* provides the required reasoning and control through its inference engine, and a standard Web browser provides the end-user interface with the ES.

5.2 INTELLIGENT PERFORMANCE SUPPORT THROUGH ADAPTIVE HYPERMEDIA

The main focus of this section is the provision of intelligent performance support through the adaptive retrieval of diagnostic information. First, the adaptive hypermedia concepts, which are used in this section, are introduced. Next, a strategy for providing adaptive support based on a stereotype model of the knowledge of the users is introduced. Finally, a general architecture for the adaptive hypermedia system is outlined.

5.2.1 Adaptive Retrieval of Hypermedia-Based Diagnostic Information

An adaptive hypermedia system has been developed in order to improve the retrieval of the fault diagnosis and correction information required by the users of the diagnostic ES. This information is stored as logical metadata in a technical

documentation repository, and as physical data in a file system. The system applies some features of the user in order to ensure that the retrieved information is relevant and its presentation suitable to the user.

5.2.1.1 Stereotype Model for User Knowledge

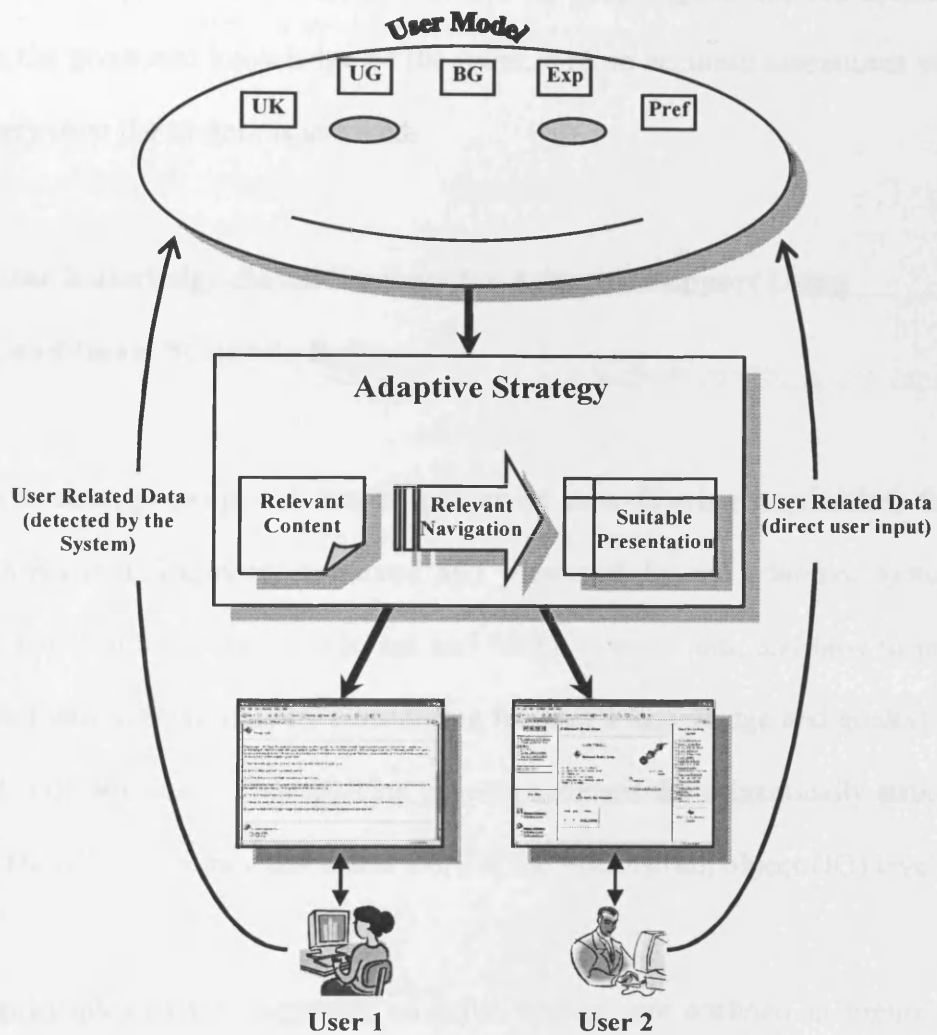
Five different features of the user are identified which can be used as a source of the adaptation, namely, user knowledge, goal(s), background, experience, and preferences. The first two are the most important and widely used features of the user in adaptive hypermedia and they are, along with other features, encapsulated in a user model, which is a representation of the user's state of mind. The knowledge of users in a certain subject is user-dependent, i.e. it changes from one user to another. In contrast, the user's goal or task is a feature related to the context of the user's work with the hypermedia system, i.e. why the user is using the hypermedia system, and what does s/he want to achieve, rather than to the user as an individual. The identification of the user's knowledge and goal is the first step for providing adaptive hypermedia support, in which users are provided with the "relevant information". Within this context, "relevant information" implies, first, information related to the user's goal(s), and second, information that the user can or is prepared to comprehend (user's knowledge).

Another important issue in adaptive hypermedia is the features of the system that can differ for different users, i.e. what can be adapted in adaptive hypermedia. These are the content of the hypermedia pages (adaptive content), the access methods and hyperlinks from these pages (adaptive navigation), and the presentation of these pages

(adaptive presentation). Figure 5.11 sums up the adaptive hypermedia issues mentioned above in an abstract overview of adaptive hypermedia systems. The figure outlines an adaptive strategy at the core of the system which guides the provision of relevant hypermedia content and navigation, and suitable presentation. The user features (user model attributes) are either implicitly detected by the system or explicitly provided through direct input from the user, e.g. through a questionnaire.

User models are often represented by either an “overlay” model or by a simpler “stereotype” model. The former is a representation of the user’s features as an overlay on top of the domain model. The latter distinguishes several typical or stereotype users. The context of the user’s work (goal) is a deterministic factor in selecting the best type of user model to use and the features of the user to consider. In the fault diagnosis domain, the user’s high level goal is “diagnosis”, which is stable throughout the user interaction with the adaptive system. In addition, the user’s low level goal is the diagnosis or correction of a particular fault, which may change quite often during the work session.

The accurate estimation, by the adaptive system, of the user’s knowledge of a certain subject and their low level goal is a complicated process, which normally provides poor estimations. This will, eventually, result in providing the user with irrelevant and/or unsuitable information. In fault diagnosis, the required information is usually concise and precise (e.g. a single diagnosis or correction procedure), which enables the users themselves to estimate their knowledge of the task they are performing. The suggested stereotype user model used to represent the user’s knowledge of the subject



UK: User Knowledge
UG: User Goal
BG: Background
Exp: Experience
Pref: Preferences

Figure 5.11 Adaptive Hypermedia Systems - Abstract Overview

domain includes three stereotype values, namely poor, sufficient, and outstanding. The user goal is identified through the provision of the identification code of the required subject supplied by the diagnostic ES. By adopting this explicit method for identifying the goals and knowledge of the users, a more accurate assessment will be yielded every time the system is invoked.

5.2.1.2 User Knowledge-Based Strategy for Adaptive Support Using Conditional Semantic Rules

The adaptive strategy proposed can be perceived as a filtering mechanism for the domain information fragments retrieved and presented by the adaptive system. It determines which information is relevant and which is irrelevant, and how to present relevant information fragments by considering the user's knowledge and goal(s) using conditional semantic rules. This filtering process accesses the semantically structured data of the technical documentation and work at the information object (IO) level.

The main principles of the suggested adaptive strategy are outlined in Figure 5.12. The figure shows the relationship between the current user knowledge and the adaptive support features provided to the user. As the user's knowledge increases from "poor" through "sufficient" to "outstanding", the complexity and detail of information and the number of visible references increases, and vice versa. This is because a qualified user is prepared to accept, and even demands, more detailed and deeper information. On the other hand, the level of additional explanations, guidance, and clarification annotations decreases as the current user's knowledge increases. This

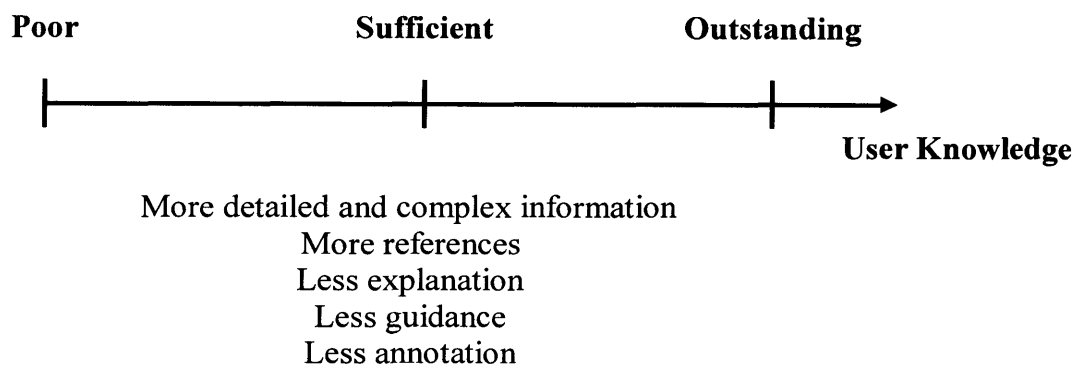


Figure 5.12 Main Principles of the Adaptive Strategy

is because novice users require explanation, guidance, and clarification while more qualified users need them less.

Table 5.1 depicts, in more detail, the relationship between the adaptive support provided by the hypermedia system and the user's knowledge. The adaptive support is categorised by content, navigation, and presentation. *Adaptive content support* is achieved by adapting the main content of the page accessed by a particular user to current user knowledge. It is assessed using three benchmarks:

- *Main content* is the content directly related to the requested information, e.g. the steps of a diagnosis procedure. The actual content can be displayed “expanded” to the viewer, or outlined “collapsed” in an index-like format with clickable headings.
- *Main content clarification* includes all the examples, comments, illustrations, etc. associated with the main content, e.g. a graphical illustration of a part.
- *Main content advice* includes all the recommendations, requirements, warnings, cautions, etc. associated with the main content, e.g. parts required to repair a fault, related warnings, and so forth.

Adaptive navigation support is used to help users to find their way in the hypermedia space by adapting the page access methods and the provision of hyperlinks to current user knowledge. It is assessed using four benchmarks:

Table 5.1 Relationship between Adaptive Support and User Knowledge

Adaptive Support \ User Knowledge	Poor	Sufficient	Outstanding
Content:			
Main content	expanded	expanded	collapsed
Main content clarification	yes	no	no
Main content advice	yes	yes	no
Navigation:			
Links to fundamental information	visible	visible	visible
Links to deep, complex and detailed information	none visible	some visible	all visible
Hot keywords	visible	not visible	not visible
Access method (guidance)	strict guided tour	expanded list	collapsed list
Presentation:			
Clarification icons	yes	yes	no

- *Links to fundamental information* include definitions, facts, principles, concepts, etc., e.g. links to simple introductory information, performance data, etc.
- *Links to deep, complex, and detailed information* include specifications, detailed descriptions, deep explanations, etc., e.g. links to specifications of parts or assemblies.
- *Hot keywords* are used to provide extra clarification to a piece of textual information, e.g. a reference link to the definition of a concept.
- *Access methods (guidance)* identify the level of guidance provided to the current user and the way hypermedia pages are accessed. They include guided tour, expanded list, and collapsed list.

Adaptive presentation support is achieved by adapting the presentation of a page accessed by a particular user to current user knowledge. It can be implemented using colours, icons, font sizes, etc. Frames and visual icons are used to provide extra clarification to the presentation of the hypermedia. A distinct visual icon is associated with every IO and navigational hyperlink in order to clarify its functional characteristic, i.e. fundamental, procedure, clarification, advice, or specification (see section 4.5).

The values associated with every benchmark are an interpretation of the main principles of the adaptive strategy outlined in Figure 5.12. The adaptive support strategy is implemented by transforming the benchmarks and their values into conditional semantic rules which are applied on relationships between IOs. These IOs are richly indexed in accordance with the semantic data model of the technical documentation described in chapters 3 and 4. The “purpose” and the “functional

characteristics” of IOs are used as arguments for the conditional relationships. Figure 5.13 shows an adaptive filtering mechanism for identifying relevant IOs based on conditional semantic rules applied to user knowledge. The filtering information space is divided into two domains namely, the *content domain* and the *navigation domain*, which physically correspond to the technical documentation metadata repository and the navigational relationships repository, respectively. In the figure, the main requested information is the procedural information represented by the composite organisational object IO₁. IO₁ is an organisation of a procedure, which contains the steps IO₁₁, IO₁₂, etc. The procedural step IO₁₁ is semantically related to the information fragments IO₁₁₁ and IO₁₁₂ which provide extra clarification and advice, respectively. Beyond the content domain is the navigation domain where IO₁ has outgoing referential relationships with “fundamental” and “specification” information fragments IO₂, and IO₃, respectively, and it is referenced by an in-coming referential relationship from IO₄. In addition, IO₁₁ has a keyword relationship with IO₂ and IO₃. With regard to the queried topic, these referential relationships provide related extra explanations and more detailed information that are available in the technical documentation.

The main objective of the adaptive filter is to distinguish the “relevant” information objects from irrelevant ones, depending on the user knowledge stereotypes, i.e. poor, sufficient, or outstanding. The relevant objects are then passed to be rendered and the irrelevant ones are discarded. In the example shown in Figure 5.13, the full list of the pre-filtered IOs includes IO₁, IO₂, IO₃, IO₄, and all their descendents, which include

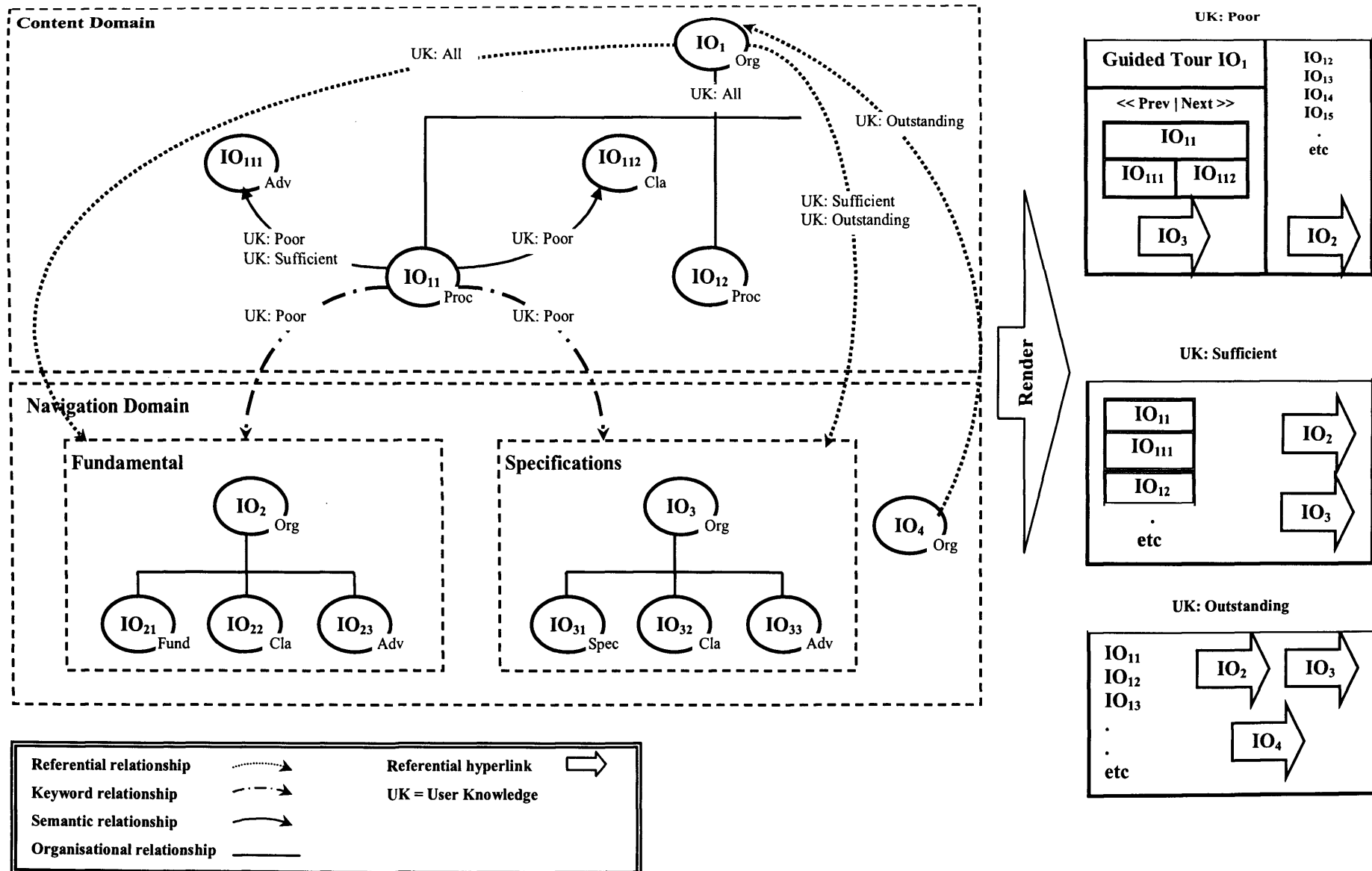


Figure 5.13 User Knowledge-Based Conditions for Identifying “Relevant” Information Objects

IO₁₁, IO₁₁₁, IO₁₁₂, IO₁₂, etc. The conditions shown on the relationship arcs, which are an interpretation of the adaptive strategy, will determine the selection criteria of these information objects. For instance, the case where “UK: poor” will invoke a subset of information objects to be rendered in the content of the main page, including IO₁, IO₁₁, IO₁₁₁, IO₁₁₂, IO₁₂, etc, together with the target IOs: IO₂, and IO₃ to be rendered as keyword hyperlinks. In addition, IO₂ will be rendered as a referential hyperlink. In contrast, in the case where “UK: outstanding”, only IO₁₁, IO₁₂, and IO₁₃, etc., will be selected for content rendering and IO₂, IO₃, and IO₄ will be rendered as referential hyperlinks.

Figure 5.13, moreover, depicts a conceptual outline of the presentation templates used to render the hypermedia pages. It shows the level of guidance given to the user and the navigational hyperlinks provided depending on the user knowledge of the subject. Through these conceptual outlines, the included/discarded information fragments and referential hyperlinks with respect to the user knowledge can be easily compared.

5.2.1.3 Adaptive Hypermedia Support for Diagnosis Information

The adaptive hypermedia system has been developed in order to assist in the retrieval of the faults diagnosis and correction information required by the users of the diagnostic ES. Figure 5.14 illustrates the adaptive hypermedia support through alternative views of the “check the servo brake” diagnosis procedure, provided to users with different knowledge stereotypes. It shows that different users with different knowledge stereotypes, requesting the same information, receive different types of content, navigation, and presentation. These hypermedia pages are generated on the

fly (as virtual documents) when requested by the user. The “topic ID” (5.9.9.1) is passed by the diagnostic ES and the user identifies his/her knowledge of the procedure by selecting one of the three stereotypes. The procedure “check the servo brake” has 15 steps where every step is associated with a procedural IO and may be associated with one or more clarification and/or advice IOs. The following is a comparative description of the three cases shown in Figure 5.14:

Case (1) “UK: poor”:

- The content IOs of every procedure including the clarification and advice IOs are visible (expanded).
- Hyperlinks are provided for fundamental information only, e.g. “PP: Servo Brake”, which is a hyperlink to the fundamental planning and performance data of the servo brake.
- Hot keywords hyperlinks are provided on the textual description of the steps of the procedure, e.g. “PP: Hand Brake”.
- The information is presented using a guided tour, which provides the novice user with maximum and strict guidance.
- Every IO and hyperlink is associated with an icon in order to clarify its content or destination, respectively.

Case (2) “UK: sufficient”:

- The procedural and the advice IOs are visible. The clarification IOs are invisible.

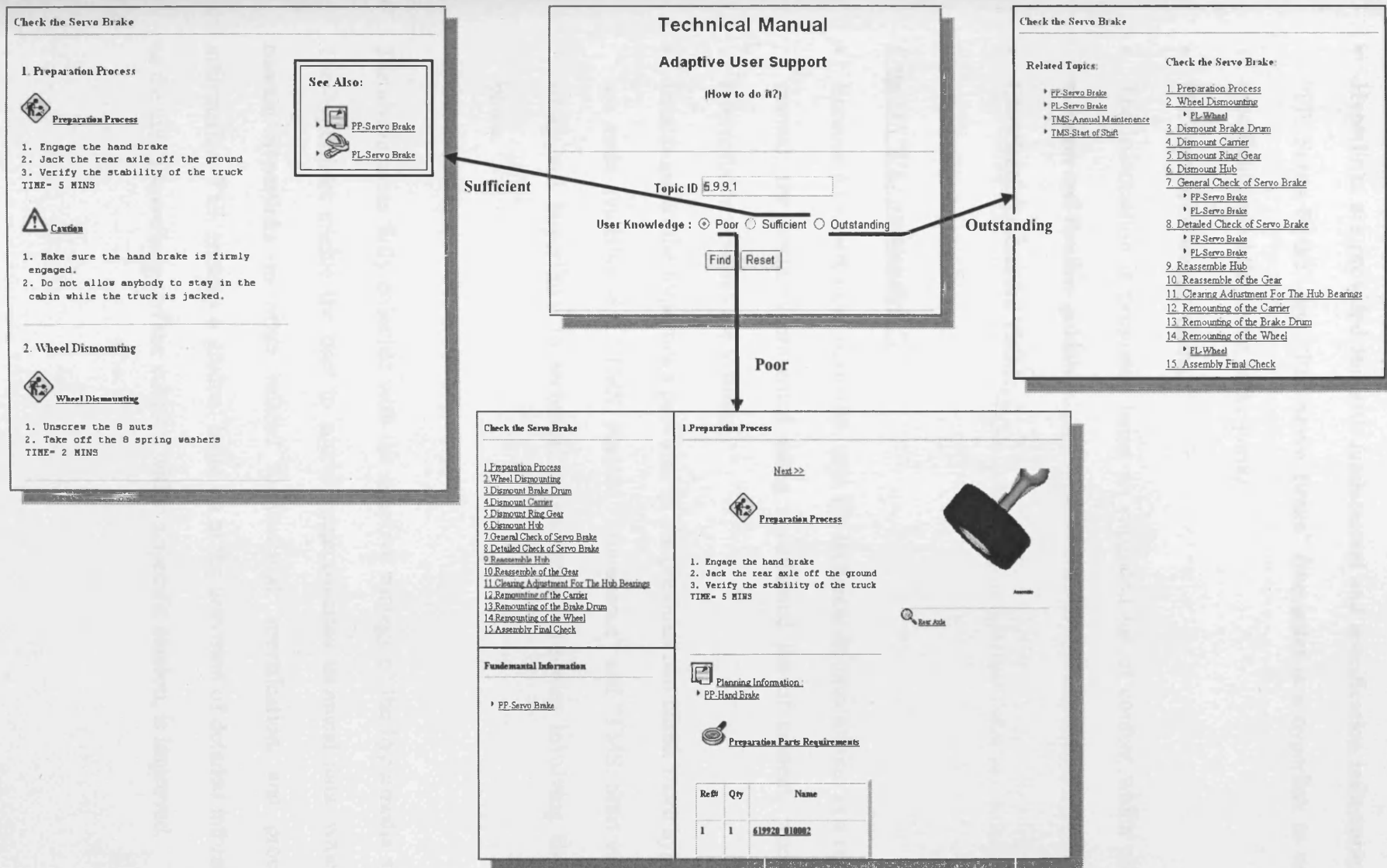


Figure 5.14 Alternative Views of “Check the Servo Brake” Diagnosis Procedure for Different User Knowledge Stereotypes

- Hyperlinks are provided for both fundamental and specification information, e.g. “PP: Servo Brake” and “PL: Servo Brake” (the latter is a hyperlink to detailed specifications of the servo brake parts).
- Hot keywords are not visible.
- The information is presented using an expanded list of content which provides medium and flexible guidance.
- Clarification icons are visible.

Case (3) “UK: outstanding”:

- Because an expert user, normally, uses the technical documentation as a reference manual, the result is presented using a collapsed list of content, where only referential hyperlinks are visible.
- In addition to the hyperlinks provided in the previous two cases, more hyperlinks are made available, e.g. “TMS: Annual Maintenance” and “TMS: Start of Shift”, which are hyperlinks to technical maintenance schedules involving the servo brake.

The above cases fully coincide with the adaptive strategy of the hypermedia system.

The hyperlinks enable the user to access subject-related technical data, which also contain hyperlinks to other related fundamental, specification, and procedural information. This enables a gradual build up in the provision of detailed information, as the user’s knowledge of the subject, within a specific session, is improved.

5.2.2 General Architecture for the Adaptive Hypermedia System

Figure 5.15 presents the general architecture of the adaptive hypermedia system used to implement the adaptive strategy. At the core of the system is the *Adaptive Hypermedia Generator* AHG, which consists of three components namely, the *Adaptive Support Engine* ASE, the *Search Utility* SU, and the *Adaptive Rendering Utility* ARU. The ASE is in the core of the AHG, which controls and synchronises the adaptive hypermedia generation process. It receives the user knowledge assessment (UK) and the topic identifier (T_ID) from the user through the information retrieval display. It passes the T_ID to the SU and receives the search result, and then it passes the search result with the UK to the ARU and receives the rendered HTML files and passes them to the user. The SU searches the technical documentation metadata repository for all the descendents of the topic identified by the ASE, and sends the resulting list of identifiers back to the ASE. The ARU receives a list of identifiers and executes a set of adaptive content, navigation, and presentation conditions in order to filter the search result. The ARU accesses the technical documentation metadata and navigational relationships repository for retrieving the required content and navigation information, and renders the relevant data into HTML files. It then sends these files to the ASE, which passes them to a special adaptive information display. The referential hyperlinks refer the user to more detailed hypermedia pages which are already available in the Web-based technical manual. The user uses a standard Web browser to identify the subject and an assessment of his/her knowledge of this subject. The ASE is a Java Servlet, which interacts with the Java-based classes SU and ARU.

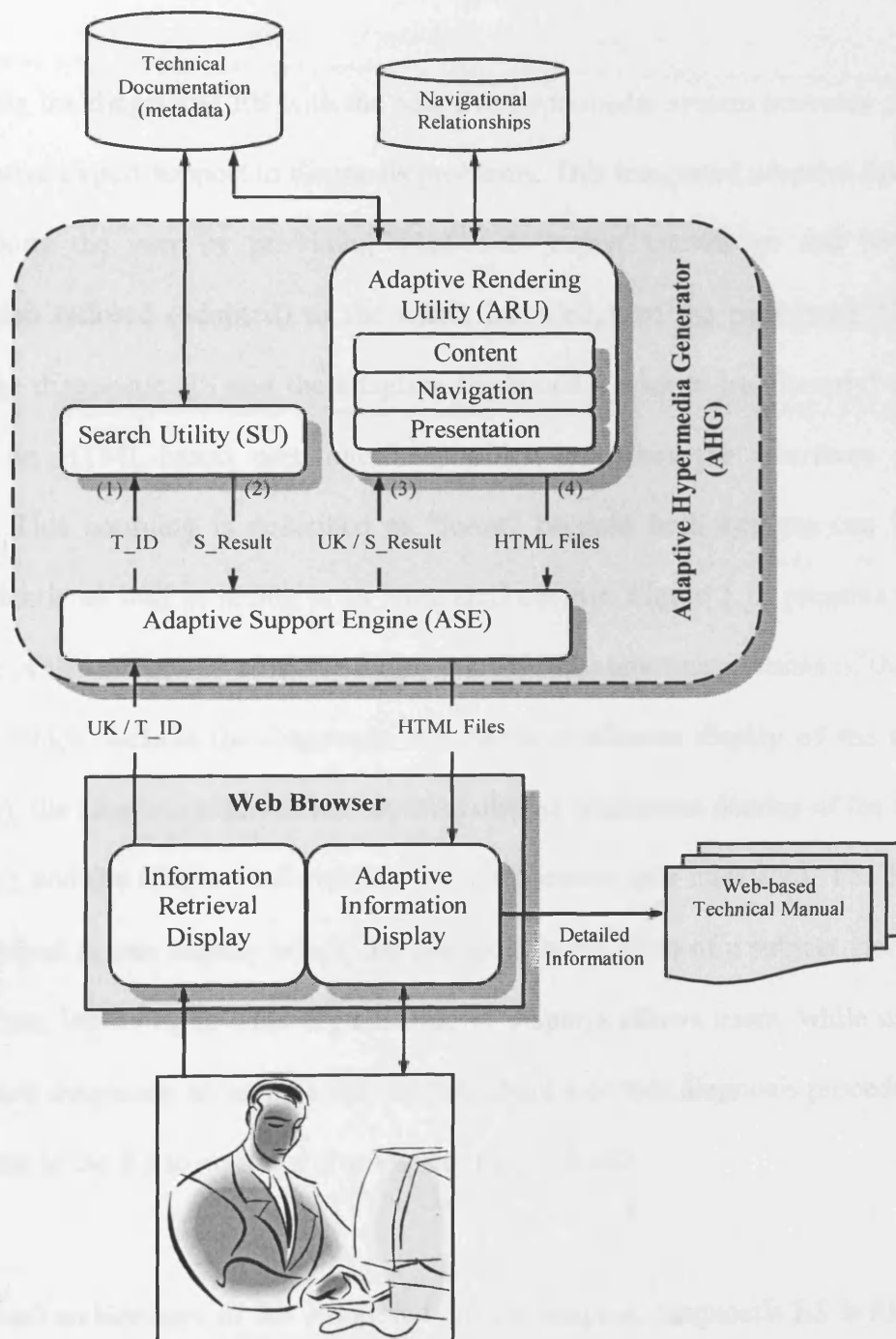
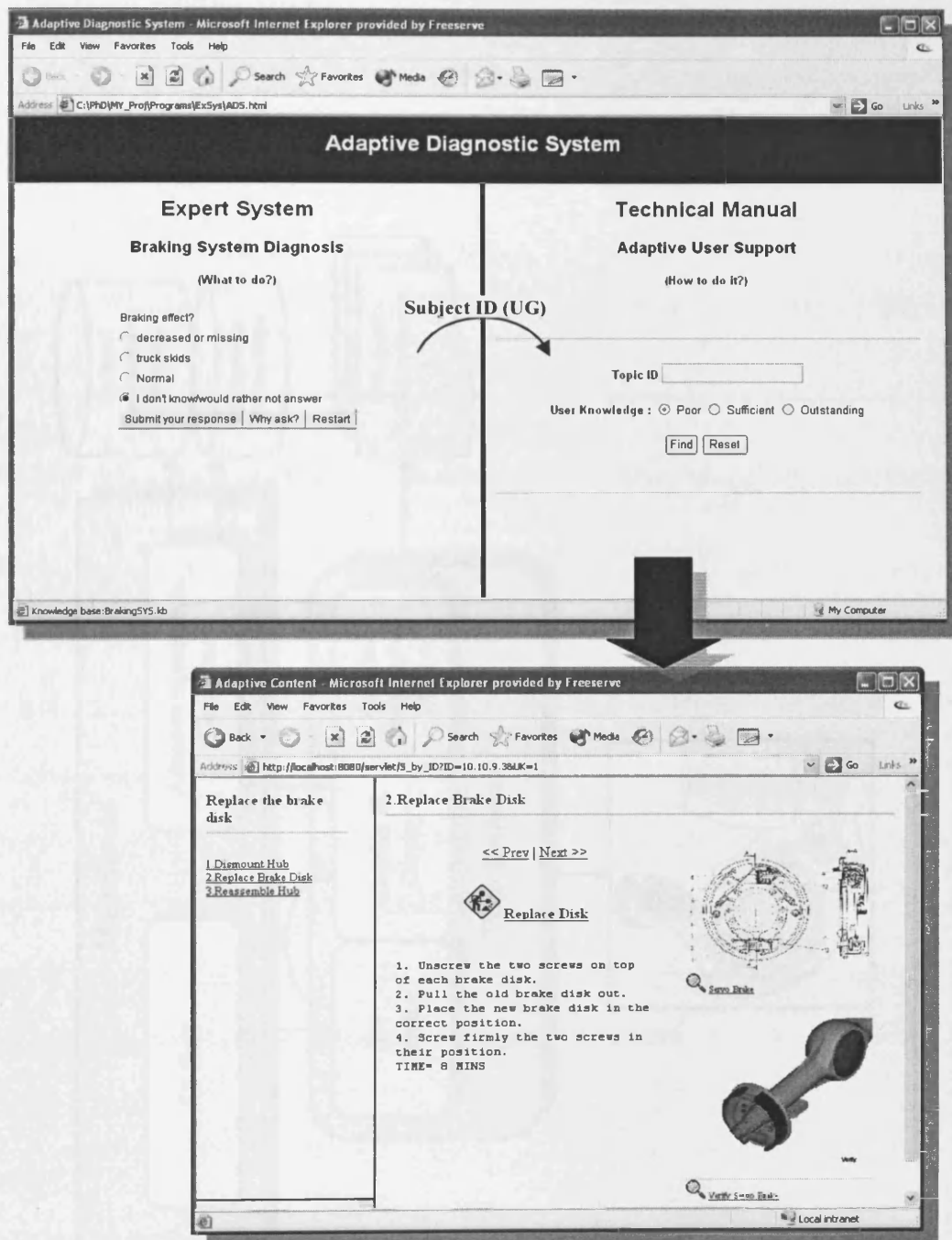


Figure 5.15 General Architecture for the Adaptive Hypermedia System

5.3 INTEGRATED ADAPTIVE DIAGNOSTIC EXPERT SYSTEM

Integrating the diagnostic ES with the adaptive hypermedia system provides effective and adaptive expert support to diagnosis problems. This integrated adaptive diagnostic ES supports the user by providing what-to-do expert knowledge and how-to-do information tailored (adapted) to the user's knowledge of the performed diagnosis tasks. The diagnostic ES and the adaptive hypermedia system are "loosely" coupled through an HTML-based user interface, which combines the interfaces of both systems. This coupling is described as "loose" because both systems can function independently as well as acting in an integrated manner. Figure 5.16 presents the user interface of the integrated adaptive diagnostic ES. The interface consists of three user displays which include the diagnostic ES display (leftmost display of the top user interface), the adaptive information retrieval display (rightmost display of the top user interface), and the adapted information display (bottom user interface). The first two are combined in one display where the user goal in the form of a subject identifier is passed from left to right. This organisation of displays allows users, while using the ES for fault diagnosis, to request information about a certain diagnosis procedure, and then return to the ES to continue from where they left off.

The general architecture of the integrated on-line adaptive diagnostic ES is illustrated in Figure 5.17. The Figure outlines the user interface which is accessible through a standard Web browser. It also shows the ES shell and the adaptive hypermedia generator mounted on the Web server together with the KB, technical documentation



UG = User Goal

Figure 5.16 User Interface for the Integrated Adaptive Diagnostic Expert System

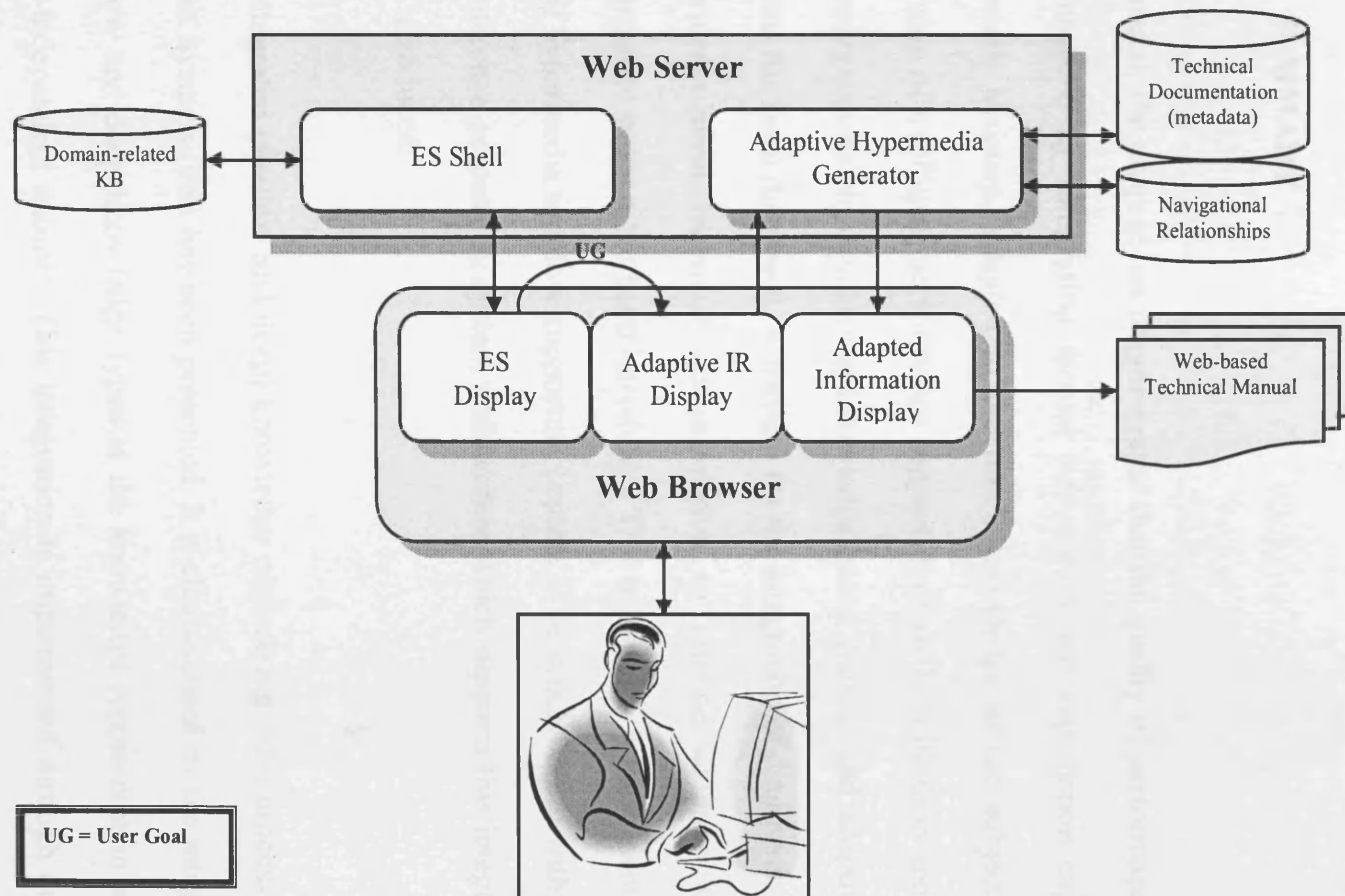


Figure 5.17 General Architecture of the Integrated On-Line Adaptive Expert Diagnostic System

metadata repository, and the navigational relationships repository. The Web server that has been employed is the Java Server Web Development Kit version 1.0.1 [JSWDK, 1.0.1], which contains a Web container that runs Java Servlets.

5.4 SUMMARY

In general, the chapter has demonstrated that the quality of performance support can be enhanced by integrating factual information and explanation capabilities with diagnostic assistant, adapted to the user's knowledge of the subject domain. The provision of intelligent performance support through the utilisation and integration of technologies used in developing knowledge-based systems and adaptive hypermedia systems has been discussed. A method for the integration of diagnostic assistance and hypermedia-based technical documentation to provide adaptive and intelligent performance support has been introduced. This integration is implemented through the use of hypermedia to allow supporting content to be synchronized with the diagnostic ES inference process. A system architecture which supports this integration has also been introduced.

An integrated (shallow and deep) knowledge engineering (KE) process for diagnostic Expert Systems (ES) has been presented. It is characterised by the integration of both shallow and deep knowledge types at the knowledge representation stage in an ES shell-independent manner. This integration is implemented through the inclusion of "deep knowledge" references in the KB of the ES. This process is supported by: (i) an integrated knowledge model which formally represents a fault in terms of its symptoms, causes, diagnosis procedure, and correction procedure, and (ii)

implementation techniques for automatically generating the rule-based KB in specific ES shell format, and the automatic update of deep knowledge data. Consequently, procedure or training manuals are referenced by the diagnosis steps presented by the diagnostic ES. In addition, glossaries or other support materials are linked to the progress of the diagnostic ES to integrate learning with problem solving. Moreover, a general architecture for supporting the KE process and implementing the diagnostic ES has been introduced.

Furthermore, a strategy for providing adaptive hypermedia support has been outlined. This strategy is implemented on top of the technical documentation semantic data model using conditional semantic rules. It relies on a stereotype model of the knowledge of the users. The strategy draws a relationship between the current user knowledge and the adaptive support features provided to the user. In addition, a general architecture for the adaptive hypermedia system has been introduced. This architecture has been used to implement the adaptive strategy of the hypermedia system.

The main contributions of the work reported in this chapter are, first, an integrated KE process for diagnostic ESs and an integrated knowledge model. Second, a strategy for retrieving hypermedia-based diagnosis information adapted to the knowledge of the user and operating in conjunction with the existing hypermedia-based technical documentation. Third, a system architecture for an integrated adaptive diagnostic ES for the Web.

CHAPTER 6

CONTRIBUTIONS, CONCLUSIONS AND FUTURE WORK

This chapter summarises the contributions made and conclusions reached and suggests possible directions for future research.

6.1 CONTRIBUTIONS

The main product of this research is a systematic methodology for the development of hypermedia-based Performance Support Systems (PSS) for the Web, which adheres to the main characteristics of advanced PSSs. These characteristics are outlined in a conceptual model that complies with state-of-the-art technologies and current practices in the field of user performance support. The work reported in this thesis is an attempt to apply integrated knowledge-based and adaptive hypermedia technologies in the area of electronic PSSs. Moreover, this work is a contribution in the direction of structured hypermedia authoring of technical documentation. It tackles the main challenges associated with the systematic development of hypermedia-based technical documentation for the Web which include design, authoring, and implementation, and the creation of supporting CASE tools. The specific contributions of this work are as follows:

1. *Conceptual model for advanced PSSs.*

Advanced PSSs are characterised by this work as mainly consisting of two interlinked and loosely coupled components that are designed and accessed in a task-based and

user-centred manner. The first component is a freely browsed technical documentation of the application domain which provides the user with “how-to-do” type of information. The second component is an expert advisor that provides assistance for more specific, complex, and difficult to learn tasks i.e. “what-to-do” type of information. The integrated technologies utilised in advanced PSSs include adaptive Web-based hypermedia and knowledge-based systems.

2. Usage-based data model for the design of technical documentation.

The proposed model abstracts the intended purpose of the documentation, the tasks supported by the documentation, and the functional characteristics of documents. These abstractions are integrated in a usage-based semantic network. Moreover, rules and valid relationships within the semantic network are identified. This design framework has been used in order to organise, generate, and maintain the technical documentation (authoring). In addition, this model has been employed to support a strategy for the adaptive retrieval of hypermedia-based technical documents.

3. Model-driven approach for authoring hypermedia-based technical documentation.

This approach utilises the usage-based data model for the design of technical documentation (outlined above). In addition, it complies with the principled guidelines of structured authoring. The original methods, techniques, and architectures that have been introduced to assist in the practical implementation of this authoring approach include:

- ***Method for building an implementation language-independent structure for the hypermedia-based technical documentation.*** This method is based on a

semantically enhanced version of the system-based structuring method. The implementation of this method has required the introduction of novel techniques. These include (i) a technique for indexing the information objects (IOs) of the technical documentation, (ii) a technique for generating dynamic identification codes for IOs that maintains their structural and semantic properties, and (iii) a technique for the automatic generation of hypermedia pages. In order to demonstrate that the built structure is implementation language-independent, the technical documentation have been generated using two of the most widely used Web-based mark-up languages, namely, HTML and XML.

- ***Methods for the automatic identification of semantically-based navigational relationships.*** Two different methods for the automatic identification of navigational relationships between IOs based on their semantics have been presented. These methods include (i) a method for generating context-driven navigational relationships, and (ii) a method for generating purpose-driven navigational relationships. These relationships have been automatically converted into physical hyperlinks by the run-time hyperlink generator.
- ***Method for the presentation of hypermedia pages.*** This method uses a combination of frame-based presentation templates, icons, and colours. The presentation templates used to create the user interface are based on HTML frames, which divide the interface window into several navigation-able areas. Presentation icons are used to represent the type of function performed by every IO in a visual manner. Different background colours are associated with

different information categories in order to enable users to determine the type of the delivered information in a visual manner, and hence, minimise disorientation.

- ***System architecture for implementing the structured authoring approach.***

New system architecture has been created to support the structured authoring approach of Web-based technical documentation. The support is realised through a set of specially developed Java-based CASE tools and code generators. The architecture distinguishes two environments, namely, off-line and on-line. The former environment comprises specially developed: (i) authoring tools which include the *Structure Builder* and the *Navigational Relationships Generator*, and (ii) implementation tools which include the *Web Pages Generator* and the *Presentation Templates Generator*. The latter mainly comprises specially developed run-time tool namely the *Navigational Hyperlinks Generator*.

4. Methodology for providing intelligent diagnosis support through knowledge-based expert systems.

The original methods, techniques, and architectures that have been introduced to assist in the practical implementation of this methodology include:

- ***Integrated (shallow and deep) knowledge engineering (KE) process for diagnostic Expert Systems (ES).*** This KE process is the core of the methodology. It is characterised by the integration of both shallow and deep knowledge types at the knowledge representation stage in an ES shell-independent manner. This integration is implemented through the inclusion of

“deep knowledge” references in the KB of the ES. These references point towards the associated deep knowledge fragments of the technical documentation metadata which represent an “existing” hypermedia-based technical manual. This process is supported by: (i) an integrated knowledge model which formally represents a fault in terms of its symptoms, causes, diagnosis procedure, and correction procedure, and (ii) implementation techniques for automatically generating the rule-based Knowledge Base (KB) in specific ES shell format, and the automatic update of deep knowledge data.

- ***General architecture for the diagnostic ES.*** The main components in this architecture are the *KB Builder*, the *KB Generator*, and the *ES Shell*. It also comprises two types of interfaces namely (i) the domain expert interface which facilitates the building and updating of the KB through a Graphical User Interface (GUI), and (ii) the end-user interface through a standard Web browser.

5. Methodology for the adaptive retrieval of diagnosis information using conditional semantic rules

The original strategy and architecture that have been introduced to assist in the practical implementation of this methodology include:

- ***Strategy for providing adaptive hypermedia support.*** This strategy is implemented on top of the technical documentation semantic data model discussed earlier, and it is based on a stereotype model of the knowledge of the users. The strategy draws a relationship between the current user knowledge and the adaptive support features provided to the user using conditional semantic rules.

- ***General architecture for the adaptive hypermedia system.*** This architecture has been used to implement the adaptive strategy of the hypermedia system. The core of this architecture is the *adaptive hypermedia generator*, which consists of three components namely, the *adaptive support engine*, the *search utility*, and the *adaptive rendering utility*. Users interact with the adaptive system using two types of displays through a standard Web browser. These are (i) the information retrieval display, and (ii) the adaptive information display.

6. Architecture for integrating the diagnostic ES and the adaptive hypermedia documentation system.

This architecture integrates the architectures of the diagnostic ES and the adaptive hypermedia system (both outlined earlier). This integration is facilitated through an HTML-based user interface, which combines the interfaces of both systems. The special organisation of the user interface displays allows users, while using the ES for fault diagnosis, to request information about a certain diagnosis procedure, and then return to the ES to continue from where they left off. The integrated adaptive diagnostic ES supports the user by providing what-to-do expert knowledge and how-to-do information tailored (adapted) to the declared user knowledge of the subject domain.

6.2 CONCLUSIONS

- The main objective of an advanced PSS is to provide operators with “how-to-do” and “what-to-do” types of information. The former is achieved through a freely browsed technical documentation. The latter is achieved through an expert advisor

that provides assistance for more specific, complex, and difficult to learn tasks.

These services provide a better support when they are interlinked and are designed to be accessed in a task-based and user-centred manner.

- Advanced PSSs are achievable through an integrated technological solution which includes Web-based hypermedia, knowledge-based systems, and adaptive hypermedia.
- Traditional hypermedia authoring approaches are complex and knowledge intensive activities that suffer from many deficiencies. These deficiencies greatly affect the suitability of these approaches to handle data-intensive applications.
- The development of data-intensive hypermedia applications needs to be organised into a well-defined process and to utilise structured design methods amenable to the benefits of software engineering.
- It is important for technical authors to categorise the information used in the documentation in a way that closely reflect their subsequent usage. A usage-based design approach should consider the intended purpose of the documentation, the tasks supported by the documentation, and the functional characteristics of documents.
- Structured authoring of technical documentation must be supported by an abstract data model. This model can be used to adapt the delivered technical information to the user's knowledge of the material.
- Information objects (IO) are the smallest units of publishable information that are created and modified by hypermedia authors. These IOs should be as small as possible, in order to ensure high flexibility, and at the same time they must be large enough to stand alone as part of a topic and/or to be reused in another.

- Usually, there exist a close relationship between planning, support, and action types of technical information. These relationships should be identified and extracted during the hypermedia authoring phase, and they should be converted into physical hyperlinks at the implementation phase.
- The use of semantically-based frames, visual icons, and colouring schemes can significantly improve the presentation of the technical documentation.
- In PSSs, “how-to-do” and “what-to-do” types of information complement each other, and their integration provides effective knowledgeable support for users’ performance. These types of information can be integrated at the representation stage of a knowledge engineering process.
- Delivering hypermedia-based technical information in an adaptive manner can reduce information disorientation and cognitive overload by providing user- and task-relevant information and discarding irrelevant ones. This eliminates confusion and speeds up the learning process.
- The quality of performance support can be substantially enhanced by integrating factual information and explanation capabilities with a knowledge-based expert assistance, adapted to the user’s knowledge of the performed task.

6.3 FUTURE RESEARCH

A possible future research direction is the introduction of new attributes to enhance the indexing mechanism of the hypermedia objects. For example, the “level of detail” can be used as a measure of content complexity for textual documents. In addition, presentation-related attributes such as the level of conveyed attractiveness,

abstraction, spatiality, and temporality can be used to classify non-textual multimedia documents. These attributes would improve the performance of the filtering process of the adaptive user support system.

The indexing of the technical documents can be further enhanced by utilising AI techniques. For example, Natural Language Recognition techniques can be used with textual documents, and Image Processing techniques with graphics. These techniques can be used to automatically identify the semantic properties of hypermedia IOs. This would save time and effort and would simplify the authoring process.

Another useful research direction is the integration of semantic domain data models with AI techniques to support the navigation of hypermedia-based technical documentation. For example, Artificial Neural Networks (ANN) and Fuzzy Logic (FL) systems can be used to recommend the best next hypermedia object(s) to browse. The former can represent experience, behaviour, and decisions of experts. The latter can represent the learning strategy of the author. In addition, machine learning techniques (e.g. data mining, clustering, etc.) can be used to automatically improve the organisation of the Web-based technical documentation by learning from visitors access patterns and detecting navigational trends.

Finally, with regard to user modelling, further research can be conducted on investigating the application of implicit user models, i.e. those automatically detected by the system, in providing adaptive diagnosis support. In addition, a hybrid approach to user modelling which combines a stereotype and an overlay user model to represent

the user knowledge of the technical documentation, might improve the accuracy of the adaptively delivered information.

APPENDIX A

INFORMATION OBJECTS METADATA

Table A.1 illustrates the fully processed set of Information Objects (IO) and their metadata. It shows the full set of attributes and their associated values. The attributes are classified as either of type definition, semantic, structure, presentation, or navigation. In addition, attributes are also classified as system-set or author-set. The attributes *name* and *ID* are system-set. The remainder are author-set attributes.

Table A.1 Information Objects and Metadata

ID	Name	Description	ICat	Task	Function	Container	Seq	Location	Form	ConForm
1.1.9.1	f_1.1.9.1	Fork-Lift Truck	1	1	9	1.1	1	../ipm/	0	3
1.1.9.1.1.1	f_1.1.9.1.1.1	Introduction	1	1	1	1.1.9.1	1	../ipm/	1	0
1.1.9.1.3.2	f_1.1.9.1.3.2	General Truck Picture	1	1	3	1.1.9.1	2	../ipm/pictures/truck.gif	2	0
1.1.9.1.3.3	f_1.1.9.1.3.3	Truck Video	1	1	3	1.1.9.1	3	../ipm/videos/artic_x0003.avi	3	0
1.1.9.1.4.4	f_1.1.9.1.4.4	Important Attention !!	1	1	4	1.1.9.1	4	../ipm/	1	0
1.1.9.2	f_1.1.9.2	Operational Data	1	1	9	1.1	2	../ipm/	0	3
1.1.9.2.1.1	f_1.1.9.2.1.1	General Performance	1	1	1	1.1.9.2	1	../ipm/	1	0
1.1.9.3	f_1.1.9.3	Attachments	1	1	9	1.1	3	../ipm/	0	3
1.1.9.3.1.1	f_1.1.9.3.1.1	Possible Settings	1	1	1	1.1.9.3	1	../ipm/pictures/Attachments.jpg	2	0
1.3.9.1	f_1.3.9.1	Fork-Lift Truck	1	3	9	1.3	1	../ipm/	0	3
1.3.9.1.1.1	f_1.3.9.1.1.1	General Performance and Dimensions	1	3	1	1.3.9.1	1	../ipm/	1	0
1.3.9.1.1.2	f_1.3.9.1.1.2	Back View	1	3	1	1.3.9.1	2	../ipm/pictures/artic1.gif	2	0
1.3.9.1.1.3	f_1.3.9.1.1.3	Side View	1	3	1	1.3.9.1	3	../ipm/pictures/artic2.gif	2	0
1.3.9.1.1.4	f_1.3.9.1.1.4	Truck Back View	1	3	1	1.3.9.1	4	../ipm/videos/artic_x0003.avi	3	0
1.3.9.1.1.5	f_1.3.9.1.1.5	Truck Side View	1	3	1	1.3.9.1	5	../ipm/videos/artic_y0003.avi	3	0
1.3.9.2	f_1.3.9.2	Braking System	1	3	9	1.3	2	../ipm/	0	1
1.3.9.2.1.1	f_1.3.9.2.1.1	Servo Brake	1	3	1	1.3.9.2	1	../ipm/	1	0
1.3.9.2.1.2	f_1.3.9.2.1.2	Hand Brake	1	3	1	1.3.9.2	2	../ipm/	1	0
1.3.9.2.1.3	f_1.3.9.2.1.3	Brake Fluid Pipes	1	3	1	1.3.9.2	3	../ipm/	1	0
1.3.9.2.1.4	f_1.3.9.2.1.4	Brake Pedal	1	3	1	1.3.9.2	4	../ipm/	1	0
1.3.9.2.1.5	f_1.3.9.2.1.5	Brake Fluid	1	3	1	1.3.9.2	5	../ipm/	1	0
1.3.9.3	f_1.3.9.3	Lifting System	1	3	9	1.3	3	../ipm/	0	3
1.3.9.3.1.1	f_1.3.9.3.1.1	Lifting Gear	1	3	1	1.3.9.3	1	../ipm/	1	0
1.3.9.4	f_1.3.9.4	Drive Axle	1	3	9	1.3	4	../ipm/	0	3

1.3.9.4.1.1	f_1.3.9.4.1.1	Axle - General	1	3	1	1.3.9.4	1 ../ipm/	1	0
1.3.9.5	f_1.3.9.5	Engine	1	3	9	1.3	5 ../ipm/	0	3
1.3.9.5.1.1	f_1.3.9.8.1.1	Engine - General	1	3	1	1.3.9.5	1 ../ipm/	1	0
1.3.9.6	f_1.3.9.6	Steering system	1	3	9	1.3	6 ../ipm/	0	3
1.3.9.6.1.1	f_1.3.9.9.1.1	Steering - General	1	3	1	1.3.9.6	1 ../ipm/	1	0
1.3.9.7	f_1.3.9.7	Hydraulic system	1	3	9	1.3	7 ../ipm/	0	3
1.3.9.7.1.1	f_1.3.9.10.1.1	Hydraulic - General	1	3	1	1.3.9.7	1 ../ipm/	1	0
1.3.9.8	f_1.3.9.8	Electrical system	1	3	9	1.3	8 ../ipm/	0	3
1.3.9.8.1.1	f_1.3.9.11.1.1	Electrical system - General	1	3	1	1.3.9.8	1 ../ipm/	1	0
2.3.9.1	f_2.3.9.1	Fork-Lift Truck	2	3	9	2.3	1 ../ipm/	0	3
2.3.9.1.5.1	f_2.3.9.1.5.1	General View	2	3	5	2.3.9.1	1 ../ipm/pictures/PL_G_View.gif	2	0
2.3.9.2	f_2.3.9.2	Braking System	2	3	9	2.3	2 ../ipm/	0	1
2.3.9.2.9.1	f_2.3.9.2.9.1	Braking System - Complete	2	3	9	2.3.9.2	1 ../ipm/	0	3
2.3.9.2.9.1.5.1	f_2.3.9.2.9.1.5.1	Braking System 1547475	2	3	5	2.3.9.2.9.1	1 ../ipm/pictures/PL_B_System.gif	2	0
2.3.9.2.9.1.5.2	f_2.3.9.2.9.1.5.2	Braking System Parts	2	3	5	2.3.9.2.9.1	2 ../ipm/	1	0
2.3.9.2.9.2	f_2.3.9.2.9.2	Hand Brake	2	3	9	2.3.9.2	2 ../ipm/	0	3
2.3.9.2.9.2.5.1	f_2.3.9.2.9.2.5.1	Hand Brake 1416533	2	3	5	2.3.9.2.9.2	1 ../ipm/pictures/H_Brake.gif	2	0
2.3.9.2.9.2.5.2	f_2.3.9.2.9.2.5.2	Hand Brake - Complete	2	3	5	2.3.9.2.9.2	2 ../ipm/	1	0
2.3.9.2.9.2.5.3	f_2.3.9.2.9.2.5.3	Hand Brake as part of the Braking System	2	3	5	2.3.9.2.9.2	3 ../ipm/pictures/b_sys1.gif	2	0
2.3.9.2.9.3	f_2.3.9.2.9.3	Brake Fluid Pipes	2	3	9	2.3.9.2	3 ../ipm/	0	3
2.3.9.2.9.3.5.1	f_2.3.9.2.9.3.5.1	Pipes	2	3	5	2.3.9.2.9.3	1 ../ipm/	1	0
2.3.9.2.9.3.5.2	f_2.3.9.2.9.3.5.2	Pipes - Technical View	2	3	5	2.3.9.2.9.3	2 ../ipm/pictures/BF_Pipes.gif	2	0
2.3.9.2.9.4	f_2.3.9.2.9.4	Brake Pedal	2	3	9	2.3.9.2	4 ../ipm/	0	3
2.3.9.2.9.4.3.2	f_2.3.9.2.9.4.3.2	Brake Pedal - Overall	2	3	3	2.3.9.2.9.4	2 ../ipm/pictures/pedal_ov.gif	2	0
2.3.9.2.9.4.5.1	f_2.3.9.2.9.4.5.1	Brake Pedal - Parts	2	3	5	2.3.9.2.9.4	1 ../ipm/	1	0
2.3.9.2.9.4.5.3	f_2.3.9.2.9.4.5.3	Brake Pedal - Parts View	2	3	5	2.3.9.2.9.4	3 ../ipm/pictures/pedal_p.gif	2	0
2.3.9.2.9.5	f_2.3.9.4.9.5	Servo Brake	2	3	9	2.3.9.2	5 ../ipm/	0	3
2.3.9.2.9.5.5.1	f_2.3.9.4.9.5.5.1	Servo Brake 1431442 - Technical View	2	3	5	2.3.9.2.9.5	1 ../ipm/pictures/S_Brake.Gif	2	0
2.3.9.2.9.5.5.2	f_2.3.9.4.9.5.5.2	Servo Brake 1431442 - Parts	2	3	5	2.3.9.2.9.5	2 ../ipm/	1	0

2.3.9.2.9.6	f_2.3.9.2.9.6	Brake Fluid Reservoir	2	3	9	2.3.9.2	6	../ipm/	0	3
2.3.9.2.9.6.5.1	f_2.3.9.2.9.6.5.1	Brake Fluid Reservoir - Parts	2	3	5	2.3.9.2.9.6	1	../ipm/	1	0
2.3.9.2.9.6.5.2	f_2.3.9.2.9.6.5.2	Brake Fluid Reservoir - Technical View	2	3	5	2.3.9.2.9.6	2	../ipm/pictures/b_sys1.gif	2	0
2.3.9.3	f_2.3.9.3	Lifting System	2	3	9	2.3	3	../ipm/	0	2
2.3.9.3.9.1	f_2.3.9.3.9.1	Lifting System - Complete	2	3	9	2.3.9.3	1	../ipm/	0	3
2.3.9.3.9.1.5.1	f_2.3.9.3.9.1.5.1	Lifting System 1547121	2	3	5	2.3.9.3.9.1	1	../ipm/pictures/L_Sys.gif	2	0
2.3.9.3.9.1.5.2	f_2.3.9.3.9.1.5.2	Lifting System Parts	2	3	5	2.3.9.3.9.1	2	../ipm/	1	0
2.3.9.3.9.2	f_2.3.9.3.9.2	Hydraulic Lifting System	2	3	9	2.3.9.3	2	../ipm/	0	3
2.3.9.3.9.2.5.1	f_2.3.9.3.9.2.5.1	Hydraulic Cylinder 1423430	2	3	5	2.3.9.3.9.2	1	../ipm/pictures/H_Sys.gif	2	0
2.3.9.3.9.2.5.2	f_2.3.9.3.9.2.5.2	Parts for Hydraulic Cylinder 1423430	2	3	5	2.3.9.3.9.2	2	../ipm/	1	0
2.3.9.3.9.2.5.3	f_2.3.9.3.9.2.5.3	Hydraulic Cylinder 1430511	2	3	5	2.3.9.3.9.2	3	../ipm/pictures/H_Sys2.gif	2	0
2.3.9.3.9.2.5.4	f_2.3.9.3.9.2.5.4	Parts for Hydraulic Cylinder 1430511	2	3	5	2.3.9.3.9.2	4	../ipm/	1	0
2.3.9.3.9.3	f_2.3.9.3.9.3	Fork Arm	2	3	9	2.3.9.3	3	../ipm/	0	3
2.3.9.3.9.3.5.1	f_2.3.9.3.9.3.5.1	Fork Arm 1430523	2	3	5	2.3.9.3.9.3	1	../ipm/pictures/F_Arm.gif	2	0
2.3.9.3.9.3.5.2	f_2.3.9.3.9.3.5.2	Parts for Fork Arm 1430523	2	3	5	2.3.9.3.9.3	2	../ipm/	1	0
2.3.9.4	f_2.3.9.4	Drive Axle	2	3	9	2.3	4	../ipm/	0	2
2.3.9.4.9.1	f_2.3.9.4.9.1	Front Drive Axle	2	3	9	2.3.9.4	1	../ipm/	0	3
2.3.9.4.9.1.5.1	f_2.3.9.4.9.1.5.1	Front Drive Axle 1425505	2	3	5	2.3.9.4.9.1	1	../ipm/	1	0
2.3.9.4.9.1.5.2	f_2.3.9.4.9.1.5.2	Parts for Front Drive Axle	2	3	5	2.3.9.4.9.1	2	../ipm/pictures/F_Axle.gif	2	0
2.3.9.4.9.2	f_2.3.9.4.9.2	Rear Drive Axle	2	3	9	2.3.9.4	2	../ipm/	0	3
2.3.9.4.9.2.5.1	f_2.3.9.4.9.2.5.1	Driving Axle 1431112	2	3	5	2.3.9.4.9.2	1	../ipm/pictures/R_Daxle.gif	2	0
2.3.9.4.9.2.5.2	f_2.3.9.4.9.2.5.2	Parts for Driving Axle 1431112	2	3	5	2.3.9.4.9.2	2	../ipm/	1	0
2.3.9.4.9.3	f_2.3.9.4.9.3	Wheel	2	3	9	2.3.9.4	3	../ipm/	0	3
2.3.9.4.9.3.5.1	f_2.3.9.4.9.3.5.1	Wheel Parts 1459781	2	3	5	2.3.9.4.9.3	1	../ipm/	1	0
2.3.9.4.9.3.5.2	f_2.3.9.4.9.3.5.2	Wheel Parts	2	3	5	2.3.9.4.9.3	2	../ipm/pictures/Wheel_parts.gif	2	0
2.3.9.4.9.4	f_2.3.9.4.9.4	Gear Box	2	3	9	2.3.9.4	4	../ipm/	0	3
2.3.9.4.9.4.5.1	f_2.3.9.4.9.4.5.1	Gear Box 1483448	2	3	5	2.3.9.4.9.4	1	../ipm/pictures/G_Box.gif	2	0
2.3.9.4.9.4.5.2	f_2.3.9.4.9.4.5.2	Parts for Gear Box 1483448	2	3	5	2.3.9.4.9.4	2	../ipm/	1	0
2.3.9.5	f_2.3.9.5	Engine	2	3	9	2.3	5	../ipm/	0	3

2.3.9.5.5.1	f_2.3.9.5.5.1	Engine - General View	2	3	5	2.3.9.5	1 ../ipm/pictures/Engine.gif	2	0
2.3.9.6	f_2.3.9.6	Steering system	2	3	9	2.3	6 ../ipm/	0	3
2.3.9.6.5.1	f_2.3.9.6.5.1	Steering system - Parts	2	3	5	2.3.9.6	1 ../ipm/	1	0
2.3.9.6.5.2	f_2.3.9.6.5.2	Steering System - Complete	2	3	5	2.3.9.6	2 ../ipm/pictures/Steering_S.gif	2	0
2.3.9.7	f_2.3.9.7	Hydraulic system	2	3	9	2.3	7 ../ipm/	0	3
2.3.9.7.5.1	f_2.3.9.7.5.1	Hydraulic System - Parts	2	3	5	2.3.9.7	1 ../ipm/	1	0
2.3.9.7.5.2	f_2.3.9.7.5.2	Hydraulic System - Complete	2	3	5	2.3.9.7	2 ../ipm/pictures/H_Sys.gif	2	0
2.3.9.8	f_2.3.9.8	Electrical system	2	3	9	2.3	8 ../ipm/	0	3
2.3.9.8.5.1	f_2.3.9.8.5.1	Electrical System - Parts	2	3	5	2.3.9.8	1 ../ipm/	1	0
2.3.9.8.5.2	f_2.3.9.8.5.2	Electrical System - Complete	2	3	5	2.3.9.8	2 ../ipm/pictures/E_Sys.gif	2	0
3.3.9.1	f_3.3.9.1	Fork-Lift Truck	3	3	9	3.3	1 ../ipm/	0	3
3.3.9.1.5.1	f_3.3.9.1.5.1	Fork-Lift Truck - Description	3	3	5	3.3.9.1	1 ../ipm/	1	0
3.3.9.1.5.2	f_3.3.9.1.5.2	Cabin	3	3	5	3.3.9.1	2 ../ipm/pictures/Cabin.gif	2	0
3.3.9.2	f_3.3.9.2	Braking System	3	3	9	3.3	2 ../ipm/	0	3
3.3.9.2.5.1	f_3.3.9.2.5.1	Braking System - Technical View	3	3	5	3.3.9.2	1 ../ipm/	1	0
3.3.9.2.5.2	f_3.3.9.2.5.2	Braking System - Technical View	3	3	5	3.3.9.2	2 ../ipm/pictures/TD_B_SYS.gif	2	0
3.3.9.2.5.3	f_3.3.9.2.5.3	Brake Fluid	3	3	5	3.3.9.2	3 ../ipm/	1	0
3.3.9.3	f_3.3.9.3	Lifting System	3	3	9	3.3	3 ../ipm/	0	3
3.3.9.3.5.1	f_3.3.9.3.5.1	Lifting Gear - Technical View	3	3	5	3.3.9.3	1 ../ipm/	1	0
3.3.9.4	f_3.3.9.4	Drive Axle	3	3	9	3.3	4 ../ipm/	0	3
3.3.9.4.5.1	f_3.3.9.4.5.1	Drive Axle - Technical View	3	3	5	3.3.9.4	1 ../ipm/	1	0
3.3.9.4.5.2	f_3.3.9.4.5.2	Drive Axle - Technical View	3	3	5	3.3.9.4	2 ../ipm/pictures/TD_D_Axle.gif	2	0
3.3.9.4.5.3	f_3.3.9.4.5.3	Chassis	3	3	5	3.3.9.4	3 ../ipm/	1	0
3.3.9.4.5.4	f_3.3.9.4.5.4	Chassis Description	3	3	5	3.3.9.4	4 ../ipm/pictures/Chassis.gif	2	0
3.3.9.5	f_3.3.9.5	Engine	3	3	9	3.3	5 ../ipm/	0	3
3.3.9.5.5.1	f_3.3.9.5.5.1	Engine Technical View	3	3	5	3.3.9.5	1 ../ipm/	1	0
3.3.9.6	f_3.3.9.6	Steering system	3	3	9	3.3	6 ../ipm/	0	3
3.3.9.6.5.1	f_3.3.9.6.5.1	Steering System -Technical View	3	3	5	3.3.9.6	1 ../ipm/	1	0
3.3.9.7	f_3.3.9.7	Hydraulic system	3	3	9	3.3	7 ../ipm/	0	3

3.3.9.7.5.1	f_3.3.9.7.5.1	Hydraulic System - Technical View	3	3	5	3.3.9.7	1 ../ipm/	1	0
3.3.9.8	f_3.3.9.8	Electrical system	3	3	9	3.3	8 ../ipm/	0	3
3.3.9.8.5.1	f_3.3.9.8.5.1	Electrical system - Technical View	3	3	5	3.3.9.8	1 ../ipm/	1	0
4.5.2.1	f_4.5.2.1	Truck Acceptance	4	5	2	4.5	1 ../ipm/	1	0
4.5.2.2	f_4.5.2.2	Running the New Truck	4	5	2	4.5	2 ../ipm/	1	0
4.6.2.1	f_4.6.2.1	Starting the Engine	4	6	2	4.6	1 ../ipm/	1	0
4.6.2.2	f_4.6.2.2	Shutting the Engine	4	6	2	4.6	2 ../ipm/	1	0
4.7.2.1	f_4.7.2.1	Moving and Accelerating	4	7	2	4.7	1 ../ipm/	1	0
4.7.2.2	f_4.7.2.2	Decelerating and Stopping	4	7	2	4.7	2 ../ipm/	1	0
4.7.2.3	f_4.7.2.3	Reverse	4	7	2	4.7	3 ../ipm/	1	0
4.7.2.4	f_4.7.2.4	Parking	4	7	2	4.7	4 ../ipm/	1	0
4.7.2.5	f_4.7.2.5	Picking Loads	4	7	2	4.7	5 ../ipm/	1	0
4.7.2.6	f_4.7.2.6	Unloading the Truck	4	7	2	4.7	6 ../ipm/	1	0
4.7.2.7	f_4.7.2.7	Working on Sites with Uneven Surfaces	4	7	2	4.7	7 ../ipm/	1	0
4.7.2.8	f_4.7.2.8	Unloading/Loading of the Truck from/on	4	7	2	4.7	8 ../ipm/	1	0
5.9.9.1	f_5.9.9.1	Check the Servo Brake	5	9	9	5.9	1 ../ipm/	0	2
5.9.9.1.9.1	f_5.9.9.1.9.1	Preparation Process	5	9	9	5.9.9.1	1 ../ipm/	0	3
5.9.9.1.9.1.2.1	f_5.9.9.1.9.1.2.1	Preparation Process	5	9	2	5.9.9.1.9.1	1 ../ipm/	1	0
5.9.9.1.9.1.3.2	f_5.9.9.1.9.1.3.2	Rear Axle	5	9	3	5.9.9.1.9.1	2 ../ipm/pictures/step_001.gif	2	0
5.9.9.1.9.1.3.3	f_5.9.9.1.9.1.4.3	Preparation Parts Requirements	5	9	3	5.9.9.1.9.1	3 ../ipm/	1	0
5.9.9.1.9.1.4.4	f_5.9.9.1.9.1.4.4	Caution	5	9	4	5.9.9.1.9.1	4 ../ipm/	1	0
5.9.9.1.9.10	f_5.9.9.1.9.10	Reassemble of the Gear	5	9	9	5.9.9.1	10 ../ipm/	0	3
5.9.9.1.9.10.2.1	f_5.9.9.1.9.10.2.1	Gear Reassembly	5	9	2	5.9.9.1.9.10	1 ../ipm/	1	0
5.9.9.1.9.10.3.2	f_5.9.9.1.9.10.3.2	Gear Reassembly	5	9	3	5.9.9.1.9.10	2 ../ipm/pictures/step010.gif	2	0
5.9.9.1.9.10.3.3	f_5.9.9.1.9.10.3.3	Gear reassemble on video	5	9	3	5.9.9.1.9.10	3 ../ipm/videos/step010.avi	3	0
5.9.9.1.9.11	f_5.9.9.1.9.11	Clearing Adjustment For The Hub Bearings	5	9	9	5.9.9.1	11 ../ipm/	0	3
5.9.9.1.9.11.2.1	f_5.9.9.1.9.11.2.1	Clearing adjustment for the hub bearings	5	9	2	5.9.9.1.9.11	1 ../ipm/	1	0
5.9.9.1.9.11.3.2	f_5.9.9.1.9.11.3.2	Hub Bearings	5	9	3	5.9.9.1.9.11	2 ../ipm/pictures/step011.gif	2	0
5.9.9.1.9.11.3.3	f_5.9.9.1.9.11.3.3	Clear and Adjust Hub Bearings	5	9	3	5.9.9.1.9.11	3 ../ipm/videos/step011.avi	3	0

5.9.9.1.9.12	f_5.9.9.1.9.12	Remounting of the Carrier	5	9	9	5.9.9.1	12	../ipm/	0	3
5.9.9.1.9.12.2.1	f_5.9.9.1.9.12.2.1	Carrier Remount	5	9	2	5.9.9.1.9.12	1	../ipm/	1	0
5.9.9.1.9.12.3.2	f_5.9.9.1.9.12.3.2	Place the Carrier	5	9	3	5.9.9.1.9.12	2	../ipm/pictures/step012.gif	2	0
5.9.9.1.9.12.3.3	f_5.9.9.1.9.12.3.3	Replace screws, spring washers and bolts	5	9	3	5.9.9.1.9.12	3	../ipm/pictures/step012_1.gif	2	0
5.9.9.1.9.12.3.4	f_5.9.9.1.9.12.3.4	Place the Carrier	5	9	3	5.9.9.1.9.12	4	../ipm/videos/step012.avi	3	0
5.9.9.1.9.12.3.5	f_5.9.9.1.9.12.3.5	Replace screws, spring washers and bolts	5	9	3	5.9.9.1.9.12	5	../ipm/videos/step012_1.avi	3	0
5.9.9.1.9.13	f_5.9.9.1.9.13	Remounting of the Brake Drum	5	9	9	5.9.9.1	13	../ipm/	0	3
5.9.9.1.9.13.2.1	f_5.9.9.1.9.13.2.1	Remount the Brake Drum	5	9	2	5.9.9.1.9.13	1	../ipm/	1	0
5.9.9.1.9.13.3.2	f_5.9.9.1.9.13.3.2	Replace the Brake Drum	5	9	3	5.9.9.1.9.13	2	../ipm/pictures/step013.gif	2	0
5.9.9.1.9.13.3.3	f_5.9.9.1.9.13.3.3	Replace the Brake Drum	5	9	3	5.9.9.1.9.13	3	../ipm/videos/step013.avi	3	0
5.9.9.1.9.14	f_5.9.9.1.9.14	Remounting of the Wheel	5	9	9	5.9.9.1	14	../ipm/	0	3
5.9.9.1.9.14.2.1	f_5.9.9.1.9.14.2.1	Wheel Remount	5	9	2	5.9.9.1.9.14	1	../ipm/	1	0
5.9.9.1.9.14.3.2	f_5.9.9.1.9.14.3.2	Replace the Wheel	5	9	3	5.9.9.1.9.14	2	../ipm/pictures/step014.gif	2	0
5.9.9.1.9.14.3.3	f_5.9.9.1.9.14.3.3	Step 2 to 5	5	9	3	5.9.9.1.9.14	3	../ipm/pictures/step014_1.gif	2	0
5.9.9.1.9.14.3.4	f_5.9.9.1.9.14.3.4	Replace the Wheel	5	9	3	5.9.9.1.9.14	4	../ipm/videos/step014.avi	3	0
5.9.9.1.9.14.3.5	f_5.9.9.1.9.14.3.5	Step 2 to 5	5	9	3	5.9.9.1.9.14	5	../ipm/videos/step014_1.avi	3	0
5.9.9.1.9.15	f_5.9.9.1.9.15	Assembly Final Check	5	9	9	5.9.9.1	15	../ipm/	0	3
5.9.9.1.9.15.2.1	f_5.9.9.1.9.15.2.1	Final Check	5	9	2	5.9.9.1.9.15	1	../ipm/	1	0
5.9.9.1.9.15.3.2	f_5.9.9.1.9.15.3.2	Wheel Nuts	5	9	3	5.9.9.1.9.15	2	../ipm/pictures/step015.gif	2	0
5.9.9.1.9.15.4.3	f_5.9.9.1.9.15.4.3	!! Important Caution !!	5	9	4	5.9.9.1.9.15	3	../ipm/	1	0
5.9.9.1.9.2	f_5.9.9.1.9.2	Wheel Dismounting	5	9	9	5.9.9.1	2	../ipm/	0	3
5.9.9.1.9.2.2.1	f_5.9.9.1.9.2.2.1	Wheel Dismounting	5	9	2	5.9.9.1.9.2	1	../ipm/	1	0
5.9.9.1.9.2.3.2	f_5.9.9.1.9.2.3.2	Wheel Dismounting Parts Requirements	5	9	3	5.9.9.1.9.2	2	../ipm/	1	0
5.9.9.1.9.2.3.3	f_5.9.9.1.9.2.3.3	Unscrew the nuts	5	9	3	5.9.9.1.9.2	3	../ipm/pictures/step_002_1.gif	2	0
5.9.9.1.9.2.3.4	f_5.9.9.1.9.2.3.4	Unscrew the nuts	5	9	3	5.9.9.1.9.2	4	../ipm/videos/step_002_1.avi	3	0
5.9.9.1.9.2.3.5	f_5.9.9.1.9.2.3.5	Take off the spring washers	5	9	3	5.9.9.1.9.2	5	../ipm/pictures/step_002_2.gif	2	0
5.9.9.1.9.2.3.6	f_5.9.9.1.9.2.3.6	Take off the spring washers	5	9	3	5.9.9.1.9.2	6	../ipm/videos/Step_002_2.avi	3	0
5.9.9.1.9.3	f_5.9.9.1.9.3	Dismount Brake Drum	5	9	9	5.9.9.1	3	../ipm/	0	3
5.9.9.1.9.3.2.1	f_5.9.9.1.9.3.2.1	Dismount Brake Drum	5	9	2	5.9.9.1.9.3	1	../ipm/	1	0

5.9.9.1.9.3.3.2	f_5.9.9.1.9.3.3.2	Brake Drum	5	9	3	5.9.9.1.9.3	2	../ipm/pictures/step_003.gif	2	0
5.9.9.1.9.3.3.3	f_5.9.9.1.9.3.4.3	Parts for Drum Dismount	5	9	3	5.9.9.1.9.3	3	../ipm/	1	0
5.9.9.1.9.3.3.4	f_5.9.9.1.9.3.3.4	Dissassemble Brake Drum	5	9	3	5.9.9.1.9.3	4	../ipm/videos/Step003.avi	3	0
5.9.9.1.9.4	f_5.9.9.1.9.4	Dismount Carrier	5	9	9	5.9.9.1	4	../ipm/	0	3
5.9.9.1.9.4.2.1	f_5.9.9.1.9.4.2.1	Dismount Carrier	5	9	2	5.9.9.1.9.4	1	../ipm/	1	0
5.9.9.1.9.4.3.2	f_5.9.9.1.9.4.3.2	Take off bolts, washers and screws	5	9	3	5.9.9.1.9.4	2	../ipm/pictures/Step004.gif	2	0
5.9.9.1.9.4.3.3	f_5.9.9.1.9.4.3.3	Take off bolts, washers and screws	5	9	3	5.9.9.1.9.4	3	../ipm/videos/step004.avi	3	0
5.9.9.1.9.4.3.4	f_5.9.9.1.9.4.3.4	Take out the Carrier	5	9	3	5.9.9.1.9.4	4	../ipm/pictures/step004_1.gif	2	0
5.9.9.1.9.4.3.5	f_5.9.9.1.9.4.3.5	Take out the Carrier	5	9	3	5.9.9.1.9.4	5	../ipm/videos/step004_1.avi	3	0
5.9.9.1.9.5	f_5.9.9.1.9.5	Dismount Ring Gear	5	9	9	5.9.9.1	5	../ipm/	0	3
5.9.9.1.9.5.2.1	f_5.9.9.1.9.5.2.1	Dismount Ring Gear	5	9	2	5.9.9.1.9.5	1	../ipm/	1	0
5.9.9.1.9.5.3.2	f_5.9.9.1.9.5.3.2	Take off the safty washer	5	9	3	5.9.9.1.9.5	2	../ipm/pictures/step005.gif	2	0
5.9.9.1.9.5.3.3	f_5.9.9.1.9.5.3.3	Take out the ring gear	5	9	3	5.9.9.1.9.5	3	../ipm/pictures/step005_1.gif	2	0
5.9.9.1.9.5.3.4	f_5.9.9.1.9.5.3.4	Take off the safty washer	5	9	3	5.9.9.1.9.5	4	../ipm/videos/step005.avi	3	0
5.9.9.1.9.5.3.5	f_5.9.9.1.9.5.3.5	Take out the ring gear	5	9	3	5.9.9.1.9.5	5	../ipm/videos/step005_1.avi	3	0
5.9.9.1.9.6	f_5.9.9.1.9.6	Dismount Hub	5	9	9	5.9.9.1	6	../ipm/	0	3
5.9.9.1.9.6.2.1	f_5.9.9.1.9.6.2.1	Dismount Hub	5	9	2	5.9.9.1.9.6	1	../ipm/	1	0
5.9.9.1.9.6.3.2	f_5.9.9.1.9.6.3.2	Dissassemble Hub	5	9	3	5.9.9.1.9.6	2	../ipm/pictures/Step006.gif	2	0
5.9.9.1.9.6.3.3	f_5.9.9.1.9.6.3.3	Dissassemble Hub	5	9	3	5.9.9.1.9.6	3	../ipm/videos/Step006.avi	3	0
5.9.9.1.9.7	f_5.9.9.1.9.7	General Check of Servo Brake	5	9	9	5.9.9.1	7	../ipm/	0	3
5.9.9.1.9.7.2.1	f_5.9.9.1.9.7.2.1	General Check	5	9	2	5.9.9.1.9.7	1	../ipm/	1	0
5.9.9.1.9.7.3.2	f_5.9.9.1.9.7.3.2	Verify Servo Brake	5	9	3	5.9.9.1.9.7	2	../ipm/pictures/step007.gif	2	0
5.9.9.1.9.8	f_5.9.9.1.9.8	Detailed Check of Servo Brake	5	9	9	5.9.9.1	8	../ipm/	0	3
5.9.9.1.9.8.2.1	f_5.9.9.1.9.8.2.1	Detailed Check	5	9	2	5.9.9.1.9.8	1	../ipm/	1	0
5.9.9.1.9.8.3.2	f_5.9.9.1.9.8.3.2	Servo Brake - Parts	5	9	3	5.9.9.1.9.8	2	../ipm/pictures/servo_b.gif	2	0
5.9.9.1.9.8.3.3	f_5.9.9.1.9.8.3.3	Servo Brake Assembly	5	9	3	5.9.9.1.9.8	3	../ipm/pictures/servo_b1.gif	2	0
5.9.9.1.9.9	f_5.9.9.1.9.9	Reassemble Hub	5	9	9	5.9.9.1	9	../ipm/	0	3
5.9.9.1.9.9.2.1	f_5.9.9.1.9.9.2.1	Reassemble Hub	5	9	2	5.9.9.1.9.9	1	../ipm/	1	0
5.9.9.1.9.9.3.2	f_5.9.9.1.9.9.3.2	Reassemble Hub	5	9	3	5.9.9.1.9.9	2	../ipm/pictures/step009.gif	2	0

5.9.9.1.9.9.3.3	f_5.9.9.1.9.9.3.3	Reassemble Hub on Video	5	9	3	5.9.9.1.9.9	3	../ipm/videos/step009.avi	3	0
5.9.9.2	f_5.9.9.2	Check the hydrostatic steering system	5	9	9	5.9	2	../ipm/	0	1
5.9.9.3	f_5.9.9.3	Check the steering mechanism	5	9	9	5.9	3	../ipm/	0	1
5.9.9.4	f_5.9.9.4	Check the motion control system	5	9	9	5.9	4	../ipm/	0	1
5.9.9.5	f_5.9.9.5	Check the mechanism for engagement of the	5	9	9	5.9	5	../ipm/	0	1
6.2.1.1	f_6.2.1.1	Annual Maintenance	6	2	1	6.2	1	../ipm/	1	0
6.2.4.5	f_6.2.4.5	General Maintenance Caution!!	6	2	4	6.2	5	../ipm/	1	0
6.2.9.1.4.1	f_6.2.9.1.4.1	CAUTION !!	6	2	4	6.2.9.1	1	../ipm/	1	0
6.2.9.2	f_6.2.9.2	Shift Maintenance	6	2	9	6.2	2	../ipm/	0	1
6.2.9.2.1.1	f_6.2.9.2.1.1	Start of Shift	6	2	1	6.2.9.2	1	../ipm/	1	0
6.2.9.2.1.2	f_6.2.9.2.1.2	End of Shift	6	2	1	6.2.9.2	2	../ipm/	1	0
6.2.9.2.4.3	f_6.2.9.2.4.3	CAUTION !!	6	2	4	6.2.9.2	3	../ipm/	1	0
6.2.9.3	f_6.2.9.3	Every 100 working hours	6	2	9	6.2	3	../ipm/	0	1
6.2.9.3.4.1	f_6.2.9.3.4.1	CAUTION !!	6	2	4	6.2.9.3	1	../ipm/	1	0
6.2.9.4	f_6.2.9.4	Every 400 working hours	6	2	9	6.2	4	../ipm/	0	1
6.2.9.4.4.1	f_6.2.9.4.4.1	CAUTION !!	6	2	4	6.2.9.4	1	../ipm/	1	0

APPENDIX B

CODING INFORMATION OBJECTS USING STRUCTURE BUILDER

Structure Builder is an authoring tool which is created in order to assist authors in building a semantically valid and implementation language-independent structure. The following figures show different types of atomic IOs coded using *Structure Builder*:

- Figure B.1 depicts the coding of a textual procedure “wheel dismounting” (*Form: 1, Function: 2*).
- Figure B.2 depicts the coding of a clarification image “Unscrew the nuts” (*Function: 3, Form: 2*).
- Figure B.3 depicts the coding of a clarification animation “Unscrew the nuts” (*Function: 3, Form: 3*).
- Figure B.4 depicts the coding of a clarification image “Take off the spring washers” (*Function: 3, Form: 2*).

The above IOs are all included in one container of type collection which is used to fully represent the “Wheel dismounting” procedure (*ID: 5.9.9.1.9.2*). This procedure is used for maintenance (*Cat: 5*), and for checking the truck (*Task: 9*).

Desc	Wheel Dismounting		ID	5.9.9.19.2.2.1
Category	5		Name	f_5.9.9.19.2.2.1
Task	9			
Function	2			
Container	5.9.9.19.2		C Form	0
Seq				
Form	1			
Location	/ipm/			
Content	1. Unscrew the 8 nuts 2. Take off the 8 spring washers TIME= 2 MINS			

Record: 14 | 1 | 1 | 2 | 3 | 4 | 5 | 6 | of 6 (Filtered)

Figure B.1 Textual Procedure “Wheel dismount”

Desc	Unscrew the nuts		ID	5.9.9.19.2.3.3
Category	5		Name	f_5.9.9.19.2.3.3
Task	9			
Function	3			
Container	5.9.9.19.2		C Form	0
Seq				
Form	2			
Location	/ipm/pictures/step_002			
Content	Empty			

Record: 14 | 4 | 3 | 2 | 3 | 4 | 5 | 6 | of 6 (Filtered)

Figure B.2 Clarification Image “Unscrew the nuts”

The screenshot shows a software application window titled 'ICs'. It contains a form with the following fields and values:

- Desc:** Unscrew the nuts
- Category:** 5
- Task:** 9
- Function:** 3
- Container:** 5.9.9.1.9.2
- Seq:** (empty)
- Form:** 3
- Location:** /ipm/videos/step_002_
- Content:** Empty
- Id:** 59919234
- Name:** f_59919234

At the bottom of the window, there is a status bar that reads: 'Record: 4 of 6 (Filtered)'.

Figure B.3 Clarification animation “Unscrew the nuts”

IOs

Desc Take off the spring washers

Category 5

Task 9

Function 3

Container 599.192 **G Form** 0

Seq

Form 2

Location ..\pics\pictures\step_002

Content Empty

ID	599.192.35
Name	f_599.192.35

Record: 5 of 6 (Filtered)

Figure B.4 Clarification Image “Take off the spring washers”

APPENDIX C

AUTOMATICALLY EXTRACTED PURPOSE-DRIVEN NAVIGATIONAL RELATIONSHIPS

The full set of the automatically-generated purpose-driven navigational relationships, “Action_Applied”, “Plan_Info”, and “Supported_By” are depicted in Tables C.1, C.2, and C.3 respectively.

Table C.1 Automatically-Generated Purpose-Driven “Action_Applied” Navigational Relationships

Source Description	Target Description	Source ID	Target ID
Electrical System - Parts	OI-Reverse	2.3.9.8.5.1	4.7.2.3
General Maintenance Caution!!	OI-Parking	6.2.4.5	4.7.2.4
Hand Brake	OI-Parking	1.3.9.2.1.2	4.7.2.4
Annual Maintenance	TM-Check the Servo Brake	6.2.1.1	5.9.9.1
Start of Shift	TM-Check the Servo Brake	6.2.9.2.1.1	5.9.9.1
Annual Maintenance	TM-Check the hydrostatic steering system	6.2.1.1	5.9.9.2
Annual Maintenance	TM-Check the steering mechanism	6.2.1.1	5.9.9.3
Annual Maintenance	TM-Check the motion control system	6.2.1.1	5.9.9.4
Annual Maintenance	TM-Check the mechanism for engagement of the	6.2.1.1	5.9.9.5
Annual Maintenance	TS-Check the servo brake	6.2.1.1	10.9.9.2
Start of Shift	TS-Check the servo brake	6.2.9.2.1.1	10.9.9.2

Table C.2 Automatically-Generated Purpose-Driven “Plan_Info” Navigational Relationships

SDesc	TDesc	Source	Target
Fork-Lift Truck	PP-Fork-Lift Truck	2.3.9.1	1.1.9.1
Fork-Lift Truck	PP-Fork-Lift Truck	3.3.9.1	1.1.9.1
Running the New Truck	PP-Fork-Lift Truck	4.5.2.2	1.1.9.1
Fork-Lift Truck - Description	PP-Fork-Lift Truck	3.3.9.1.5.1	1.1.9.1
Engine Technical View	PP-Fork-Lift Truck	3.3.9.5.5.1	1.1.9.1
Parking	TMS-Shift Maintenance	4.7.2.4	6.2.9.2
Engine - General View	PP-Engine - General	2.3.9.5.5.1	1.3.9.5.1.1
Running the New Truck	PP-Lifting Gear	4.5.2.2	1.3.9.3.1.1
Moving and Accelerating	PP-Lifting Gear	4.7.2.1	1.3.9.3.1.1
Picking Loads	PP-Lifting Gear	4.7.2.5	1.3.9.3.1.1
Unloading the Truck	PP-Lifting Gear	4.7.2.6	1.3.9.3.1.1
Lifting Gear - Technical View	PP-Lifting Gear	3.3.9.3.5.1	1.3.9.3.1.1
Hydraulic System - Technical View	PP-Lifting Gear	3.3.9.7.5.1	1.3.9.3.1.1
Fork-Lift Truck	PP-Fork-Lift Truck	2.3.9.1	1.3.9.1
Fork-Lift Truck	PP-Fork-Lift Truck	3.3.9.1	1.3.9.1
Running the New Truck	PP-Fork-Lift Truck	4.5.2.2	1.3.9.1
Fork-Lift Truck - Description	PP-Fork-Lift Truck	3.3.9.1.5.1	1.3.9.1
Engine Technical View	PP-Fork-Lift Truck	3.3.9.5.5.1	1.3.9.1
Braking System	PP-Braking System	2.3.9.2	1.3.9.2
Braking System - Complete	PP-Braking System	2.3.9.2.9.1	1.3.9.2
Braking System 1547475	PP-Braking System	2.3.9.2.9.1.5.1	1.3.9.2
Braking System Parts	PP-Braking System	2.3.9.2.9.1.5.2	1.3.9.2
Hand Brake - Complete	PP-Braking System	2.3.9.2.9.2.5.2	1.3.9.2

Braking System	PP-Braking System	3.3.9.2	1.3.9.2
Braking System - Technical View	PP-Braking System	3.3.9.2.5.1	1.3.9.2
Braking System - Technical View	PP-Braking System	3.3.9.2.5.2	1.3.9.2
Hand Brake as part of the Braking System	PP-Braking System	2.3.9.2.9.2.5.3	1.3.9.2
Lifting System	PP-Lifting System	2.3.9.3	1.3.9.3
Hydraulic Lifting System	PP-Lifting System	2.3.9.3.9.2	1.3.9.3
Lifting System - Complete	PP-Lifting System	2.3.9.3.9.1	1.3.9.3
Lifting System 1547121	PP-Lifting System	2.3.9.3.9.1.5.1	1.3.9.3
Lifting System Parts	PP-Lifting System	2.3.9.3.9.1.5.2	1.3.9.3
Lifting System	PP-Lifting System	3.3.9.3	1.3.9.3
Drive Axle	PP-Drive Axle	2.3.9.4	1.3.9.4
Front Drive Axle	PP-Drive Axle	2.3.9.4.9.1	1.3.9.4
Rear Drive Axle	PP-Drive Axle	2.3.9.4.9.2	1.3.9.4
Front Drive Axle 1425505	PP-Drive Axle	2.3.9.4.9.1.5.1	1.3.9.4
Parts for Front Drive Axle	PP-Drive Axle	2.3.9.4.9.1.5.2	1.3.9.4
Running the New Truck	PP-Drive Axle	4.5.2.2	1.3.9.4
Working on Sites with Uneven Surfaces	PP-Drive Axle	4.7.2.7	1.3.9.4
Check the mechanism for engagement of the rear drive axle	PP-Drive Axle	5.9.9.5	1.3.9.4
Braking System - Technical View	PP-Drive Axle	3.3.9.2.5.1	1.3.9.4
Drive Axle	PP-Drive Axle	3.3.9.4	1.3.9.4
Drive Axle - Technical View	PP-Drive Axle	3.3.9.4.5.1	1.3.9.4
Drive Axle - Technical View	PP-Drive Axle	3.3.9.4.5.2	1.3.9.4
Chassis	PP-Drive Axle	3.3.9.4.5.3	1.3.9.4
Pipes	PP-Drive Axle	2.3.9.2.9.3.5.1	1.3.9.4
Adjust lining to drum clearance	PP-Drive Axle	10.10.9.8.2.1	1.3.9.4
Check rope	PP-Drive Axle	10.9.9.6.2.1	1.3.9.4
Running the New Truck	PP-Engine	4.5.2.2	1.3.9.5
Starting the Engine	PP-Engine	4.6.2.1	1.3.9.5
Shutting the Engine	PP-Engine	4.6.2.2	1.3.9.5
Moving and Accelerating	PP-Engine	4.7.2.1	1.3.9.5

Decelerating and Stopping	PP-Engine	4.7.2.2	1.3.9.5
Parking	PP-Engine	4.7.2.4	1.3.9.5
Engine	PP-Engine	3.3.9.5	1.3.9.5
Engine Technical View	PP-Engine	3.3.9.5.5.1	1.3.9.5
Engine	PP-Engine	2.3.9.5	1.3.9.5
Engine - General View	PP-Engine	2.3.9.5.5.1	1.3.9.5
Chassis	PP-Engine	3.3.9.4.5.3	1.3.9.5
Check the hydrostatic steering system	PP-Steering system	5.9.9.2	1.3.9.6
Steering system	PP-Steering system	3.3.9.6	1.3.9.6
Steering System -Technical View	PP-Steering system	3.3.9.6.5.1	1.3.9.6
Steering system	PP-Steering system	2.3.9.6	1.3.9.6
Steering system - Parts	PP-Steering system	2.3.9.6.5.1	1.3.9.6
Hydraulic System - Technical View	PP-Steering system	3.3.9.7.5.1	1.3.9.6
Steering System - Complete	PP-Steering system	2.3.9.6.5.2	1.3.9.6
Lifting System Parts	PP-Hydraulic system	2.3.9.3.9.1.5.2	1.3.9.7
Running the New Truck	PP-Hydraulic system	4.5.2.2	1.3.9.7
Hydraulic system	PP-Hydraulic system	2.3.9.7	1.3.9.7
Hydraulic System - Parts	PP-Hydraulic system	2.3.9.7.5.1	1.3.9.7
Hydraulic system	PP-Hydraulic system	3.3.9.7	1.3.9.7
Hydraulic System - Technical View	PP-Hydraulic system	3.3.9.7.5.1	1.3.9.7
Hydraulic System - Complete	PP-Hydraulic system	2.3.9.7.5.2	1.3.9.7
Electrical system	PP-Electrical system	2.3.9.8	1.3.9.8
Electrical System - Parts	PP-Electrical system	2.3.9.8.5.1	1.3.9.8
Electrical system	PP-Electrical system	3.3.9.8	1.3.9.8
Electrical system - Technical View	PP-Electrical system	3.3.9.8.5.1	1.3.9.8
Electrical System - Complete	PP-Electrical system	2.3.9.8.5.2	1.3.9.8
Front Drive Axle 1425505	PP-Servo Brake	2.3.9.4.9.1.5.1	1.3.9.2.1.1
Servo Brake	PP-Servo Brake	2.3.9.2.9.5	1.3.9.2.1.1
Parts for Gear Box 1483448	PP-Servo Brake	2.3.9.4.9.4.5.2	1.3.9.2.1.1
Servo Brake 1431442 - Technical View	PP-Servo Brake	2.3.9.2.9.5.5.1	1.3.9.2.1.1

Servo Brake 1431442 - Parts	PP-Servo Brake	2.3.9.2.9.5.5.2	1.3.9.2.1.1
Check the Servo Brake	PP-Servo Brake	5.9.9.1	1.3.9.2.1.1
General Check of Servo Brake	PP-Servo Brake	5.9.9.1.9.7	1.3.9.2.1.1
Verify Servo Brake	PP-Servo Brake	5.9.9.1.9.7.3.2	1.3.9.2.1.1
Detailed Check of Servo Brake	PP-Servo Brake	5.9.9.1.9.8	1.3.9.2.1.1
Servo Brake - Parts	PP-Servo Brake	5.9.9.1.9.8.3.2	1.3.9.2.1.1
Braking System - Technical View	PP-Servo Brake	3.3.9.2.5.1	1.3.9.2.1.1
Servo Brake	PP-Servo Brake	10.10.9.2.9.2.3.2	1.3.9.2.1.1
Verify Servo Brake	PP-Servo Brake	10.10.9.2.9.2.3.3	1.3.9.2.1.1
Servo Brake	PP-Servo Brake	10.10.9.3.9.2.3.2	1.3.9.2.1.1
Check the servo brake	PP-Servo Brake	10.9.9.2	1.3.9.2.1.1
Adjust lining to drum clearance	PP-Servo Brake	10.10.9.8.2.1	1.3.9.2.1.1
Servo brake	PP-Servo Brake	10.10.9.8.3.2	1.3.9.2.1.1
Verify Servo Brake	PP-Servo Brake	10.10.9.3.9.2.3.3	1.3.9.2.1.1
Servo Brake	PP-Servo Brake	10.10.9.4.9.2.3.2	1.3.9.2.1.1
Verify Servo Brake	PP-Servo Brake	10.10.9.4.9.2.3.3	1.3.9.2.1.1
Servo Brake	PP-Servo Brake	10.10.9.7.9.2.3.2	1.3.9.2.1.1
Verify Servo Brake	PP-Servo Brake	10.10.9.7.9.2.3.3	1.3.9.2.1.1
Servo Brake Assembly	PP-Servo Brake	5.9.9.1.9.8.3.3	1.3.9.2.1.1
Hand Brake	PP-Hand Brake	2.3.9.2.9.2	1.3.9.2.1.2
Hand Brake 1416533	PP-Hand Brake	2.3.9.2.9.2.5.1	1.3.9.2.1.2
Hand Brake - Complete	PP-Hand Brake	2.3.9.2.9.2.5.2	1.3.9.2.1.2
Moving and Accelerating	PP-Hand Brake	4.7.2.1	1.3.9.2.1.2
Picking Loads	PP-Hand Brake	4.7.2.5	1.3.9.2.1.2
Preparation Process	PP-Hand Brake	5.9.9.1.9.1.2.1	1.3.9.2.1.2
Wheel Remount	PP-Hand Brake	5.9.9.1.9.14.2.1	1.3.9.2.1.2
Final Check	PP-Hand Brake	5.9.9.1.9.15.2.1	1.3.9.2.1.2
Dismount Brake Drum	PP-Hand Brake	5.9.9.1.9.3.2.1	1.3.9.2.1.2
Hand Brake as part of the Braking System	PP-Hand Brake	2.3.9.2.9.2.5.3	1.3.9.2.1.2
Check the hand brake rope	PP-Hand Brake	10.9.9.6	1.3.9.2.1.2

Adjust the hand brake rope	PP-Hand Brake	10.10.9.9	1.3.9.2.1.2
Replace the hand brake rope	PP-Hand Brake	10.10.9.10	1.3.9.2.1.2
Hand brake	PP-Hand Brake	10.9.9.6.3.2	1.3.9.2.1.2
hand brake	PP-Hand Brake	10.10.9.10.3.2	1.3.9.2.1.2
Hand brake rope	PP-Hand Brake	10.10.9.9.3.2	1.3.9.2.1.2
Caution	PP-Hand Brake	5.9.9.1.9.1.4.4	1.3.9.2.1.2
Brake Fluid Pipes	PP-Brake Fluid Pipes	2.3.9.2.9.3	1.3.9.2.1.3
Pipes	PP-Brake Fluid Pipes	2.3.9.2.9.3.5.1	1.3.9.2.1.3
Check the brake fluid pipes	PP-Brake Fluid Pipes	10.9.9.1	1.3.9.2.1.3
Replace damaged brake fluid pipes	PP-Brake Fluid Pipes	10.10.9.1	1.3.9.2.1.3
Brake fluid pipes	PP-Brake Fluid Pipes	10.9.9.1.3.2	1.3.9.2.1.3
Moving and Accelerating	PP-Brake Pedal	4.7.2.1	1.3.9.2.1.4
Decelerating and Stopping	PP-Brake Pedal	4.7.2.2	1.3.9.2.1.4
Reverse	PP-Brake Pedal	4.7.2.3	1.3.9.2.1.4
Braking System - Technical View	PP-Brake Pedal	3.3.9.2.5.1	1.3.9.2.1.4
Pipes	PP-Brake Pedal	2.3.9.2.9.3.5.1	1.3.9.2.1.4
Brake Pedal	PP-Brake Pedal	2.3.9.2.9.4	1.3.9.2.1.4
Brake Pedal - Parts	PP-Brake Pedal	2.3.9.2.9.4.5.1	1.3.9.2.1.4
Brake Pedal - Overall	PP-Brake Pedal	2.3.9.2.9.4.3.2	1.3.9.2.1.4
Brake Pedal - Parts View	PP-Brake Pedal	2.3.9.2.9.4.5.3	1.3.9.2.1.4
Check the brake pedal	PP-Brake Pedal	10.9.9.3	1.3.9.2.1.4
Unblock brake pedal bushings	PP-Brake Pedal	10.10.9.5	1.3.9.2.1.4
Brake Pedal	PP-Brake Pedal	10.9.9.3.3.2	1.3.9.2.1.4
Brake Fluid Pipes	PP-Brake Fluid	2.3.9.2.9.3	1.3.9.2.1.5
Pipes	PP-Brake Fluid	2.3.9.2.9.3.5.1	1.3.9.2.1.5
Brake Fluid Reservoir	PP-Brake Fluid	2.3.9.2.9.6	1.3.9.2.1.5
Brake Fluid Reservoir - Parts	PP-Brake Fluid	2.3.9.2.9.6.5.1	1.3.9.2.1.5
Brake Fluid Reservoir - Technical View	PP-Brake Fluid	2.3.9.2.9.6.5.2	1.3.9.2.1.5
Check the brake fluid pipes	PP-Brake Fluid	10.9.9.1	1.3.9.2.1.5
Replace damaged brake fluid pipes	PP-Brake Fluid	10.10.9.1	1.3.9.2.1.5

Brake fluid pipes	PP-Brake Fluid	10.9.9.1.3.2	1.3.9.2.1.5
Caution	PP-Brake Fluid	10.10.9.1.4.2	1.3.9.2.1.5
Brake Fluid	PP-Brake Fluid	3.3.9.2.5.3	1.3.9.2.1.5

Table C.3 Automatically-Generated Purpose-Driven “Supported_By” Navigational Relationships

SDesc	TDesc	Source	Target
Introduction	PL-Fork-Lift Truck	1.1.9.1.1.1	2.3.9.1
Fork-Lift Truck	PL-Fork-Lift Truck	1.1.9.1	2.3.9.1
Running the New Truck	PL-Fork-Lift Truck	4.5.2.2	2.3.9.1
Fork-Lift Truck	PL-Fork-Lift Truck	1.3.9.1	2.3.9.1
Start of Shift	PL-Braking System	6.2.9.2.1.1	2.3.9.2
Braking System	PL-Braking System	1.3.9.2	2.3.9.2
Moving and Accelerating	PL-Hand Brake	4.7.2.1	2.3.9.2.9.2
Picking Loads	PL-Hand Brake	4.7.2.5	2.3.9.2.9.2
Preparation Process	PL-Hand Brake	5.9.9.1.9.1.2.1	2.3.9.2.9.2
Wheel Remount	PL-Hand Brake	5.9.9.1.9.14.2.1	2.3.9.2.9.2
Final Check	PL-Hand Brake	5.9.9.1.9.15.2.1	2.3.9.2.9.2
Dismount Brake Drum	PL-Hand Brake	5.9.9.1.9.3.2.1	2.3.9.2.9.2
Check the hand brake rope	PL-Hand Brake	10.9.9.6	2.3.9.2.9.2
Adjust the hand brake rope	PL-Hand Brake	10.10.9.9	2.3.9.2.9.2
Replace the hand brake rope	PL-Hand Brake	10.10.9.10	2.3.9.2.9.2
Hand Brake	PL-Hand Brake	1.3.9.2.1.2	2.3.9.2.9.2
Lifting System	PL-Lifting System	1.3.9.3	2.3.9.3
Picking Loads	PL-Fork Arm	4.7.2.5	2.3.9.3.9.3
Unloading the Truck	PL-Fork Arm	4.7.2.6	2.3.9.3.9.3
General Performance and Dimensions	PL-Fork Arm	1.3.9.1.1.1	2.3.9.3.9.3
Running the New Truck	PL-Drive Axle	4.5.2.2	2.3.9.4
Working on Sites with Uneven Surfaces	PL-Drive Axle	4.7.2.7	2.3.9.4
Check the mechanism for engagement of the rear drive axle	PL-Drive Axle	5.9.9.5	2.3.9.4
Annual Maintenance	PL-Drive Axle	6.2.1.1	2.3.9.4

Start of Shift	PL-Drive Axle	6.2.9.2.1.1	2.3.9.4
Drive Axle	PL-Drive Axle	1.3.9.4	2.3.9.4
Adjust linning to drum clearance	PL-Drive Axle	10.10.9.8.2.1	2.3.9.4
Check rope	PL-Drive Axle	10.9.9.6.2.1	2.3.9.4
Adjust linning to drum clearance	PL-Front Drive Axle	10.10.9.8.2.1	2.3.9.4.9.1
Working on Sites with Uneven Surfaces	PL-Rear Drive Axle	4.7.2.7	2.3.9.4.9.2
Check the mechanism for engagement of the rear drive axle	PL-Rear Drive Axle	5.9.9.5	2.3.9.4.9.2
Annual Maintenance	PL-Rear Drive Axle	6.2.1.1	2.3.9.4.9.2
Check rope	PL-Rear Drive Axle	10.9.9.6.2.1	2.3.9.4.9.2
Introduction	TD-Fork-Lift Truck	1.1.9.1.1.1	3.3.9.1
Fork-Lift Truck	TD-Fork-Lift Truck	1.1.9.1	3.3.9.1
Running the New Truck	TD-Fork-Lift Truck	4.5.2.2	3.3.9.1
Fork-Lift Truck	TD-Fork-Lift Truck	1.3.9.1	3.3.9.1
Remounting of the Wheel	PL-Wheel	5.9.9.1.9.14	2.3.9.4.9.3
Wheel Remount	PL-Wheel	5.9.9.1.9.14.2.1	2.3.9.4.9.3
Final Check	PL-Wheel	5.9.9.1.9.15.2.1	2.3.9.4.9.3
Wheel Dismounting	PL-Wheel	5.9.9.1.9.2	2.3.9.4.9.3
Wheel Dismounting	PL-Wheel	5.9.9.1.9.2.2.1	2.3.9.4.9.3
Start of Shift	PL-Wheel	6.2.9.2.1.1	2.3.9.4.9.3
General Performance and Dimensions	PL-Wheel	1.3.9.1.1.1	2.3.9.4.9.3
Check Clearance	PL-Wheel	10.9.9.5.2.1	2.3.9.4.9.3
Adjust linning to drum clearance	PL-Wheel	10.10.9.8.2.1	2.3.9.4.9.3
Check rope	PL-Wheel	10.9.9.6.2.1	2.3.9.4.9.3
Servo Brake	PL-Wheel	1.3.9.2.1.1	2.3.9.4.9.3
Hand Brake	PL-Wheel	1.3.9.2.1.2	2.3.9.4.9.3
Check the Servo Brake	PL-Servo Brake	5.9.9.1	2.3.9.2.9.5
General Check of Servo Brake	PL-Servo Brake	5.9.9.1.9.7	2.3.9.2.9.5
Annual Maintenance	PL-Servo Brake	6.2.1.1	2.3.9.2.9.5
Detailed Check of Servo Brake	PL-Servo Brake	5.9.9.1.9.8	2.3.9.2.9.5
Start of Shift	PL-Servo Brake	6.2.9.2.1.1	2.3.9.2.9.5

Check the servo brake	PL-Servo Brake	10.9.9.2	2.3.9.2.9.5
Adjust lining to drum clearance	PL-Servo Brake	10.10.9.8.2.1	2.3.9.2.9.5
Servo Brake	PL-Servo Brake	1.3.9.2.1.1	2.3.9.2.9.5
Start of Shift	TD-Cabin	6.2.9.2.1.1	3.3.9.1.5.2
Brake Pedal	TD-Cabin	1.3.9.2.1.4	3.3.9.1.5.2
Running the New Truck	PL-Hydraulic system	4.5.2.2	2.3.9.7
Start of Shift	PL-Hydraulic system	6.2.9.2.1.1	2.3.9.7
Hydraulic system	PL-Hydraulic system	1.3.9.7	2.3.9.7
Electrical system - General	PL-Electrical system	1.3.9.8.1.1	2.3.9.8
Electrical system	PL-Electrical system	1.3.9.8	2.3.9.8
Start of Shift	TD-Braking System	6.2.9.2.1.1	3.3.9.2
Braking System	TD-Braking System	1.3.9.2	3.3.9.2
Lifting System	TD-Lifting System	1.3.9.3	3.3.9.3
Running the New Truck	TD-Drive Axle	4.5.2.2	3.3.9.4
Working on Sites with Uneven Surfaces	TD-Drive Axle	4.7.2.7	3.3.9.4
Check the mechanism for engagement of the rear drive axle	TD-Drive Axle	5.9.9.5	3.3.9.4
Annual Maintenance	TD-Drive Axle	6.2.1.1	3.3.9.4
Start of Shift	TD-Drive Axle	6.2.9.2.1.1	3.3.9.4
Drive Axle	TD-Drive Axle	1.3.9.4	3.3.9.4
Adjust lining to drum clearance	TD-Drive Axle	10.10.9.8.2.1	3.3.9.4
Check rope	TD-Drive Axle	10.9.9.6.2.1	3.3.9.4
Running the New Truck	TD-Engine	4.5.2.2	3.3.9.5
Starting the Engine	TD-Engine	4.6.2.1	3.3.9.5
Shutting the Engine	TD-Engine	4.6.2.2	3.3.9.5
Moving and Accelerating	TD-Engine	4.7.2.1	3.3.9.5
Decelerating and Stopping	TD-Engine	4.7.2.2	3.3.9.5
Parking	TD-Engine	4.7.2.4	3.3.9.5
Start of Shift	TD-Engine	6.2.9.2.1.1	3.3.9.5
End of Shift	TD-Engine	6.2.9.2.1.2	3.3.9.5
Engine - General	TD-Engine	1.3.9.5.1.1	3.3.9.5

General Performance and Dimensions	TD-Engine	1.3.9.1.1.1	3.3.9.5
Engine	TD-Engine	1.3.9.5	3.3.9.5
Check the hydrostatic steering system	TD-Steering system	5.9.9.2	3.3.9.6
Annual Maintenance	TD-Steering system	6.2.1.1	3.3.9.6
Steering system	TD-Steering system	1.3.9.6	3.3.9.6
Running the New Truck	PL-Engine	4.5.2.2	2.3.9.5
Starting the Engine	PL-Engine	4.6.2.1	2.3.9.5
Shutting the Engine	PL-Engine	4.6.2.2	2.3.9.5
Moving and Accelerating	PL-Engine	4.7.2.1	2.3.9.5
Decelerating and Stopping	PL-Engine	4.7.2.2	2.3.9.5
Parking	PL-Engine	4.7.2.4	2.3.9.5
Start of Shift	PL-Engine	6.2.9.2.1.1	2.3.9.5
End of Shift	PL-Engine	6.2.9.2.1.2	2.3.9.5
Engine - General	PL-Engine	1.3.9.5.1.1	2.3.9.5
General Performance and Dimensions	PL-Engine	1.3.9.1.1.1	2.3.9.5
Engine	PL-Engine	1.3.9.5	2.3.9.5
Check the hydrostatic steering system	PL-Steering system	5.9.9.2	2.3.9.6
Annual Maintenance	PL-Steering system	6.2.1.1	2.3.9.6
Steering system	PL-Steering system	1.3.9.6	2.3.9.6
Running the New Truck	TD-Hydraulic system	4.5.2.2	3.3.9.7
Start of Shift	TD-Hydraulic system	6.2.9.2.1.1	3.3.9.7
Hydraulic system	TD-Hydraulic system	1.3.9.7	3.3.9.7
Electrical system - General	TD-Electrical system	1.3.9.8.1.1	3.3.9.8
Electrical system	TD-Electrical system	1.3.9.8	3.3.9.8
Unloading/Loading of the Truck from/on Transport Vehicles	TD-Chassis	4.7.2.8	3.3.9.4.5.3
Start of Shift	TD-Chassis	6.2.9.2.1.1	3.3.9.4.5.3
Check the brake fluid pipes	PL-Brake Fluid Pipes	10.9.9.1	2.3.9.2.9.3
Replace damaged brake fluid pipes	PL-Brake Fluid Pipes	10.10.9.1	2.3.9.2.9.3
Brake Fluid Pipes	PL-Brake Fluid Pipes	1.3.9.2.1.3	2.3.9.2.9.3
Check the brake fluid pipes	PL-Pipes	10.9.9.1	2.3.9.2.9.3.5.1

Replace damaged brake fluid pipes	PL-Pipes	10.10.9.1	2.3.9.2.9.3.5.1
Check pipes	PL-Pipes	10.9.9.1.2.1	2.3.9.2.9.3.5.1
Repair pipes	PL-Pipes	10.10.9.1.2.1	2.3.9.2.9.3.5.1
Brake Fluid Pipes	PL-Pipes	1.3.9.2.1.3	2.3.9.2.9.3.5.1
Moving and Accelerating	PL-Brake Pedal	4.7.2.1	2.3.9.2.9.4
Decelerating and Stopping	PL-Brake Pedal	4.7.2.2	2.3.9.2.9.4
Reverse	PL-Brake Pedal	4.7.2.3	2.3.9.2.9.4
Check the brake pedal	PL-Brake Pedal	10.9.9.3	2.3.9.2.9.4
Unblock brake pedal bushings	PL-Brake Pedal	10.10.9.5	2.3.9.2.9.4
Brake Fluid Pipes	PL-Brake Pedal	1.3.9.2.1.3	2.3.9.2.9.4
Brake Pedal	PL-Brake Pedal	1.3.9.2.1.4	2.3.9.2.9.4
Start of Shift	PL-Brake Fluid Reservoir	6.2.9.2.1.1	2.3.9.2.9.6
Brake Fluid	PL-Brake Fluid Reservoir	1.3.9.2.1.5	2.3.9.2.9.6
Start of Shift	TD-Brake Fluid	6.2.9.2.1.1	3.3.9.2.5.3
Check the brake fluid pipes	TD-Brake Fluid	10.9.9.1	3.3.9.2.5.3
Replace damaged brake fluid pipes	TD-Brake Fluid	10.10.9.1	3.3.9.2.5.3
Brake Fluid Pipes	TD-Brake Fluid	1.3.9.2.1.3	3.3.9.2.5.3
Brake Fluid	TD-Brake Fluid	1.3.9.2.1.5	3.3.9.2.5.3

APPENDIX D

FRAME-BASED PRESENTATION TEMPLATES

The presentation templates used to create the user interface are based on HTML frames, which divide the interface window into a number of navigation-able areas. Every composite IO is associated with a presentation template which is based on its access method. Thus, there are three types of templates namely index, collection, and guided tour. These templates are created using Java classes that generate Java Server Pages (JSP™) code. JSP™ is an HTML-based document with embedded Java-based control statements.

D.1 Index Template (TmpIndex.java)

```
import java.sql.*;
import java.io.*;

public class TmpIndex {

    FileWriter Temp;
    // String BColor="#FFD700";
    String BColor="#FAEBD7";
    TmpIndex(String Dir, String Name, String Desc, String E_ID) throws IOException{

        Temp = new FileWriter(Dir+"/F_"+Name+".jsp");
        Temp.write("<%@ page import=\"Frames.Frame\" %>");
        Temp.write("<html><head><title>"+Desc+"</title><base target=\"_top\"></head>");
        Temp.write("<frameset cols=\"20%,60%,20%\" FRAMEBORDER=yes
            BORDERCOLOR=\"Black\"> ");
        Temp.write("<frame src=\"Activities.html\" name=\"Activities\" FRAMEBORDER=no > ");
        Temp.write("<frame src=\""+Name+".html\" name=\"Content\" FRAMEBORDER=no > ");
        Temp.write("<frame src=\"CLinks.html\" name=\"CLinks\" FRAMEBORDER=no > ");
        Temp.write("</frameset> ");
        Temp.write("<body> ");
        Temp.write("<% Frame Fr = new Frame(\""+E_ID+"\" );");
        Temp.write(" Fr.CrActivities(); ");
        Temp.write(" Fr.CrCLinks(); ");
        Temp.write(" Fr.CrPath(); %> ");
        Temp.write("</body> ");
        Temp.write("</html> ");
```

```

        Temp.close();
    }
}

```

D.2 Collection Template (TmpCollection.java)

```

import java.sql.*;
import java.io.*;

public class TmpCollection {

    FileWriter Temp;
    // String BColor="#FFD700";
    String BColor="#FAEBD7";
    TmpCollection(String Dir, String Name, String Desc, String E_ID) throws IOException{

        Temp = new FileWriter(Dir+"/F_"+Name+".jsp");
        Temp.write("<%@ page import=\"Frames.Frame\" %>");
        Temp.write("<html><head><title>"+Desc+"</title><base target=\"_top\"></head>");
        Temp.write("<frameset cols=\"20%,60%,20%\" FRAMEBORDER=yes
            BORDERCOLOR=\"Black\"> ");
        Temp.write("<frame src=\"Activities.html\" name=\"Activities\" FRAMEBORDER=no > ");
        Temp.write("<frame src=\""+Name+".html\" name=\"Content\" FRAMEBORDER=no> ");
        Temp.write("<frame src=\"CLinks.html\" name=\"CLinks\" FRAMEBORDER=no> ");
        Temp.write("</frameset> ");
        Temp.write("<body> ");
        Temp.write("<% Frame Fr = new Frame(\""+E_ID+"\" ); ");
        Temp.write(" Fr.CrActivities(); ");
        Temp.write(" Fr.CrCLinks(); ");
        Temp.write(" Fr.CrPath(); %> ");
        Temp.write("</body> ");
        Temp.write("</html> ");
        Temp.close();
    }
}

```

D.3 Guided Tour Template (TmpGTour.java)

```

import java.sql.*;
import java.io.*;

public class TmpGTour {

    FileWriter Temp;

    public TmpGTour(String Dir, String Name, String Desc, String E_ID, String IOName) throws
    IOException{

        Temp = new FileWriter(Dir+"/F_"+Name+".jsp");
        Temp.write("<%@ page import=\"Frames.Frame\" %>");
        Temp.write("<html><head><title>"+Desc+"</title><base target=\"_top\"></head>");
        Temp.write("<frameset cols=\"20%,60%,20%\" FRAMEBORDER=yes
            BORDERCOLOR=\"Black\"> ");
        Temp.write("<frameset rows=\"60%,40%\" FRAMEBORDER=yes
            BORDERCOLOR=\"Black\"> ");
    }
}

```



```

Temp.write("<frame src=\"Activities.html\" name=\"Activities\" FRAMEBORDER=no > ");
Temp.write("<frame src=\"CLinks.html\" name=\"CLinks\" FRAMEBORDER=no > ");
Temp.write("</frameset> ");
Temp.write("<frame src=\""+IOName+".html\" name=\"Content\" FRAMEBORDER=no> ");
Temp.write("<frame src=\"List.html\" name=\"List\" FRAMEBORDER=yes
        BORDERCOLOR=\"Black\"> ");
Temp.write("</frameset> ");
Temp.write("<body> ");
Temp.write("<% Frame Fr = new Frame(\""+E_ID+""); ");
Temp.write(" Fr.CrActivities(); ");
Temp.write(" Fr.CrCLinks(); ");
Temp.write(" Fr.CrList();");
Temp.write(" Fr.CrPath(); %> ");
Temp.write("</body> ");
Temp.write("</html> ");
Temp.close();
}

```

```

}

```

APPENDIX E

AUTOMATIC GENERATION OF KNOWLEDGE BASES IN

e2gLite ES SHELL FORMAT

E.1 KB Generator – Source Code

The *KB Generator* is a Java-based ES shell-dependent tool that has been created in order to automatically generate the rule-based KB by transforming the structured shallow knowledge data into rules and prompts in e2gLite ES shell format. It interacts with the technical documentation metadata to resolve the deep knowledge references embedded in the structured shallow knowledge data. In addition, it realises the new procedures and inserts them in the technical documentation.

E.2 the Complete Rule-Based KB Generated in e2gLite ES Shell Format

The ES shell that has been selected for demonstration purposes is the freely available e2gLite ES shell [e2gLite, 2003]. The e2gLite ES shell is a Java applet that is embedded in a Web page and downloaded from the Web server by the user's browser. The applet loads a knowledge base from the server and then runs entirely on the browser. It uses a simple, special e2gLite language for encoding KBs, and it is fully Web enabled. The generated KB file is divided into rules and prompts.

E.1 KB Generator – Source Code (KB.java)

```
import java.sql.*;
import java.io.*;

public class KB {

    public static void main(String args[]) throws IOException {

        FileWriter out= new FileWriter("C:\\PhD\\MY_Proj\\Programs\\ExSys\\BrakingSYS.kb");
        String Proc_ID="";
        int i=1;

        // Insert new Diag. and Correct procedure into the Manual
        // and Update their ID in the Faults tables
        try{
            DataBase db2=new DataBase();
            db2.Connect("DSIS");
            ResultSet rs2=db2.Query(" SELECT DISTINCT D_Proc, C_Proc FROM ES_Faults ");

            while (rs2.next())
            {
                String D_Proc=rs2.getString(1);
                String C_Proc=rs2.getString(2);

                DataBase db3=new DataBase();
                db3.Connect("DSIS");
                ResultSet rs3=db3.Query(" SELECT Description FROM IOs "+
                    " WHERE ID = '\""+D_Proc+"\"'");
                if (!rs3.next()) {
                    newIO DProc = new newIO();
                    Proc_ID=DProc.Insert_n_Update(D_Proc,0);
                }
                db3.disconnect();
                DataBase db4=new DataBase();
                db4.Connect("DSIS");
                ResultSet rs4=db4.Query(" SELECT Description FROM IOs "+
                    " WHERE ID = '\""+C_Proc+"\"'");
                if (!rs4.next()) {
```

```

        newIO CProc = new newIO();
        Proc_ID=CProc.Insert_n_Update(C_Proc,1);
    }
    db4.disconnect();
} //while rs2.next
db2.disconnect();

// Generate RULES

DataBase db1=new DataBase();
db1.Connect("DSIS");

ResultSet rs1=db1.Query(" SELECT ES_Faults.ID, IOs.Description, ES_Faults.Outcome "+
    " FROM (ES_Faults INNER JOIN IOs ON ES_Faults.D_Proc = IOs.ID) "+
    " INNER JOIN IOs AS IOs_1 ON ES_Faults.C_Proc = IOs_1.ID "+
    " ORDER BY ES_Faults.ID; ");

while (rs1.next())
{
    int F_ID=rs1.getInt(1);
    String DProc_Desc=rs1.getString(2);
    String Outcome=rs1.getString(3);

// Fault Symptoms with Causes
    db2=new DataBase();
    db2.Connect("DSIS");
    rs2=db2.Query( " SELECT ES_C_Class.Desc, ES_Causes.Status, ES_FCs.CF "+
        " FROM (ES_C_Class INNER JOIN ES_Causes ON ES_C_Class.ID = ES_Causes.C_ID) "+
        " INNER JOIN ES_FCs ON ES_Causes.ID = ES_FCs.C_ID "+
        " WHERE ((ES_FCs.F_ID)="+F_ID+") "+
        " ORDER BY ES_FCs.CF DESC; ");

    while (rs2.next())
    {
        String C_Desc = rs2.getString(1);
        String C_Status = rs2.getString(2);
        out.write("RULE ["+i+"] \n");
        out.write("If ");
        i++;
    }
}

```

```

        DataBase db3=new DataBase();
        db3.Connect("DSIS");
        ResultSet rs3=db3.Query("    SELECT ES_S_Class.Desc, ES_Symptoms.Status, ES_Symptoms.ID "+
            " FROM (ES_S_Class INNER JOIN ES_Symptoms ON ES_S_Class.ID = ES_Symptoms.C_ID) "+
            " INNER JOIN ES_FSs ON ES_Symptoms.ID = ES_FSs.S_ID "+
            " WHERE (((ES_FSs.F_ID)="+F_ID+")) "+
            " ORDER BY ES_Symptoms.ID; ");
        while (rs3.next())
        {
            String S_Desc = rs3.getString(1);
            String S_Status = rs3.getString(2);

            out.write("[ "+S_Desc+" ] = \"+S_Status+"\" and \n");
        } //while (rs3.next)
        db3.disconnect();
        out.write("[ "+C_Desc+" ] = \"+C_Status+"\" and \n");
        out.write("[ "+DProc_Desc+" ] = \"+Outcome+"\" \n");
        out.write("Then [Fault Code] = \"+F_ID+"\" and\n");
        out.write("[Cause] = \"+C_Desc+" is "+C_Status+"\" \n\n");
    } //While rs2.next

    db2.disconnect();

// Fault Symptoms without Causes

    out.write("RULE [ "+i+" ] \n");
    out.write("If ");
    i++;

```

```

        DataBase db3=new DataBase();
        db3.Connect("DSIS");
        ResultSet rs3=db3.Query("    SELECT ES_S_Class.Desc, ES_Symptoms.Status, ES_Symptoms.ID "+
            " FROM (ES_S_Class INNER JOIN ES_Symptoms ON ES_S_Class.ID = ES_Symptoms.C_ID) "+
            " INNER JOIN ES_FSs ON ES_Symptoms.ID = ES_FSs.S_ID "+
            " WHERE (((ES_FSs.F_ID)="+F_ID+")) "+
            " ORDER BY ES_Symptoms.ID; ");
        while (rs3.next())
        {
            String S_Desc = rs3.getString(1);
            String S_Status = rs3.getString(2);

```

```

        out.write("[ "+S_Desc+"] = \"+S_Status+\" and \n");
    } //while (rs3.next)
    db3.disconnect();

    out.write("[ "+DProc_Desc+"] = \"+Outcome+\" \n");
    out.write("Then [Fault Code] = \"+F_ID+\" \n\n");

} //while rs1.next
db1.disconnect();

//-----
// Fault Rules
db2=new DataBase();
db2.Connect("DSIS");
rs2=db2.Query(" SELECT ID, D_Proc, Outcome, C_Proc FROM ES_Faults ");

while (rs2.next())
{
    String F_ID=rs2.getString(1);
    String D_Proc=rs2.getString(2);
    String Outcome=rs2.getString(3);
    String CProc_ID=rs2.getString(4);

    DataBase db3=new DataBase();
    db3.Connect("DSIS");
    ResultSet rs3=db3.Query(" SELECT Description FROM IOs "+
        " WHERE ID = \"+D_Proc+"\"");
    rs3.next();
    String DProc_Desc = rs3.getString(1);
    db3.disconnect();

    DataBase db4=new DataBase();
    db4.Connect("DSIS");
    ResultSet rs4=db4.Query(" SELECT Description FROM IOs "+
        " WHERE ID = \"+CProc_ID+"\"");
    rs4.next();
    String CProc_Desc = rs4.getString(1);
    db4.disconnect();

```

```

        out.write("RULE [" + i + "] \n");
        i++;

        out.write("If [Fault Code] = \"" + F_ID + "\" \n");
        out.write("Then [Fault] = \"The outcome of " + DProc_Desc + " procedure is: " + Outcome + "\" and \n");
        out.write("[Recommendation] = \"" + CProc_Desc + " (Man: " + CProc_ID + ")\" \n \n");

    } //while rs2.next
    db2.disconnect();

// Generate PROMPTS
// Symptoms PROMPTS
    db1=new DataBase();
    db1.Connect("DSIS");

    rs1=db1.Query(" SELECT ID, Desc FROM ES_S_Class; ");
    while (rs1.next())
    {
        int C_ID=rs1.getInt(1);
        String C_Desc=rs1.getString(2);

        out.write("PROMPT [" + C_Desc + "] MultChoice \n");
        out.write("\"" + C_Desc + "? \" \n");

        db2=new DataBase();
        db2.Connect("DSIS");

        rs2=db2.Query(" SELECT Status FROM ES_Symptoms " +
            "WHERE C_ID = " + C_ID + ";");
        while (rs2.next())
        {
            String Status=rs2.getString(1);
            out.write("\"" + Status + "\" \n");
        } //While rs2.next
        db2.disconnect();
        out.write("\"Normal\" \n \n");
    } //While rs1.next
    db1.disconnect();

```

```

// TROUBLE CAUSES PROMPTS
db1=new DataBase();
db1.Connect("DSIS");

rs1=db1.Query(" SELECT DISTINCT c.ID, c.Desc FROM ES_C_Class c, ES_Causes h, ES_FCs t "+
              " WHERE c.ID = h.C_ID and h.ID = t.C_ID; ");
while (rs1.next())
{
    int C_ID=rs1.getInt(1);
    String C_Desc=rs1.getString(2);

    if (C_ID != 0){
        out.write("PROMPT ["+C_Desc+"] MultChoice CF \n");
        out.write("\\""+C_Desc+"? \\""\n");

        db2=new DataBase();
        db2.Connect("DSIS");

        rs2=db2.Query(" SELECT Status FROM ES_Causes "+
                      "WHERE C_ID = "+C_ID+";");
        while (rs2.next())
        {
            String Status=rs2.getString(1);
            out.write("\\""+Status+"\\""\n");
        } //While rs2.next
        db2.disconnect();
        out.write("\\"Normal\\""\n\n");
    } // if C_ID !=0
} //While rs1.next
db1.disconnect();

// Diagnosis PROMPTS
String D_Proc_Desc="";
db1=new DataBase();
db1.Connect("DSIS");

rs1=db1.Query(" SELECT DISTINCT D_Proc FROM ES_Faults f ; ");
while (rs1.next())
{
    String D_Proc = rs1.getString(1);

```



```

        DataBase db3=new DataBase();
        db3.Connect("DSIS");
        ResultSet rs3=db3.Query(" SELECT Description FROM IOs "+
                                " WHERE ID = \''+D_Proc+'\"");
        if (rs3.next()) {
            D_Proc_Desc=rs3.getString(1);
        } else D_Proc_Desc = D_Proc;
        db3.disconnect();
        out.write("PROMPT ["+D_Proc_Desc+"] MultChoice \n");
        out.write("\nYou should "+D_Proc_Desc+" (Man:"+D_Proc+"), for the following fault(s): \n\n");

        db2=new DataBase();
        db2.Connect("DSIS");

        rs2=db2.Query(" SELECT Outcome FROM ES_Faults "+
                      " WHERE D_Proc Like \''+D_Proc+'\"");
        while(rs2.next())
        {
            String Status=rs2.getString(1);
            out.write("\n"+Status+"\n \n");
        } //While rs2.next
        db2.disconnect();
        out.write("\nNormal\n\n\n");
    } //While rs1.next
    db1.disconnect();

    out.write("\n\n");
    out.write("GOAL [Cause]\n");
    out.write("GOAL [Fault]\n");
    out.write("GOAL [Recommendation]\n");
    out.write("MINCF 60");
    out.close();
} catch(SQLException ex) {
    System.err.println("SQLException: " + ex.getMessage());
}

}
}

```

E.2 the Complete Rule-Based KB Generated in e2gLite ES Shell Format

```
RULE [1]
If [Braking effect] = "decreased or missing" and
[Quantity of brake fluid] = "considerably less than average" and
[Brake fluid visual inspection] = "fluid leaking" and
[Brake pipes last checked] = "> 6 months" and
[Check the brake fluid pipes] = "Damaged"
Then [Fault Code] = "1" and
[Cause] = "Brake pipes last checked is > 6 months"
```

```
RULE [2]
If [Braking effect] = "decreased or missing" and
[Quantity of brake fluid] = "considerably less than average" and
[Brake fluid visual inspection] = "fluid leaking" and
[Working environment] = "very rough and bumpy" and
[Check the brake fluid pipes] = "Damaged"
Then [Fault Code] = "1" and
[Cause] = "Working environment is very rough and bumpy"
```

```
RULE [3]
If [Braking effect] = "decreased or missing" and
[Quantity of brake fluid] = "considerably less than average" and
[Brake fluid visual inspection] = "fluid leaking" and
[Check the brake fluid pipes] = "Damaged"
Then [Fault Code] = "1"
```

```
RULE [4]
If [Pedal feel] = "soft" and
[Braking effect] = "truck skids" and
[Brake cylinder last replaced] = "> 2 Years" and
[Check brake cylinder] = "Damaged brake cylinder"
Then [Fault Code] = "3" and
[Cause] = "Brake cylinder last replaced is > 2 Years"
```

```
RULE [5]
If [Pedal feel] = "soft" and
[Braking effect] = "truck skids" and
[Working environment] = "very rough and bumpy" and
[Check brake cylinder] = "Damaged brake cylinder"
Then [Fault Code] = "3" and
[Cause] = "Working environment is very rough and bumpy"
```

```
RULE [6]
If [Pedal feel] = "soft" and
[Braking effect] = "truck skids" and
[Check brake cylinder] = "Damaged brake cylinder"
Then [Fault Code] = "3"
```

```
RULE [7]
If [Pedal feel] = "hard" and
[Braking effect] = "decreased or missing" and
[Condition of the brake fluid] = "contaminated/low quality" and
[Check brake piston] = "Blocked piston"
```

Then [Fault Code] = "4" and
[Cause] = "Condition of the brake fluid is contaminated/low quality"

RULE [8]
If [Pedal feel] = "hard" and
[Braking effect] = "decreased or missing" and
[Check brake piston] = "Blocked piston"
Then [Fault Code] = "4"

RULE [9]
If [Pedal feel] = "soft" and
[Braking effect] = "decreased or missing" and
[Quantity of brake fluid] = "little less than average" and
[Brake shoes last replaced] = "> 90 days" and
[Check the Servo Brake] = "Worn-out brake shoe"
Then [Fault Code] = "5" and
[Cause] = "Brake shoes last replaced is > 90 days"

RULE [10]
If [Pedal feel] = "soft" and
[Braking effect] = "decreased or missing" and
[Quantity of brake fluid] = "little less than average" and
[Weather condition] = "very hot" and
[Check the Servo Brake] = "Worn-out brake shoe"
Then [Fault Code] = "5" and
[Cause] = "Weather condition is very hot"

RULE [11]
If [Pedal feel] = "soft" and
[Braking effect] = "decreased or missing" and
[Quantity of brake fluid] = "little less than average" and
[Check the Servo Brake] = "Worn-out brake shoe"
Then [Fault Code] = "5"

RULE [12]
If [Pedal feel] = "soft" and
[Braking effect] = "truck skids" and
[Working environment] = "very rough and bumpy" and
[Check brake disk] = "Worn-out disk"
Then [Fault Code] = "6" and
[Cause] = "Working environment is very rough and bumpy"

RULE [13]
If [Pedal feel] = "soft" and
[Braking effect] = "truck skids" and
[Check brake disk] = "Worn-out disk"
Then [Fault Code] = "6"

RULE [14]
If [Pedal feel] = "hard" and
[Braking effect] = "decreased or missing" and
[Working environment] = "dirty / dusty" and
[Check the brake pedal] = "Blocked bushings"
Then [Fault Code] = "7" and
[Cause] = "Working environment is dirty / dusty"

RULE [15]

If [Pedal feel] = "hard" and
[Braking effect] = "decreased or missing" and
[Weather condition] = "very cold" and
[Check the brake pedal] = "Blocked bushings"
Then [Fault Code] = "7" and
[Cause] = "Weather condition is very cold"

RULE [16]
If [Pedal feel] = "hard" and
[Braking effect] = "decreased or missing" and
[Check the brake pedal] = "Blocked bushings"
Then [Fault Code] = "7"

RULE [17]
If [Braking effect] = "truck skids" and
[Check the brake clearance] = "Different in the two wheels"
Then [Fault Code] = "8"

RULE [18]
If [Parking brake] = "does not operate" and
[Check the hand brake rope] = "Incorrectly adjusted length"
Then [Fault Code] = "9"

RULE [19]
If [Parking brake] = "does not operate" and
[Hand brake rope last changed] = "> 2 Years" and
[Check the hand brake rope] = "Broken"
Then [Fault Code] = "10" and
[Cause] = "Hand brake rope last changed is > 2 Years"

RULE [20]
If [Parking brake] = "does not operate" and
[Check the hand brake rope] = "Broken"
Then [Fault Code] = "10"

RULE [21]
If [Pedal feel] = "hard" and
[Condition of the brake fluid] = "contaminated/low quality" and
[Check the main cylinder rubber glands] = "Swollen"
Then [Fault Code] = "11" and
[Cause] = "Condition of the brake fluid is contaminated/low quality"

RULE [22]
If [Pedal feel] = "hard" and
[Check the main cylinder rubber glands] = "Swollen"
Then [Fault Code] = "11"

RULE [23]
If [Fault Code] = "1"
Then [Fault] = "The outcome of Check the brake fluid pipes procedure
is: Damaged" and
[Recommendation] = "Replace damaged brake fluid pipes (Man:10.10.9.1)"

RULE [24]
If [Fault Code] = "3"
Then [Fault] = "The outcome of Check brake cylinder procedure is:
Damaged brake cylinder" and

[Recommendation] = "Replace the brake cylinder (Man:10.10.9.4)"

RULE [25]

If [Fault Code] = "4"

Then [Fault] = "The outcome of Check brake piston procedure is: Blocked piston" and

[Recommendation] = "Replace the brake piston (Man:10.10.9.7)"

RULE [26]

If [Fault Code] = "5"

Then [Fault] = "The outcome of Check the Servo Brake procedure is: Worn-out brake shoe" and

[Recommendation] = "Replace the brake shoe (Man:10.10.9.2)"

RULE [27]

If [Fault Code] = "6"

Then [Fault] = "The outcome of Check brake disk procedure is: Worn-out disk" and

[Recommendation] = "Replace the brake disk (Man:10.10.9.3)"

RULE [28]

If [Fault Code] = "7"

Then [Fault] = "The outcome of Check the brake pedal procedure is: Blocked bushings" and

[Recommendation] = "Unblock brake pedal bushings (Man:10.10.9.5)"

RULE [29]

If [Fault Code] = "8"

Then [Fault] = "The outcome of Check the brake clearance procedure is: Different in the two wheels" and

[Recommendation] = "Adjust the brake clearance (Man:10.10.9.8)"

RULE [30]

If [Fault Code] = "9"

Then [Fault] = "The outcome of Check the hand brake rope procedure is: Incorrectly adjusted length" and

[Recommendation] = "Adjust the hand brake rope (Man:10.10.9.9)"

RULE [31]

If [Fault Code] = "10"

Then [Fault] = "The outcome of Check the hand brake rope procedure is: Broken" and

[Recommendation] = "Replace the hand brake rope (Man:10.10.9.10)"

RULE [32]

If [Fault Code] = "11"

Then [Fault] = "The outcome of Check the main cylinder rubber glands procedure is: Swollen" and

[Recommendation] = "Replace rubber glands in the main cylinder (Man:10.10.9.6)"

PROMPT [Pedal feel] MultChoice

"Pedal feel? "

"soft"

"hard"

"Normal"

PROMPT [Braking effect] MultChoice

"Braking effect? "
"decreased or missing"
"truck skids"
"Normal"

PROMPT [Parking brake] MultChoice

"Parking brake? "
"does not operate"
"Normal"

PROMPT [Quantity of brake fluid] MultChoice

"Quantity of brake fluid? "
"little less than average"
"considerably less than average"
"Normal"

PROMPT [Brake fluid visual inspection] MultChoice

"Brake fluid visual inspection? "
"fluid leaking"
"Normal"

PROMPT [Condition of the brake fluid] MultChoice CF

"Condition of the brake fluid? "
"contaminated/low quality"
"Normal"

PROMPT [Working environment] MultChoice CF

"Working environment? "
"dirty / dusty"
"very rough and bumpy"
"Normal"

PROMPT [Weather condition] MultChoice CF

"Weather condition? "
"very hot"
"very cold"
"Normal"

PROMPT [Brake shoes last replaced] MultChoice CF

"Brake shoes last replaced? "
"> 90 days"
"Normal"

PROMPT [Brake cylinder last replaced] MultChoice CF

"Brake cylinder last replaced? "
"> 2 Years"
"Normal"

PROMPT [Hand brake rope last changed] MultChoice CF

"Hand brake rope last changed? "
"> 2 Years"
"Normal"

PROMPT [Brake pipes last checked] MultChoice CF

"Brake pipes last checked? "
"> 6 months"

"Normal"

PROMPT [Check the brake fluid pipes] MultChoice

"You should Check the brake fluid pipes (Man:10.9.9.1), for the following fault(s): "

"Damaged"

"Normal"

PROMPT [Check the brake pedal] MultChoice

"You should Check the brake pedal (Man:10.9.9.3), for the following fault(s): "

"Blocked bushings"

"Normal"

PROMPT [Check the main cylinder rubber glands] MultChoice

"You should Check the main cylinder rubber glands (Man:10.9.9.4), for the following fault(s): "

"Swollen"

"Normal"

PROMPT [Check the brake clearance] MultChoice

"You should Check the brake clearance (Man:10.9.9.5), for the following fault(s): "

"Different in the two wheels"

"Normal"

PROMPT [Check the hand brake rope] MultChoice

"You should Check the hand brake rope (Man:10.9.9.6), for the following fault(s): "

"Incorrectly adjusted length"

"Broken"

"Normal"

PROMPT [Check brake cylinder] MultChoice

"You should Check brake cylinder (Man:10.9.9.7), for the following fault(s): "

"Damaged brake cylinder"

"Normal"

PROMPT [Check brake disk] MultChoice

"You should Check brake disk (Man:10.9.9.8), for the following fault(s): "

"Worn-out disk"

"Normal"

PROMPT [Check brake piston] MultChoice

"You should Check brake piston (Man:10.9.9.9), for the following fault(s): "

"Blocked piston"

"Normal"

PROMPT [Check the Servo Brake] MultChoice

"You should Check the Servo Brake (Man:5.9.9.1), for the following fault(s): "

"Worn-out brake shoe"

"Normal"

GOAL [Cause]
GOAL [Fault]
GOAL [Recommendation]
MINCF 60

REFERENCES

Aikat S., and Aikat D. (1996); Shared Techniques Between Print and Online Documentation; Proceedings of the 14th Annual International Conference on Computer Documentation SIGDOC'96, Research Triangle Park, NC, USA, pp. 125-129.

Bezanson W. R. (1995); Performance Support Online, Integrated Documentation and Training; Proceedings of the 13th Conference on Engineering from Chaos: Solutions for the Growing Complexity of our Jobs; Sept 30-Oct 3, 1995, Savannah, GA, USA.

Bieber M. (1998); Hypertext and Web Engineering; The proceedings of the 9th ACM conference on Hypertext & Hypermedia "HyperText 98", Pittsburgh-Pennsylvania. pp. 277-278.

Boley D., Gini M., Gross R., Han E. S., Hastings K., Karypis G., Kumar V., Mobasher B., and Moore J. (1999); Partitioning-based Clustering for Web Document Categorization; Decision Support Systems, Vol. 27, pp. 320-341

Bondendorf F., and Langer K. (1997); Hypermedia Navigation Support by Fuzzy Logic and Neural Networks; Proceedings of the IEEE International Conference on

Intelligent Processing Systems; Vol. 1, pp. 180-184.

Bonfigli M. E., Casadei G., and Salomoni P. (2000); Adaptive Intelligent Hypermedia using XML; Proceedings of SAC 2000 – ACM Symposium on Applied Computing; Como, Italy; Vol. 2, pp. 922-926.

Broder A. Z., Glassman S. C., Manasse M. S. and Zweig G. (1997); Syntactic Clustering of the Web; Computer Networks and ISDN Systems, Vol. 29, pp. 1157-1166

Brusilovsky P. (1996); Methods and Techniques of Adaptive Hypermedia; User Modeling and User-Adapted Interaction, Vol. 6, pp 87-129.

Brusilovsky P., and Cooper D. W. (1999); ADAPTS: Adaptive Hypermedia for a Web-based Performance Support System; Proceedings of the 2nd Workshop on Adaptive Systems and User Modeling on the WWW; Toronto, Canada, May 11-14; pp 41-47.

Brusilovsky P., and Cooper D. W. (2002); Domain, Task, and User Models for an Adaptive Hypermedia Performance Support System; Proceedings of the International Conference on Intelligent User Interfaces; San Francisco, CA, USA.

BS 4884 (1992); Technical Manuals, Part 1: Specification for Presentation of Essential Information; British Standards Institution BSI, London.

BS 4884 (1993a); Technical Manuals, Part 2: Guide to Content; British Standards Institution BSI, London.

Cantando M. (1996); Vision 2000: Multimedia Electronic Performance Support Systems; Proceedings of the 14th Annual International Conference on Marshalling New Technological Forces: Building a Corporate, Academic, and User Oriented Triangle; Research Triangle - United States; Oct 19-22.

Ceri S., Fraternali P., and Bongio A. (2000); Web Modelling Language (WebML): a Modelling Language for Designing Web Sites; Computer Networks, Vol 33, Issue 1, 2000, Pages 137-157.

Chung C., Shih T.K., and Kuo C. (1996); On the Construction of Intelligent Multimedia Presentations; Information Sciences, Vol. 89, No. 1-2, pp. 131-155.

Coda F., Ghezzi C., Vigna G., and Garzotto F (1998); Towards a Software Engineering Approach to Web Site Development; Proceedings of the 9th IEEE International Conference, Workshop on Software Specification and Design (IWSSD); Japan, 1998.

Coffey J.W., Canas A. J., Hill G., Carff R., Reichherzer T., and Suri N. (2003); Knowledge Modelling and the Creation of El-Tech: a Performance Support and Training System for Electronic Technicians; Expert Systems with Applications, Vol.

25, pp 483-492.

Coleman V. (1991); Hardcopy to Hypertext: Putting a Technical Manual Online; Proceedings of the ACM 9th Annual International Conference on Systems Documentation; Chicago - Illinois, United States; pp. 67-72.

Cooper D.W., Veitch F. P., Anderson M. M., and Clifford M. J. (1999); Adaptive Diagnostic And Personalised Technical Support (ADAPTS); Proceedings of the IEEE Aerospace Conference, Aspen, Colorado; Paper No 4.602.

Csinger A., Booth K. S., and Poole D. (1995); AI Meets Authoring: User Models for Intelligent Multimedia; Artificial Intelligence Review, Vol. 8, No. 5-6, pp. 447-468.

De Bra P. (1999); Design Issues in Adaptive Web-Site Development; Proceedings of the 2nd Workshop on Adaptive Systems and User Modelling on the Web, 8th International World Wide Web Conference, Toronto, Canada.

De Bra P., and Calvi L. (1998); AHA: a generic Adaptive Hypermedia System; Proceedings of the 2nd Workshop on Adaptive Hypertext and Hypermedia "HYPERTEXT 98", Pittsburgh, USA.

Deitel H. M., Deitel P. J., and Nieto T. R. (2000); Internet & World Wide Web – How to Program-; Prentice-Hall, Inc., USA.

Desmarais M. C., Leclair R., Fiset J., and Talbi H. (1997); Cost-Justifying Electronic Performance Support Systems; Communications of the. ACM, Vol. 40, No. 7, Jul. 1997, pp. 39 – 48.

Ding Y., Fensel D., Klein M., and Omelayenko B. (2002); The Semantic Web: Yet Another Hip?; Proceedings of the Data and Knowledge Engineering Journal, Vol. 41, pp 205 227.

e2gLite (2003); ES Shell from eXpertise2Go.com, [WWW] <URL: <http://www.expertise2go.com/>> [accessed Sept 2004]

Fernandez M., Florescu D., Levy A., and Suciu D. (2000); Declarative Specification of Web Sites with Strudel; Published in VLDB Journal , no. 9(1) , pp. 38-55

Fernandez M., Suciu D., and Tatarinov I. (1999); Declarative Specification of Data-Intensive Web Sites; In USENIX Conference on Domain-Specific Languages; Austin, Texas (USA).

Fischer O. and Horn R. (1997); Electronic Performance Support Systems; Communications of the ACM, July 1997, Vol. 40, N0.7.

Fraisse S., Nanard J., and Nanard M. (1996); Generating Hypermedia from Specifications by Sketching Multimedia Templates; Proceedings of ACM

Multimedia 96, Boston, MA, USA, pp. 353-363.

Francisco-Revilla L., and Shipman F.M. (2000); Adaptive Medical Information Delivery Combining User, Task, and Situation Models. Intelligent User Interfaces, New Orleans LA USA 2000; pp. 94-97.

Fraternali P., and Paolini P. (2000); Model-Driven Development of Web Applications: The Autoweb System; ACM Transactions on Information Systems, Vol. 28, No. 4, October 2000, Pages 323-382.

Fraternali, P. (1999); Tools and Approaches for Developing data-intensive Web Applications: A Survey; ACM Computing Surveys, Vol. 31, No. 3, Sep. 1999, pp 227 – 263.

Garlatti S., Iksal S., and Kervella P. (1999); Adaptive On-Line Information System by means of a Task Model and Spatial Views; Proceedings of the 2nd Workshop on Adaptive Systems and User Modeling on the WWW; Toronto (WWW8) and Banff (UM 99), pp. 59-66

Garzotto, F., Paolini P., and Schwabe D. (1993); HDM – A Model-Based Approach to Hypertext Application Design; ACM Transactions on Information System, Vol. 11, No. 1, January 1993.

Gomez J., Cachero C., and Pastor O. (2001); Conceptual Modeling of Device-

Independent Web Applications; IEEE Multimedia, Vol. 8, Issue 2, pp 26-39

Graham M. (1997); Performance-Based Documentation; Proceedings of the IEEE International Professional Communication Conference, SaltLake City, UT, USA, pp. 275-282.

Halasz H., and Schwartz M. (1994); The Dexter Hypertext Reference Model; Communications of the ACM, February 1994, Vol. 37 Issue 2, pp. 30-39.

Harold E. R. (1999); XML Bible; IDG Books Worldwide, Inc., An International Data Group Company, USA.

He X., Zha H., Ding C. H. Q., and Simon H. D. (2002); Web Document Clustering using Hyperlink Structures; Computational Statistics & Data Analysis, Vol. 41, Issue 1, pp. 19-45

IEEE (1990); IEEE Standard Glossary of Software Engineering Terminology, Std 610.12-1990

Isakowitz T., Stohr E. A., and Balasubramanian P. (1995); RMM: A Methodology for Structured Hypermedia Design; Communications of the ACM 38, Vol. 8, Aug. 1995, pp. 34 – 44.

ISO 8879 (1986); Information Processing – Text and Office Systems – Standard Generalized Markup Language (SGML), International Organisation for

standardisation, Geneva.

JDK (1.3.0); Java™ 2 Runtime Environment, Standard Edition (build 1.3.0-C); Sun Microsystems Inc.

JSWDK (1.0.1). Java Server Web Development Kit version 1.0.1; Sun Microsystems Inc., [WWW] <URL: <http://java.sun.com/>> [accessed Sept. 2004]

Kappe F. (1999); Hyperwave Information Server; Technical White Paper, Hyperwave™ Information Management Inc., [WWW] <URL: <http://www.hyperwave.com/>> [accessed Sept. 2004], Version 1.4, November 10, 1999.

Kemp B., and Buckner K. (1999); A Taxonomy of Design Guidance for Hypermedia Design, *Interacting with Computers*, Volume 12, Issue 2, November 1999, pp. 143-160.

Kim H., Shin H. G., and Chang J. W. (1996); OOHS: An Object-Oriented Hypermedia System; *Proceedings of the 20th Annual International Conference on Computer Software and Applications*, Seoul, S. Korea, pp. 496-501.

Klusck M. (2001); Information Agent Technology for the Internet: A Survey; *Proceedings of the Data and Knowledge Engineering Journal*, Vol. 36, pp. 337-372.

Langer K., and Bodendorf F. (1994); A System Architecture for Flexible, Knowledge-Based, Multimedia CBT-Applications; Proceedings of the IEEE first International Conference of Multimedia Engineering Education, pp. 20-29.

Lee K., Lee Y. K., and Berra P. B. (1997); Management of Multi-Structured Hypermedia Documents: A Data Model, Query Language, and Indexing Scheme; Multimedia Tools and Applications, Vol. 4, No. 2, pp. 199-223.

Levy A. Y., and Weld D. S. (2000); Intelligent Internet Systems; Proceedings of the Artificial Intelligence Journal, Vol. 118, pp. 1-14.

McGraw K. L. (1997); Defining and Designing the Performance-Centered Interface, Moving Beyond the User-Centered Interface; ACM Interactions, Vol. 4, No. 2, pp. 19-26, March/April 1997.

Mchugh J. A. (1990); Algorithmic Graph Theory; Prentice-Hall, Inc.; Englewood Cliffs, N. J.

Milosavljevic M., Vitali F., and Watters C. (1999); Introduction; Position Paper for Workshop on Virtual Documents, Hypertext Functionality and the Web at the 8th International World Wide Web Conference, Toronto – Canada, [WWW] <URL: <http://www.cs.unibo.it/~fabio/VD99/index.html> > [accessed Sept. 2004]

MS-Access (1996); Microsoft Access 97; Microsoft Corporation, USA.

Negnevitsky M. (2002); Artificial Intelligence: A Guide to Intelligent Systems; 1st Edition, Pearson Education Limited, England, UK.

Oxborrow E. (1989); Databases and Database Systems Concepts and Issues; 2nd Edition; A Chartwell-Bratt Student Text, Sweden.

Oxford Dictionary (1998); The Oxford Quick Reference Dictionary, Revised Edition; OXFORD UNIVERSITY PRESS 1996, 1998.

Patel S. A. and Kamrani A. K. (1996); Intelligent Decision Support System for Diagnosis and Maintenance of Automated Systems; Proceedings of the Computers Industrial Engineering Journal, Vol. 30, No. 2, pp. 297-319.

Paulo F.B., Augusto M., Turine S., Cristina M., De Oliviera F., and Masiero P.C. (1998); XHMBS: A Formal Model to Support Hypermedia Specification; The Proceedings of the 9th ACM Conference on Hypertext & Hypermedia "HyperText 98", Pittsburgh-Pennsylvania. pp. 161-170.

Perkowitz M., and Etzioni O. (1999); Towards Adaptive Web Sites: Conceptual Framework and Case Study; Computer Networks, Vol. 31, pp. 1245-1258.

Perkowitz M., and Etzioni O. (2000); Towards Adaptive Web Sites: Conceptual Framework and Case Study; Artificial Intelligence, Vol. 118, Issue 1-2, pp. 245-275.

Pham D. T. and Setchi R. M. (2001); Authoring Environment for Documentation Development; Proceedings of the Institution of Mechanical Engineers, Part B, Vol. 215, pp. 877-882.

Pham D. T. and Setchi R. M. (2003); Case-Based Generation of Adaptive Product Manuals; Proceedings of the Institution of Mechanical Engineers, Part B, Vol. 217, pp. 313-322.

Pham D. T., Dimov S. S., and Huneiti A. M. (2003); Semantic Data Model For Product Support Systems. Proceedings of the IEEE International Conference on Industrial Informatics INDIN 2003, August 21-24, 2003, Banff, Alberta, Canada.

Pham D. T., Dimov S. S., and Setchi R. M. (1999); Intelligent Product Manuals; Proceedings of the Institution of Mechanical Engineers, Part B, Vol. 213, Part I, pp. 65-76.

Pham D.T., Dimov S.S. and Peat, B.J. (2000); Intelligent Product Manuals; Proceedings of the Institution of Mechanical Engineers, Part B, Journal of Engineering Manufacture, Vol. 214, No. B5, pp. 411-419.

Price J. (1997); Introduction: Special Issue on Structuring Complex Information for Electronic Publication; IEEE Transaction on Professional Communication, Vol. 40, No. 2, pp. 69-77.

Raybould B. (1995); Performance Support Engineering: An Emerging Development Methodology for Enabling Organisational Learning; Performance Improvement Quarterly, Vol. 8, No. 1 (1995), pp. 7-22.

Raybould B. (2000); Building Performance-Centred Web-Based Systems, Information Systems, And Knowledge Management Systems In the 21st Century; Performance Improvement Quarterly, Vol. 39, No.6, 2000.

Reiter E., Mellish C., and Levine J. (1995); Automatic Generation of Technical Documentation; Proceedings of the Applied Artificial Intelligence Journal, Vol. 9, pp. 259-287.

Rolston D. W. (1988); Principles of Artificial Intelligence and Expert Systems Development. McGraw-Hill, Singapore, 1988.

Rossi G., Schwabe D., and Lyardet F. (2001); Web Application Models are more than Conceptual Models". Lecture Notes in Computer Science 1727, pp. 239-252.

Rossi G., Schwabe D., Lucena C.J.P., and Cowan D.D. (1995); An Object-Oriented Model for Designing the Human-Computer Interface of Hypermedia Applications; Proceedings of the International Workshop on Hypermedia Design (IWH'D'95), Springer Verlag Workshops in Computing Series, (available at [WWW] <URL: ftp://ftp.inf.puc-rio.br/pub/docs/techreports/95_07_rossi.ps.gz> [accessed Sept 2004]).

Saarela J., Turpeinen M., Puskala T., Korkea-Aho M., and Sulonen R. (1997); Logical Structure of a Hypermedia Newspaper; Information Processing and Management, Vol. 33, No. 5, pp. 599-614.

Schnase J. L., Leggett J. J., Hicks D. L., and Szabo R. L. (1993); Semantic Data Modelling of Hypermedia Associations; ACM Transactions on Information Systems, Vol. 11, No. 1, January 1993, pp 27-50.

Schwabe D., and Rossi G. (1998); An Object Oriented Approach to Web-Based Applications Design; Theory and Practice of Object Systems, Vol 4, Issue 4, 1998, pp. 207-225

Segor C., and Gaedke M. (2000): Crossing the Gap - From Design to Implementation in Web-Application Development; Information Resources Management Association International Conference 2000, Anchorage, USA, May 21-24, 2000.

Setchi R. M. (2000); Enhanced Product Support through Intelligent Product Manuals; Ph.D. Thesis Submitted to the University of Wales Cardiff, School of Engineering (March 2000).

Shih T. K. and Kuo C. H. (1996); Database Support for Intelligent Tutoring Software. Proceedings of the IEEE International Conference on Multimedia

Engineering Education, Melbourne, Australia, pp. 297-302.

Sleight D. A. (1993); Types of Electronic Performance Support Systems: Their Characteristics and Range of Designs; Educational Psychology Michigan State University, [WWW] <URL: http://www.msu.edu/~sleightd/epss_copy.html> [accessed Sept. 2004]

Sommerville I. (2001); Software Engineering; International Computer Science Series; Pearson Education Limited; 6th Edition, USA.

Staab S., Angele J., Decker S., Erdmann M., Hotho A., Maedche A., Schnurr H. P., Studer R., and Sure Y. (2000); Semantic Community Web Portals; Computer Networks, Volume 33, pp. 473-491

Suh W., and Lee H. (2001); A Methodology for Building Content-Oriented Hypermedia Systems, Journal of Systems and Software, Vol. 56, Issue 2, pp. 115-131

Takahashi K., and Liang E. (1997); Analysis and design of Web-based information systems; Computer Networks and ISDN Systems, Volume 29, Issues 8-13, September 1997, pp. 1167-1180

Ter Bekke J. H. (1992); Semantic Data Modelling; Prentice Hall International Ltd; UK

Thibeau J. (1996); Making Information Work on the World Wide Web. Proceedings of the 43rd Annual Conference of the Society for Technical Communication, Washington DC, USA, pp. 374-378.

Thomson J.R., Greer J., and Cooke J (2001); Automatic Generation of Instructional Hypermedia with APHID; Interacting with Computers, Volume 13, Issue 6, August 2001, Pages 631-654.

Tucker H., and Harvey B. (1997); SGML Documentation Objects within the Step Environment; SGML Europe, Barcelona, (available at [WWW] <URL: <http://www.eccnet.com/papers/step.html> > [accessed Sept 2004]).

Vassileva J. (1996); A Task-Centered Approach for User Modelling in a Hypermedia Office Documentation System; User Modelling and User Adapted Interaction, Vol. 6.

Ventura C. A. (2000); Why Switch From Paper to Electronic Manuals?; Proceedings of the ACM Conference on Document Processing Systems, January 2000, pp. 111-116.

W3C (2000); World Wide Web Consortium, Extensible Markup Language (XML) 1.0 (Second Edition) ; W3C Recommendation, 6 October 2000 [WWW] <URL: <http://www.w3.org/>> [accessed Sept. 2004]

Wang W., and Rada R. (1995); Experiences With Semantic Net Based Hypermedia; International Journal on Human-Computer Studies, Vol. 43:3, pp. 419-439

Wang W., and Rada R. (1998); Structured Hypertext with Domain Semantics; ACM Transactions on Information. Systems, Vol. 16, No. 4, Oct. 1998, pp. 372 – 412

White M. (1998); Designing Dynamic Hypertext; Proceedings of the 2nd Workshop on Adaptive Hypertext and Hypermedia, HYPERTEXT 98, Pittsburgh, USA, 1998.

Wilkinson R., and Smeaton A. (1999); Automatic Link Generation; ACM Computing Surveys, Vol. 31, No. 4es, December 1999.

Wilson R. J. (1986); Introduction to Graph Theory - 3rd Edition; Longman Scientific & Technical; London.

