Rule-Based Semantic Sensing Platform for Activity Monitoring

A thesis submitted in partial fulfillment

of the requirement for the degree of Doctor of Philosophy

Przemyslaw Woznowski

October 2013

Cardiff University
School of Computer Science & Informatics

Declaration

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

Signed ..... (candidate)

Date .....

Statement 1

This thesis is being submitted in partial fulfilment of the requirements for the degree of PhD.

Signed ..... (candidate)

Date .....

Statement 2

This thesis is the result of my own independent work/investigation, except where otherwise stated. Other sources are acknowledged by explicit references.

Signed ..... (candidate)

Date .....

Statement 3

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

Signed ..... (candidate)

Date .....

To my family and my collaborators
for their patience and support.

Abstract

Sensors are playing an increasingly important role in our lives, and for these devices to perform to their maximum potential, they need to work together. A single device can provide a single service or a fixed set of services but, when combined with other sensors, different classes of applications become implementable. The vital criterion for this to happen is the ability to bring information from all sensors together, so that all measured physical phenomena can contribute to the solution. Mediation between applications and physical sensors is the responsibility of sensor network middleware (SNM). Rapid growth in the kinds of sensors and applications for sensors/sensor systems, and the consequent importance of sensor network middleware has raised the need to relatively rapidly build engineering applications from those components.

A number of SNM exist, each of which attempts to solve the sensor integration problem in a different way. These solutions, based on their 'closeness' either to sensors or to applications, can be classified as low-level and high-level. Low-level SNM tends not to focus on making application development easy, while high-level SNM tends to be 'locked-in' to a particular set of sensors. We propose a SNM suitable for the task of activity monitoring founded on rules and events, integrated through a semantic event model. The proposed solution is intended to be open at the bottom – to new sensor types; and open at the top – to new applications/user requirements.

We show evidence for the effectiveness of this approach in the context of two pilot studies in rehabilitation monitoring – in both hospital and home environment. Moreover, we demonstrate how the semantic event model and rule-based approach promotes verifiability and the ability to validate the system with domain experts.

Acknowledgements

Many thanks to my supervisor Professor Alun Preece for his never-ending patience, constant support and guidance. I'm also really grateful to my second supervisor Professor Robert van Deursen who guided me through the world of clinical research. If not for them and for the great amount of support from our collaborators: Arshi Iqbal (PhD student, School of Healthcare Sciences, Cardiff University), Allison Cooper (project supervisor, College of Human and Health Sciences, Swansea University); the two pilot studies presented in Chapter 5 would never take place. I want to thank them all for all their effort and support.

I would also like to thank my final year project supervisor Professor Roger Whitaker, who convinced me not to give up on my dreams. If not his support and guidance I would not be where I am today.

This research was partially funded by the National Institute for Social Care and Health Research (NISCHR) and by the School of Computer Science & Informatics, Cardiff University.

Contents

List of Publications

The work introduced in this thesis is based on the following publications.

- Przemyslaw R Woznowski and Alun Preece. "Rule-Based Semantic Sensing" in *Proceedings of the Doctoral Consortium and Poster Session of the 5th International Symposium on Rules (RuleML 2011@IJCAI)*, p.9-16, 2011.

- Arshi S Iqbal, Przemyslaw R Woznowski. "Automated Measurement of Early Stage Functional Activities after Stroke" in *Research Development Workshop: Stroke Rehabilitation*, Cardiff, March 2012.

- Arshi S Iqbal, Przemyslaw R Woznowski, Allison Cooper, Alun Preece, and Robert van Deursen. "Reliability of a new computerised system to detect walking ability in the early stages of recovery post stroke" in *Physiotherapy Research Society 30th Scientific Meeting*, Sheffield, May 2012.

- Arshi S Iqbal, Przemyslaw R Woznowski, Allison Cooper, Alun Preece, and Robert van Deursen. "Comparing a sensor based Real Time Location System with an Obser- vation Based System and the 10-MWT to determine walking speed in the clinical setting," in *Welsh Stroke Conference*, Newport, June 2012.

- Arshi S Iqbal, Przemyslaw R Woznowski, Allison Cooper, Alun Preece, and Robert van Deursen. "Measuring functional activities of patients in a stroke unit:

Comparison of a sensor based Real Time Location System with the Observational Behaviour Mapping Technique" in *UK Stroke Forum*, Harrogate, December 2012.

- Arshi S Iqbal, Przemyslaw R Woznowski, Allison Cooper, Alun Preece, and Robert van Deursen. "A new computerised system can continuously measure functional activities op patients in a stroke rehabilitation unit" in *European Stroke Conference*, London, May 2013.

List of Figures

List of Tables

List of Algorithms

List of Acronyms

AM Activity Monitoring

AR Activity Recognition

DB Database

DL Data Layer

GSN Global Sensor Networks

GEL Goal Event Layer

HLE High-level Events

HM Home Monitoring

IR Infrared

KB Knowledge Base

LLE Low-level Events

OGC Open Geospatial Consortium

OWL Web Ontology Language

RBS Rule-based System

RBS2P Rule-based Semantic Sensing Platform

RDF Resource Description Framework

RSU Regional Stroke Unit

RTLS Real Time Location System

SEL Sensor Event Layer

SNM Sensor Network Middleware

SS Semantic Streams

SSN Semantic Sensor Network

SSW Semantic Sensor Web

SWE Sensor Web Enablement

UI User Interface

V&V Validation & Verification

WM Working Memory

WSNM Wireless Sensor Network Middleware

Chapter 1

Introduction

This chapter introduces the concept of wireless sensor networks, its applications and the middleware used in this type of network. It also briefly defines the most important concerns for this middleware. It describes rule-based systems and the problem of activity monitoring and activity recognition. It also draws the scope of the presented work, introduces the set of research questions addressed by this thesis and its contributions.

1.1 Data to Decisions

The world we live in evolves quicker than ever, hence we face new technological challenges every day. The inventors of ARPANet could not even anticipate how influential their idea of interconnecting geographically separated nodes would be. Since then, the Internet evolved dramatically and noticeably influenced the way we live our lives these days. However, apart from providing a huge amount of useful information and services, it introduced a new range of problems such as information overload. In 2002 9.1% of the World's population used the Internet and in 2012 this figure went up to 33% i.e. 2.27 billion users[1]. Chief Futurist for Cisco Systems, Dave Evans, quoted in [34] said that "humans generated more data in 2009 than in the previous 5,000 years combined, although a lot if it is useless – comparable to saving all 2,000 photos from your weekend trip to the beach". If these words were said about the year 2009, when

[1] www.themainstreetanalyst.com

only 1.73 billion users had Internet access[2] and smart-phones sales were on the level of around 172 million units[3] then how about today? The reality of today is a little different to a couple of years ago. Nowadays, the number of devices connected to the Internet is incredibly high. It is not only about personal computers or laptops but also about smart-phones and sensors.

The biggest problem with the Internet today is the same as the one mentioned by Dave Evans. He identified [34] that "filtering and sorting the exponential proliferation of data will become more and more important for computers". However, it is not only human-generated data but also automatically generated sensor data that is of concern. How can this data be effectively and efficiently processed in order to support decision making? How do we extract meaning from sensor data and how can we provide the correct information, to the right user, in the right format, at the right time? These are the main concerns of the notion called "Data to Decisions" (D2D). The list of seven Science & Technology (S&T) priorities announced by the US Office of Secretary of Defence (OSD) lists D2D at the top and describes this problem as "proliferation of sensors and large data sets are overwhelming analysts with data because they lack the tools to efficiently process, store, analyse and retrieve it"[4]. One very important aspect of sensor data that the aforementioned quote does not consider is the need to semantically encode the data, so that the knowledge can be effectively shared and reused by making it machine-readable. This class of problems is something that Sensor Network Middleware (SNM) targets.

1.2 Wireless Sensor Networks

In the same sense as we classify Internet networks as WANs and LANs, more network classifications have emerged due to the changing model of computing. Bluetooth tech-

[2] http://royal.pingdom.com/2010/01/22/internet-2009-in-numbers
[3] www.gartner.com/newsroom/id/1306513
[4] http://www.acq.osd.mil/chieftechnologist/publications/docs/OSD%2002073-11.pdf

nology allowed for formulation of the term Personal Area Network (PAN) – a network made up of personal devices belonging to one person. In such a network a master device is responsible for managing the network, and other devices called slaves report to it. Similarly, with an advancement of technology and the increasing presence of sensors in our lives, a need arose for formulation of the term Wireless Sensor Network (WSN). The technopedia online dictionary[5] defines the term WSN as something that "...refers to a group of spatially dispersed and dedicated sensors for monitoring and recording the physical conditions of the environment and organising the collected data at a central location. WSNs measure environmental conditions like temperature, sound, pollution levels, humidity, wind speed and direction, pressure, etc.". Even thought this definition is quite rich it does not exhaustively define the term. A better formulation of this term can be found in [58]. Römer et al. describes the WSN as a network that "...consists of large numbers of cooperating small-scale nodes capable of limited computation, wireless communication, and sensing. By correlating sensor output of multiple nodes, the WSN as a whole can provide functionality that an individual node cannot". This definition is much more complete and highlights a very important point – this type of network can deliver functionality that an individual sensor cannot. In order to do so, the nodes in the network need to have wireless communication capabilities and the network needs to be managed by some sort of middleware. Moreover, from the first definition we learn that the data needs to be organised at a central location. Therefore, apart from the sensing nodes that are limited in computational power, this type of network also requires at least one more powerful node that is capable of 'organising' the data. In order to provide the aforementioned functionality, WSN middleware (WSNM) is needed, which is further discussed in Section 1.4 of this chapter.

[5] www.techopedia.com/definition/25651/wireless-sensor-network-wsn

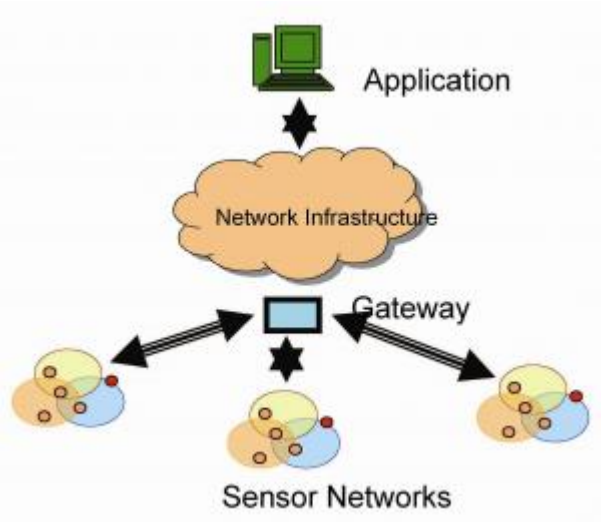


Figure 1.1: Wireless Sensor Networks

In order to understand the problem of WSNs it is crucial to understand the characteristics of a typical node in this type of network. Typically, a node is:

- limited in size and energy,
- subject to failure due to drained batteries and other environmental factors,
- mobile and hence forms a very dynamic network.

The implications of these characteristics are obvious. The power and size of a sensor node limits its computational abilities in terms of: speed, communication range, primary and secondary memory. The fact that nodes are mobile has further consequences, such as communication failures and dynamicity of the network's structure. Moreover, a WSN is often made up of heterogeneous nodes, which does not make the task of software standardisation trivial. From Römer's definition of the WSN flows another very important characteristic i.e. the scalability of the network as it can be made up of a large number of nodes. Hence management and maintenance of a WSN is difficult for software to accomplish.

1.3 Applications

The applications of WSNs are almost endless. One of the major driving factors for this is the availability, size and types of sensors that enter the market. Researchers and industry are quick in finding new application domains for these sensors and for sensor systems. The following are the most popular application areas for WSNs [43], [25]:

- aircraft control,
- robot control,
- video surveillance,
- transportation (traffic monitoring),
- automated and smart homes,
- industrial automation,
- healthcare (activity monitoring),
- air traffic control,
- weather conditions monitoring,
- geophysical monitoring (seismic activity),
- business processes (supply chain management)
- wildlife monitoring,
- ambient assisted living (AAL),
- military.

The above list is not exhaustive as the possible application areas for WSNs are not fixed, as sensor solutions can be applied to many aspects of our life – they “bridge the

gap between the physical and the virtual world" [58]. In all these applications, sensors can be deployed in a number of ways. They can be attached to items or be placed within sensing range e.g. cameras. The placement of sensors can be manual or, for example, they can be deployed from an aircraft or a boat. There is not one recipe or structure to follow, which makes it very challenging for developers to meet the desired requirements of the sensing systems.

Thanks to technological advancements, sensors have become relatively cheap (more accessible) and hence have found their way into the market of personal sensors. Nowadays, there are many sensors/sensor bundles on the market that are relatively cheap and intended for personal use, such as: Fitbit[6], Nike's+ Fuelband[7], Greengoose[8] and many more. What follows from this is the fact that the amount of available APIs for end-users to program with is increasing. Many of these devices communicate their data to the web directly (or via their base-station) to the manufacturer's managed server in order to receive various services, or to a cloud-based service that can log this information (e.g. Xively[9], formerly known as Pachube) and assist in the development of applications. This vision of sensor-derived data feeding the web to provide various services is known as the Internet of Things (IoT) and WSNs are a vital component in realising this vision. Many researchers build their own test-beds and create software to manage the sensor infrastructure. As a result, new devices enter the market either as licensed products or become available for no cost under the General Public License (GPL) e.g. GSN[10]. Therefore, inexperienced users and professionals are all using and developing applications and middleware for sensor systems.

[6] www.fitbit.com/uk

[7] http://store.nike.com/gb/en_gb/?l=shop,pdp,ctr-inline/cid-300/pid-683902/pgid-683903&cp=EUNS_KW_UK_Icons

[8] www.greengoose.com/

[9] https://xively.com/

[10] http://sourceforge.net/apps/trac/gsn/

1.4 Wireless Sensor Network Middleware

WSN middleware (WSNM) is an essential component in a wireless sensor network. Before defining what WSNM is, it is crucial to understand what the word 'middleware' means. There are many definitions of this term in the literature, depending on the computing field it is used for. A very generic definition can be found in [58] "Middleware sits between the operating system and the application". However, such a definition is very broad. Picco et al. [44] define this term as something that "...refers to software and tools that can help hide the complexity and heterogeneity of the underlying hardware and network platforms, ease the management of system resources, and increase the predictability of application executions". Even though this seems to be quite an extensive definition it feels incomplete. Sensor networks are in some aspects similar to distributed systems, and probably a definition from this computing field is the best one to start with, where middleware is "...usually defined as software that lies between the operating system and applications running on each node of the system. Generally, middleware is expected to hide the internal workings and the heterogeneity of the system, providing standard interfaces, abstractions and a set of services, that depends largely on the application" [12].

Similarly to finding a general-purpose definition of a middleware, it is even harder to find a complete definition of WSNM. Many authors write about what the WSNM should do and what its purpose is, however there is no universal definition of this term. Mainly this is due to the fact that this research field is highly dynamic and relatively new. Different people have different visions of what a complete WSNM should do and therefore made different design choices and created very different solutions. A common view in the literature is that "the main purpose of middleware for sensor networks is to support the development, maintenance, deployment, and execution of sensing-based applications" [58]. However to what extent these objectives should be met is up to the programmer. Moreover, the focus is on application development, which cannot be programmed without an ability to connect to sensors. The terms *WSNM* and

Sensor Network Middleware (SNM) are used in this thesis interchangeably, as most modern sensors are wireless and hence the term 'sensor' almost implies its wireless capabilities.

1.5 Middleware Concerns

Due to the complex nature of the problem and the characteristics of WSNs, there are many requirements and challenges that middleware has to meet. Molla et al. lists the following ten WSNM challenges [45]: *abstraction support, data fusion, resource constraints, dynamic topology, application knowledge, programming paradigm, adaptability, scalability, security* and *QoS support*. These challenges are explained in more detail in the context of activity monitoring in Section 2.1. In his survey, he highlights the importance of each of these challenges and discusses six different WSNM, of which none fulfills all of these requirements to a satisfactory level. Moreover, he points out that "existing middleware take quite ad hoc approach to address various challenges". This was the state-of-the-art in 2006, but what about today?

The report on the RUNES (Reconfigurable Ubiquitous Networked Embedded System) project [44] lists the non-functional requirements of a middleware for networked embedded systems as follows: heterogeneity, openness, scalability, failure handling, security, performance, adaptability and feasibility. These requirements do overlap in many areas with Molla's ten challenges, which only highlights the importance of those and the fact that many researchers see the problem of the WSNM similarly.

If we refer back to the definition of the WSNM presented earlier, its purpose is to support: development, maintenance, deployment, and execution of sensing-based applications. Hence, any WSNM should at least aim to satisfy those high-level objectives. Whether they are addressed and to what extent is further discussed in Chapter 2.

1.6 Activity Monitoring & Real-Time Location Systems

WSNM provides sensor data i.e. measurements of some phenomena. The challenge then becomes to use this data in a purposeful manner. If mechanical sensors are to replicate human senses, the obvious purpose for sensor-based systems is to measure/monitor surrounding environment. In many disciplines of life, especially in the military or healthcare domains, monitoring of human agents' activities is a very important task e.g. spotting suspicious enemy's activity or supervising rehabilitation session. So what exactly does the term activity monitoring mean? If the term 'monitoring' means "to test or sample, especially on a regular or ongoing basis"[11] and "to keep close watch over; supervise"[11], then the term activity monitoring simply means to closely watch over, on a regular or ongoing basis, activities of objects/people of interest in some environment. To realise the activity monitoring, either a human is required to observe and classify occurring events, or one needs to equip the environment with sensing hardware and software to perform activity recognition. Many activity monitoring solutions incorporate a real-time location system (RTLS) to sense objects in the environment or make use of visual data.

There is a wide variety of RTLS solutions, amongst which are: active radio frequency identification (RFID), bluetooth (BT), WLAN, ultrasound identification (US-ID), optical, infrared (IR), etc. Hybrid solutions are also possible i.e. active RFID-IR hybrid. Due to technical limitations, some of these systems are not 100% accurate and therefore produce non-error-free data. The main challenges these systems are often facing are: the line-of-sight (LoS) requirements, signal fluctuations, coverage area, precise positioning, object's recognition, etc. Some of these challenges can be met by altering either the hardware, the software, or the environment. Some RTLS technologies used for activity monitoring have distinct characteristics which cannot be overcome due to hardware's limitations. There is no 'silver bullet' for implementing a perfect activity monitoring system and performance of the underlying RTL technology is very much

[11] http://www.thefreedictionary.com/monitoring

dependent on the characteristics of the physical environment and the sensing requirements of the application. In order for a system to perform activity monitoring, apart from RTL hardware which delivers raw sensory data, it needs to contain a software module capable of executing the task of activity recognition.

1.7 Activity Recognition

According to [13] activity recognition (AR) is"the process whereby an actor's behaviour and his/her situated environment are monitored and analysed to infer the undergoing activities. It comprises many different tasks, namely activity modeling, behaviour and environment monitoring, data processing and pattern recognition". There are many approaches for delivering AR and they have various taxonomies. One classification is based on the data type the AR system processes and thus there are two main approaches: *vision-based AR* and *sensor-based AR* [55] [13].

Aggarwal and Ryoo [2] further classify the *vision-based* techniques into: *single-layered* approaches and *hierarchical approaches*; which are then classified further into sub- and sub-sub-categories. Chen and Khalil [13] further divide the *sensor-based* AR approaches into: *wearable sensor-based* and *object-based*. They also point out that "...it is not possible to claim that one approach is superior to the other. The suitability and performance is in the end down to the nature of the type of activities being assessed and the characteristics of the concrete applications. In most cases they are complementary and can be used in combination in order to yield optimal recognition results". It is hard to disagree with this statement and hence the work described in this thesis is not meant to introduce the silver bullet to the problem of activity recognition, but rather to present a sound solution, which improves the existing low-level WSNM by tackling all of the relevant challenges for WSNM in the context of activity monitoring.

The above taxonomy is based on the means of perceiving the physical environment by the systems i.e. *visual-based* and *sensor-based* AR systems. Another classification

is based on the recognition algorithms [13]: *machine learning techniques* and *logical modeling & reasoning*. Both taxonomies are valid and do not contradict each other, as one is based on the type of input and the other on the type of applied algorithms.

1.8 Declarative Programming and Rules

In procedural programming the programmer sets the sequence of computational procedures to be carried out. The computer is presented with a list or set of instructions telling it exactly what to do, how to do it, and in what order. Even though this style of programming is suitable for many problems i.e. for those in which inputs are well defined and a set of computational steps is easy to determine (e.g. mathematical computations), it does not work well for some classes of problems. Often it is desirable to express the logic of a computation without describing its control flow – what the computer should do rather than how to do it. This approach to programming is called declarative programming. Due to the fact that declarative programs define only the logic of the solution, they can be much shorter and easier to understand than procedural programs. "Declarative programming is often the natural way to tackle problems involving control, diagnosis, prediction, classification, pattern recognition, or situational awareness – in short, many problems without clear algorithmic solutions"[22]. Declarative programs must be executed by a runtime system which understands how to use the declarative information to solve problems and which decides on the control flow. In a rule-based systems, the programmer only writes the individual rules and the rule engine program determines which rules to fire and with what effect.

Declarative programming, and rule-based approach in particular, is well suited to the problem of activity monitoring in WSNs. With this approach, one only has to specify the logic/conditions for some activities to occur and does not need to worry about writing step-by-step procedures for a computer program to follow. This makes the rule-based approach to the problem of AM short and simple in terms of volume of the

code and its readability. Moreover, it is in line with the definition of WSN middleware which is expected to ease the development, maintenance, deployment, and execution of sensing-based applications. Since WSNs are highly dynamic it is more appropriate to write the 'what' rather than precisely 'how' to accomplish the recognition of activities of interest. The network's configuration is likely to change as sensors come in and out of its range, run out of battery, etc. and hence implementing procedures which take into account all possible scenarios is very difficult.

In mathematics and computer science a *rule* is simply a "prescribed method or procedure for solving a mathematical problem, or one constituting part of a computer program, usually expressed in an appropriate formalism"[12]. In other words, a rule in its simplest form can be thought of as an if-then statement i.e. IF some condition(s) is met THEN some action(s) takes place. The logic for recognising various activities can be captured in the form of rules, which are easy to understand even by somebody with no programming background. Therefore, domain experts can contribute in the process of rules' writing i.e. their knowledge can be stored as a set of rules.

1.9 Rule-Based Systems

A rule-based system (RBS) is made up of three components, shown in Fig.1.2. The *knowledge base* (KB), also called the rule-base is where the knowledge is stored. It consists of rules and static facts. The *working memory* (WM), sometimes called the workspace, holds the temporary data. The *inference engine* (process) has a function of applying rules to working memory [22]. It interprets, analyses and processes rules. Knowledge is stored in the KB as rules, which are of the form: IF SOME CONDITION (s)THEN SOME ACTION(s). "The IF part of a rule written in this form is often called its left-hand side (often abbreviated LHS), predicate, or premises; and the THEN part is the right hand side (RHS), actions, or conclusions." [22]. Upon satisfying the condi-

[12]http://www.thefreedictionary.com/rule

tion(s) the action(s) usually makes some modifications to the KB i.e. asserts, deletes or modifies existing facts. RBSs belong to the paradigm of declarative programming, which is usually concerned with expressing the logic of the program, rather than its control flow.

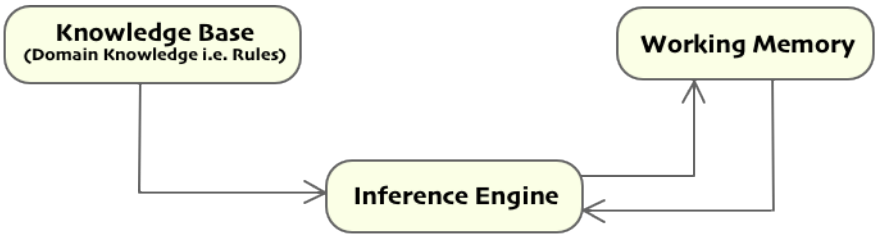


Figure 1.2: Rule-Based System

1.10 Semantic Web

The Semantic Web "is the extension of the World Wide Web that enables people to share content beyond the boundaries of applications and websites. It has been described in rather different ways: as a utopic vision, as a web of data, or merely as a natural paradigm shift in our daily use of the Web"[13]. This collaborative movement led by the World Wide Web Consortium (W3C), promotes common data formats through which data can be effectively shared and reused on the web. It aims at structuring documents on the web so that they are easily processable by machine and humans. Sir Tim Berners-Lee (director of W3C) envisioned the Semantic Web as a 'web of data' in which computers are capable of analysing the data without human assistance, where machines talk to machines in order to handle the day-to-day mechanisms of our daily lives. A lot of effort has already been made for this vision to materialise, however a lot of work still needs to be done.

The Semantic Web community considers rules to be an important part of their initiative as rules "constitute the next layer over the ontology languages of the Semantic Web

[13]http://semanticweb.org/wiki/Main_Page

and, in contrast to description logic, allow arbitrary interaction of variables in the body of the rules" [5]. The need of having rules in Semantic Web has been highlighted by many other authors, including [47],[37]. The members of the rule markup language (RuleML) initiative, which is an "international non-profit organisation covering all aspects of Web rules and their interoperation, with a Structure and Technical Groups that center on RuleML specification, tool, and application development" [14], recognised the need and importance of incorporating rules into the Semantic Web. Therefore, they specified the RuleML standard, which is "a unifying family of XML-serialized rule languages spanning across all industrially relevant kinds of Web rules"[14]. The RuleML community, through participation in other projects, contributed to the development of the Semantic Web Rule Language (SWRL) [27] and Semantic Web Services Language (SWSL[15]). Therefore, not only is the standardisation of rules' format essential, but equally important are the semantics of rules' inputs and outputs.

Nowadays, sensors which often reside at the edge of the Internet, contribute a vast amount of data to the web. In order for semantic rules to work with events captured via various sensors, these events need to carry semantics. Otherwise, machines cannot possibly be programmed to process/analyse such an unstructured data correctly. Therefore, semantic rules need semantic events to form a truly automated process of knowledge inference.

1.11 Scope and Adopted Approach

To address the WSNM challenges listed earlier in Section 1.5, in the context of activity monitoring, this thesis proposes a hybrid solution in which a rule-based system is implemented on the top of an existing low-level WSNM. The proposed layered architecture provides many benefits, which improve the low-level WSNM by overcoming its limitations. The aim of this research work is to design an efficient and effective

[14]http://ruleml.org/
[15]http://www.w3.org/Submission/SWSF-SWSL/

platform, capable of inferring semantic events of different complexity in the area of activity monitoring, which can aid decision-making. The proposed architecture is discussed in detail in Chapter 3. Much of this research was carried out in a practical context: healthcare rehabilitation monitoring (i.e. stroke patients), and real rehabilitation examples are presented throughout this thesis.

The aim of this research is to explore the usefulness of using the rule-based system in conjunction with WSNM in the area of activity monitoring. Camera sensors were not used in this research, due to the privacy considerations, and hence this work does not contribute to the field of visual computing. However, the openness of the approach does not reject the use of such sensors. In terms of the recognition algorithm taxonomy, it lies within the strand of *logical modeling & reasoning*.

The two pilot studies described in Chapters 5 & 6 were implemented to test the hypothesis that rules help to address the ten computer science challenges for WSNM [45] and ease the development, maintenance, execution and extensibility of sensing-based applications.

1.12 Research Questions

This thesis aims to answer the following research questions:

1. How effective is a rule-based approach to SNM in the context of activity monitoring:

   a) How well are the ten identified challenges for SNM met by this approach?

   b) What are the advantages & limitations of the proposed solution?

2. How does a semantic event model aid the process of sensor-data fusion/information inference?

3. How adaptable is a rule-based SNM platform in terms of changing user requirements, and bringing new sensors into the system?

4. Can rule-based systems effectively help with validation and verification of a sensor network middleware?

These questions are investigated in detail in this thesis. The following methodologies were used in conducting this research:

Case Study Research The usefulness of the proposed approach to the area of activity monitoring is assessed by considering the two case-studies that were carried out as a part of this work: the regional stroke unit and the home monitoring projects. Both these projects lie within the activity monitoring area and are specialised to the problem of rehabilitation of stroke patients.

Qualitative Analysis In both case studies the data collection protocol was carefully designed and appropriate equipment selected in order to provide an unobtrusive environment. Healthcare professionals were interviewed and a good understanding of both environments (hospital and home) was obtained. The data was collected for many participants over long periods of time to obtain unbiased results. The data was analysed and validated against recordings of manual observations. Even though high-level concepts existed prior to data analysis, it was the data's analysis which generated rules. Moreover, the obtained results have been analysed for conformance with reality.

Measurement, Validity & Reliability The observations captured by the automated system were stored and analysed in the context of validity and reliability. To judge on system's reliability, one needs to replicate an experiment and get comparable results i.e. seek for consistency. The validation of the system – judgment on how strong the outcome was, or in other words, whether the system really measured what it was designed to measure – was realised by internal validation, construct validation and external validation against hand-recorded observations.

1.13 Thesis Contribution

The main contributions of this thesis are as follows:

- The proposed layered event model (EM) is an effective approach to activity monitoring. The EM provides a solution to dealing with sensors' heterogeneity, helping to separate hardware related concepts from the domain concepts. Introducing new types of sensors only requires the lowest-level model to be extended and adopting to new domains or changing requirements requires a formulation/modification of domain's knowledge only at the upper layer i.e. domain-specific model. Moreover, the EM supports explanation of reasoning which adds transparency.

- A rule-based approach is effective in generating high-level events (HLE)s from low-level events (LLE)s. Already computed HLEs take part in the inference process of events of even greater complexity. Such an approach is efficient and saves unnecessary computations. Moreover, it fully addresses six out of ten identified challenges for wireless sensor network middleware and partially addresses another three (some of the responsibility in satisfying these also lies on the low-level sensor network middleware's side). This solution to activity monitoring was tested in two rehabilitation monitoring case studies and was tested to be valid and reliable.

- The shown combination of the layered event model and rules carries further benefits in terms of validation and verification (V&V) of the system. The verification of the system is performed by static cross-checking of rules, ontologies and sensor event data. As we show with the aid of examples from the implemented system, the proposed V&V procedures produce valuable information which increases the maintainability and transparency of the sensor network middleware. This solution allows for quick adaptation to rapidly changing requirements of sensor-based applications.

1.14 Overview and Structure

The first part of this thesis is an introduction to the problem of WSNs and Activity Monitoring. It sketches the scope of the thesis, provides the reader with essential background knowledge and the literature survey of the existing work.

- Chapter 2 expands on the challenges and problems with existing WSNM. It introduces existing standards for enhancing descriptions and meaning to sensor data, which aims to materialises the vision of semantic sensor web. It familiarises the reader with a way of classifying the middleware based on their 'closeness' to either sensors or applications. It describes three examples of low-level SNM and three cases of high-level SNM. The chapter concludes with a literature review in the area of rule-based WSNM, event detection and activity recognition for sensor-based systems.

The second part of the thesis focuses on the proposed system architecture, the modeling of the system and its implementation.

- Chapter 3 introduces the proposed layered architecture of the middleware that combines low-level SNM and an RBS. It discusses various technologies used and evaluates the proposed architecture against existing approaches.
- Chapter 4 discusses the motivation, modeling choices made and models that were created to represent various concepts within the platform. It also gives the reader some insight into the semantic encoding of the events inferred by the system and some example pseudocode of rules that the system implements.

The final part of this thesis introduces the two case studies that validate the proposed platform. The environment and the requirements for each of these case studies are described together with the selected approach and the results.

- Chapter 5 provides the reader with an insight into the motivation, requirements and approaches to the two pilot studies. The reliability and validity (further divided into script-based validation and observation-based validation) studies are described in detail together with their results. This chapter concludes with an insight into the home monitoring pilot study.

- Chapter 6 explains how the rule- and model-based approach aids validation and verification of implemented platform. The validation and verification procedures are introduced in detail and results of these strategies applied to the implemented system are shown. The chapter concludes with the summary of benefits that implemented validation & verification strategies provide.

- Chapter 7 presents a summary of the contributions made by this research. Future research directions and answers to research questions are also presented which conclude the thesis.

Chapter 2

Background & Literature Review

This chapter introduces existing approaches for enhancing descriptions and meaning of sensor data, which aim to realise the vision of a semantic sensor web. It then expands on the challenges and problems with existing wireless sensor network middleware and introduces a way of classifying the middleware based on its 'closeness' to either sensors or applications. Then examples of low-level sensor network middleware and high-level sensor network middleware are described and critically summarised. Finally, the chapter provides the reader with a review of rule-based wireless sensor network middleware, event detection and activity recognition for sensor-based systems.

2.1 Requirements for WSN Middleware for Activity Monitoring

Molla et al. identified ten middleware challenges for WSN. Their generic explanations can be found in [45]. However, what is the meaning of these challenges in the context of Activity Monitoring? What are the qualities that the WSNM has to meet to be considered a complete solution?

Abstraction Support – WSNs for activity monitoring (AM) are not necessary made of just one Real Time Location System (RTLS) but instead are often made up of a number of heterogeneous sensors. RTLSs are often limited in their capabilities;

therefore in order to meet the desired requirements of the sensing application, other sensors supplement these systems, effectively 'filling the gap'. On the other hand, a WSN in this domain may not contain any RTLS' but rather be made up of a collection of diverse sensors. Whatever sensors and/or sensor bundles are used the problem remains the same – an abstraction is needed to hide the different underlying hardware in order to provide a homogeneous view of the network.

Data Fusion – Regardless of the domain, this requirement is crucial for any WSN. Ability to merge sensor readings, that are usually fairly low-level, to form meaningful and understandable information is very important. Moreover, WSNM has to able to communicate this fused data to the interested parties.

Resource Constraints – Some environments, especially in the domain of AM, restrict the use of certain sensors e.g. cameras. Middleware for WSN has to be flexible enough to handle cases where alternative means of sensing have to be used. These alternatives may be limited in the quality of information they provide; however this can limit middleware's functionality but should not stop it from functioning.

Dynamic Topology – The configuration of WSNs is highly dynamic due to nodes failures, their mobility and communication limitations. Even in a well-equipped environment and extensively tested setting, the tracked objects (assuming they were sensors) come in and out of the system's range. Moreover, due to the fact that AM systems often execute unsupervised, WSNM has to account for the problems related to sensors' mobility.

Application Knowledge – WSNM should not be generalised, as it is aimed to support a wide variety of applications. Instead, it should allow for the integration of the application knowledge into the services provided – as a way of customising the middleware. 'One size does not fit all' hence some tailoring is desirable in order to achieve the desired functionality.

Programming Paradigm – Programming for sensor networks is very different from traditional programming. The constrained resources, unstable structure of the network and difficulties in harvesting and analysing sensor data requires different programming paradigms for the middleware.

Adaptability – Middleware for WSN in the domain of AM is no different from middleware in other domains in terms of adaptability. Requirements of various applications are vastly different, even in the area of AM, and hence WSNM must be able to adapt to the changing requirements, and components of a sensor network.

Scalability – Addressing this challenge is desirable for any system. Ability to deal with increasing number of sensors, increasing number of sensing tasks and increasing number of detectable objects is crucial for WSNM to be applicable to any size of problems.

Security – The activity monitoring involves the handling of sensitive data. This data often enables one to identify individuals, their activities and other sensitive information. It is the middleware's responsibility to safeguard the data, so that it is not easily interceptable by other parties.

QoS Support – Another challenge common to any computer system. Because WSNM often need to handle real-time scenarios they need to support a handful of QoS features. Response time, availability, node's failures, explanation of reasoning and other desirable features need to be handled appropriately in order to provide reliable services.

2.2 Survey of Sensor Network Middleware Approaches

2.2.1 Semantic Sensor Web

Tim Berners-Lee envisioned the Semantic Web as an extension of the World Wide Web, which focuses on formally defining the meaning (semantics) of information on the Web. The presence of well-described data on the Web enables machines to automatically process semantic data. Technologies of the Semantic Web include the Resource Description Framework (RDF) – data representation model; and the semantic metadata annotations, which has two principal elements: RDF Schema and Web Ontology Language (OWL). Sensors are connected 'at the edge' of the Internet and contribute large amounts of data to the Web, and for the Semantic Web vision to materialise sensor data needs to comply with these basic principles.

2.2.1.1 Open Geospatial Consortium: Sensor Web Enablement

The Open Geospatial Consortium (OGC)[1] defines the sensor web as a "web of accessible sensor networks and archived sensor data that can be discovered and accessed using standard protocols and application programing interfaces (APIs)" [11]. The Sensor Web Enablement (SWE) initiative of the OGC aims at defining and documenting a framework of standards for interacting with sensors and sensor systems connected to the Web. The vision of the SWE is to make sensor data as standardised and accessible as the HTML language and HTTP protocol standards enabled the exchange of information on the Internet. As a result, this group has established and prototyped the following set of standards: Observation & Measurements Schema (O&M), Observations and Measurements XML (OMXML), Sensor Model Language (SensorML), Sensor Observations Service (SOS), Sensor Planning Service (SPS), SWE Common Data Model, SWE Service Common, PUCK Protocol Standard, Sensor Alert Service (SAS) and

[1] http://www.opengeospatial.org/

Web Notification Services (WNS). Due to the fact that location information is crucial for accessing sensor data, the SWE standards are harmonised with OGC standards for geospatial processing. The leading technology in SWE standards is XML and XML schemas, which are used to "publish formal descriptions of the sensor's capabilities, location and interfaces" [11]. However, XML is concerned with serialisation, so that the encoded information can be passed between machines and parsed by them. In XML world what matters is the standardised serialisation format, where the principles of the Semantic Web talk about machine-readable representation of data model.

2.2.1.2 Semantic Sensor Web Ontology

Semantic Sensor Web (SSW) [60] is an approach to the problem of providing semantic descriptions and meaning to sensor data, combining the efforts of the OGC and Semantic Web Activity of the World Wide Web Consortium (W3C)[2]. The OGC SWE languages provide annotations for simple spatial and temporal concepts however, more abstract concepts are desirable e.g. spatial region, temporal interval. The SSW framework provides enhanced meaning for sensor observations to enable situation awareness – it does so by "adding semantic annotations to existing standard sensor languages of the SWE" [60]. This approach overcomes the limitations of the SWE's XML approach to sensor data by bridging it with the Semantic Web technologies i.e. RDF/OWL.

Authors of SSW chose RDFa (a W3C proposed standard) for annotating sensor data. The RDFa (Resource Description Framework in Attributes) is a technique that enables adding structured data to HTML, XHTML and XML documents i.e. "it provides a set of markup attributes to augment the visual information on the Web with machine-readable hints"[3]. From RDFa, RDF triples can be extracted using a simple mapping. This semantic annotation of XML-based languages of SWE, supports the Semantic Web vision of machine-processable data. The authors' vision is that domain-specific ontologies for the SSW should be kept in a registry to support knowledge reuse.

[2] http://www.w3.org/2001/sw/

[3] http://www.w3.org/TR/xhtml-rdfa-primer/

2.2.1.3 Semantic Sensor Network Ontology (W3C)

The W3C Semantic Sensor Network (SSN) Incubator group was initiated by CSIRO (ITC Centre, Australia), Wright State University (home institution of authors of SSW) and OGC. Their task was to develop an OWL ontology for sensors and to further investigate the problem of semantic compatibility [14]. As a result of this initiative, the SSN ontology was developed by "firstly developing core concepts and relations (sensors, features and properties, observations and systems). Then, measuring capabilities, operating and survival restrictions, and deployments were added in turn. Finally, the alignment to DOLCE-UltraLite[4] (DUL) and the realisation of the core Stimulus-Sensor-Observation ontology design pattern were added" [14]. Authors of this work claim that the SSN ontology covers large parts of SensorML and O&M standards and they list a handful of projects, which make use of this ontology. The authors of this work do not specify how the SSN ontology should link to SWE framework however, they mention it as a key in the Semantic Sensor Web and Linked Sensor Data work at 52° North[5] – it is used in the implementation of a transparent and RESTful proxy for OGC's SOS.

2.2.2 Taxonomy of Wireless Sensor Network Middleware

In [46] Georgios Mylonas asks: "does the software run on top of the WSN or does it run completely or partly on a sensor node level?" This question is answered in [58] "the scope of middleware for WSN is not restricted to the sensor network alone, but also covers devices and networks connected to the WSN". In this sense, the WSNM covers all the aspects from providing programmer with an API to program applications, to handling connection and disconnection of a single sensor to the network. "Essentially, someone could argue that since WSNs are application-specific systems all software interfacing such networks with more traditional systems can be classified as middleware,

[4] http://www.loa.istc.cnr.it/ontologies/DUL.owl
[5] http://52north.org/

since it is a means of developing sensing applications." The result of such an argument can easily be noticed in the literature. Many researchers call their software a middleware even if it does not meet all of the ten WSNM challenges [45]. Due to this fact WSNM have been developed at different levels, resulting in a vast number of often incomplete solutions. The need for classification of such middleware arises, as the term 'WSNM' now covers a very wide range of software working at different levels.

There is a number of different classifications for sensor middleware, e.g. [26] classifies middleware based on their origin and types of approaches used:

- database-inspired solutions,
- tuple space solutions,
- event-based solutions,
- service discovery-based approaches.

However, this classification tells us only about concepts the middleware is based on, and does not give a full picture of how 'complete' the solution is or how much programming effort is required to actually build an application that uses this middleware. The purpose of a wireless sensor network middleware is to provide a bridge between physical sensors and applications, allowing for fast application development by providing APIs. Therefore, a more natural way to classify existing sensor middleware is perhaps to define its closeness to entities at both ends of the bridge. This classification can be based on the abstractions used by different approaches, as according to [45], *abstraction support* is one of the key challenges in the development of WSNM. Therefore, a middleware that is close to physical sensors and is focused around basic management and maintenance duties, without a sufficient support for application development, can be classified as a *low-level WSNM*. On the other hand, if it is biased towards application development and is rather primitive at the sensor level, or just assumes that sensor data is available, it can be classified as a *high-level WSNM*. If there was a solution that can satisfy both these requirements, then it would be classified as a *complete WSNM*.

Based on how a SNM meets the ten identified computer science challenges for wireless sensor network middleware [45] and referring back to the definition of SNM defined in Chapter 1, the existing SNMs can be classified into one of the following categories: low-level SNM and high-level SNM.

2.2.3 Low-level Wireless Sensor Network Middleware

Many of the popular sensor network middleware, such as Global Sensor Networks (GSN) [1], International Technology Alliance (ITA) Sensor Fabric [71], or Sensor Web Enablement-compatibles (SWE) [11] are rather low-level. They simply provide sensor data (using different models and abstractions) and do not make it very easy for the programmer to program with them. These approaches are not complete systems that meet all the ten challenges to a satisfactory level. *Application knowledge, adaptability* and *abstraction support* are not very well addressed by these SNM.

2.2.3.1 Global Sensor Networks

Global Sensor Networks uses an abstraction called 'virtual sensor', which provides logical view on sensor networks. It abstracts from the implementation details to access sensor data and models sensor data as temporal streams of relational data. It can provide data derived from post-processing and sensor data from different sources. Virtual sensors are bound to physical nodes in the network by deploying them to GSN container that hosts virtual sensors [1]. This is a very powerful abstraction as users can work with sensor data from heterogeneous sensor (virtual or real) sources and any data derived from raw sensor data. Simply, a virtual sensor has zero or more inputs from physical sensors and produces one output (which of course can be complex and can consist of multiple fields). Due to the fact that an output of the virtual sensor is encoded in XML and the fact that virtual sensor definition is also in XML format, it is relatively easy to use any programming language to program applications. However,

application development with GSN is time consuming and there are many low-level details the developer has to worry about. The API provided by GSN is perhaps too simple and atomic to allow for fast application development. Therefore, this middleware is classified as low-level.

2.2.3.2 Sensor Web Enablement: SensorML

The Open Geospatial Consortium (OGC) has proposed a set of specifications for sensor middleware, which slowly becomes defacto a standard. Included in this specification is Sensor Model Language (SensorML). Based on the fact that "the measurement of phenomena that results in an observation consists of a series of processes (also called procedures), beginning with the processes of sampling and detecting and followed perhaps by processes of data manipulation", the key abstraction used is called a 'process'. In their White Paper [11], OGC describes it as follows: "within SensorML, everything including detectors, actuators, filters, and operators are defined as process models. A Process Model defines the inputs, outputs, parameters, and method for that process, as well as a collection of metadata useful for discovery and human assistance. SensorML treats sensor systems and system's components (e.g. sensors, actuators, platforms, etc.) as processes". Due to the fact that in SensorML all processes have input, output, parameters and methods that can be invoked on them, each component can be included as a part of one or more process chains, similarly to virtual sensors in GSN (one process'/virtual sensor's output can be an input for another process/virtual sensor). SensorML defines models and XML Schemas for describing any process, similarly to GSN defining their own models and XML Schemas for describing a virtual sensor. The abstractions and approaches used in the two middleware are very similar, which classifies SWE:SensorML as 'low-level' SNM. It is relatively time consuming to program with SensorML on its own, however if other standard frameworks proposed by SWE are adopted and used with SensorML, it may result in a complete sensor network middleware.

2.2.3.3 International Technology Alliance Sensor Fabric

The ITA Sensor Fabric uses broker-based publish/subscribe messaging pattern and claims [70] to address issues of:

- sensor identification and discovery,
- sensor access and control,
- sensor data consumability.

Most of the benefits of Sensor Fabric come from its messaging model, which provides a one-to-many distribution mechanism that utilises a central hub or broker through which all messages pass [71]. Therefore, the sensor node publishes a single sensor reading to the message broker only once, and then it is the broker's responsibility to pass the message on to the interested parties. This saves sensors' energy and sensor networks' bandwidth and entrusts the responsibility of delivering sensor data to a more powerful machine on the network. Both, publishers and subscribers, only have to connect to the broker and they have no knowledge of the source or destination of messages, unless this data is specifically included in the message content. Therefore, the complexity and heterogeneity of the sensor network is hidden via use of messages. This abstraction makes this middleware to be more on the physical sensor's side and does not help with fast application development. The programmer has to be concerned too much about low-level configurations, such as making sure that in this data-centric approach, the location of the sensor and other sensor specific data is coded in the message itself and there is no way for direct querying of sensors. Tasking the sensor also has to pass through the broker, so that in a sensor rich environment, brokers are overloaded with huge message traffic. This abstraction simplifies the whole problem, but also limits network's capabilities and results in an additional effort of implementing Fabric Extensions Points, plug-ins or Fablets, which allow for: adding functionality to the Fabric, executing processing tasks directly on messages, and introducing data into

Fabric from sensors that are not Fabric aware. Similarly to GSN, ITA Sensor Fabric can be classified as low-level sensor network middleware.

2.2.4 High-level Wireless Sensor Network Middleware

The high-level WSNM are often bounded to a particular set of supported sensors i.e. are closed at the 'bottom'. This is a serious limitation for WSNM which has to perform in a quickly evolving domain of sensors. The abstractions they use and leading concepts behind them, are often very neat and interesting, however these approaches are not flexible enough to meet the ten identified challenges to a satisfactory level.

2.2.4.1 Semantic Streams

The Semantic Streams (SS) approach is based on two fundamental elements: 'event streams' and 'inference units'. Whitehouse in [69] describes event streams as something that "represents a flow of asynchronous events, each of which represents a world event such as an object, person or car detection and has properties such as the time or location it was detected, its speed, direction, and/or identity". Inference units operate on event streams by inferring semantic information about the world from incoming events and either generate new event streams or add information to existing events as new properties. Therefore the key concepts in SS are: events, semantic (or event) streams and semantic services (inference units). In this approach, *sensor selection and service composition* is automatic, via use of a variant of backward chaining algorithm [69]. Semantic streams become facts and semantic services are treated as inference rules that can derive output semantic streams from input streams. All facts are stored in a knowledge base (KB), which initially contains all physical sensors (in SS, a sensor is simply a semantic service with no preconditions/input streams), and they are used to prove the query by constituting the inference graph [40].

In the SS approach the user only has to specify the end results rather than how these results are computed. Sensor data is delivered via use of Microsoft Research Networked Embedded Sensing Toolkit (MSR Sense) – a collection of software tools that allow users to collect, process, archive and visualise data from a sensor network[6]. Therefore Semantic Streams on its own is rather high-level. Whitehouse [68] summarises it as follows: "Semantic Streams is a step toward providing a high level abstraction for end users to interact with sensor networks and enabling transparent in-network processing and component reuse. In many domains, the users only need to specify the end goal in terms of what semantic information to collect, and the service composition framework automatically specifies and glues the necessary components to achieve that goal". Summarising, Semantic Streams is a good example of high-level sensor middleware, which is 'closed' at the low-level as it only considers data delivered by the MSR Sense platform.

2.2.4.2 Sensor Assignment to Missions

In the Sensor Assignment to Mission (SAM) approach, the key concepts are: tasks, operations and missions. These concepts are best explained in [10]. A mission (e.g. a rescue mission) is made up of operations (e.g. rescue people) and operation is simply a set of user's tasks (e.g. search for victims). The main goal of SAM is to allocate a collection of sensors/sensor platforms to some mission tasks in order to satisfy all the requirements of these tasks [52]. Similarly to Semantic Streams, SAM uses a low-level sensor middleware (in this case ITA Sensor Fabric), which provides an interface to interact with sensors (in SAM's case simulation feature of the Fabric is used to simulate sensors). Therefore SAM is another example of high-level middleware, as it assumes sensor data is available via use of some other sensor middleware. Again, by choosing SAM as the high-level sensor middleware, the choice of low-level middleware is restricted to Sensor Fabric.

[6] http://research.microsoft.com/en-us/projects/msrsense/default.aspx

2.2.4.3 Sensor Web Enablement: Sensor Planning Service

Sensor Web Enablement specifies another standard for interfacing with sensors called Sensor Planning Service (SPS). Boots et al. in [11] describes SPS as a "standard which specifies open interfaces for requesting information describing the capabilities of a SPS, for determining the feasibility of an intended sensor planning request, for submitting such a request, for inquiring about the status of such a request, and for updating or cancelling such a request". SPS assists in collection feasibility plans and processes collection requests for a sensor or sensor constellation. It achieves it by using Service Registry to discover sensors and platforms and then passes an observation request to the Sensor Observation Service (SOS) – these are another standards proposed by the SWE group. SPS is mainly focused around plans and planning requests, therefore can be classified as a high-level sensor middleware since it performs a number of actions in order to dynamically task sensors and provide information collected from sensors. SPS, due to the service-oriented architecture (SOA) adopted by SWE, does not have to worry about low-level functions such as establishing a connection to the sensor or storage of its data, as this is all provided by other services. It provides an interface to interact with sensor networks, thus it is tied more to the application than to the physical sensors.

2.2.5 Summary of Wireless Sensor Network Middleware

There are many desirable criteria that a good SNM ought to meet. The first are those relating to the definition of SNM – defined in Section 1.4 – stating that middleware eases development, maintenance, deployment, and execution of sensing-based applications. Therefore, the user should not worry about the low-level protocols – such as is the case with the low-level SNM – but only needs to focus on querying the middleware for the right data, which ought to carry some semantic meaning. Sensors often reside 'at the edge' of the Internet, contributing data to the Web, and hence should follow the

practices of the Semantic Web. In this way, their data becomes machine-processable without any room for ambiguities.

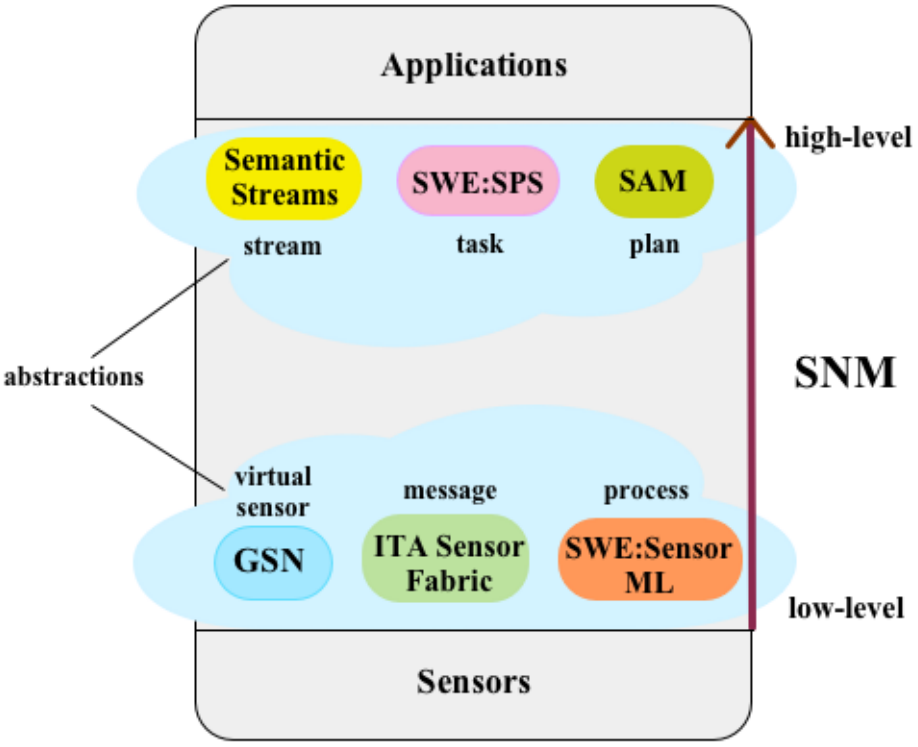


Figure 2.1: Classification of SNM and their abstractions

Secondly, it is desirable for a SNM to be universal i.e. not built for purpose. Initiatives such as Semantic Streams [69] are a very good example of exactly opposite practice, where the system is not easily configurable and only works with a pre-determined set of sensors. The ideal system should not be bound to a specific low-level SNM (closed at the bottom), so that a wide range of sensors can contribute to it. In addition, it should be open (at the top) to changing requirements and new domains, providing a flexible solution capable of adapting to new challenges. Sensor systems are often dynamic – due to their underlying technology i.e. sensor market is fast evolving – and hence

should allow for quick reconfiguration.

There are many SNM, which have been classified into two the categories, namely low-level and high-level SNM. Their abstractions and closeness to applications and sensors are mapped in Fig.2.1. Both these classes of middleware have their shortcomings and none of them can meet the aforementioned criteria to the satisfactory level.

2.2.6 Rule-Based Wireless Sensor Network Middleware

The idea of incorporating declarative programming such as rule-based systems, into WSNM is not new. It has been developed and tested by many researchers in many different ways. Some solutions work at the node level and some are applied centrally e.g. Semantic Streams [69]. FACTS [66] is a good example of the first approach, where "facts, rules and functions are local to each node of the sensor network and each node runs its own rule engine". In this approach, *rules* express algorithms and carry action semantics. The main abstraction is *events*, which trigger actions to allow programming of the system in a resource-aware manner – this approach helps to manage power consumption in nodes. FACTS uses a forward chaining inference engine in order to provide event-like semantics. Authors of this work argue that "although an *event-based approach* is the most obvious solution, it also provides the most natural way to address software development in wireless sensor networks" [66]. Moreover, they state that a *modular design* is a key design aspect to obtain a highly flexible framework. Similarly, Generic Role Assignment (GRA) project [57] encodes node behaviour using rules. D-FLER [42] proposes a similar solution – "a lightweight, distributed fuzzy logic engine operating with simple IF-THEN rules applied over both the individual sensor readings and the neighbourhood observations". Authors have tested their solution through simulation and through implementation on real sensor nodes. Similarly, Cooperative Artefacts [63] is another example of a rule-based solution, which works at the node-level. This solution is based on a Prolog interpreter and hence its functionality is limited, since it runs on sensor nodes limited in resources (memory, CPU, etc.).

REED (Rule Execution and Event Distribution) middleware [19] [18] [72] is another solution which enables applications and users to program sensor nodes' behaviour. The main contribution of [18] is the ability to program a rule-based WSN at run time. The implemented PROSEN WSN, based on REED middleware, is: event triggered and light-weight, programmable at run time, capable of event subscription and notification, and supports energy-conservation. It consists of a Policy Server, which interacts with users and operators to obtain the goals for the system. The processing node translates these goals into policies, which are in turn translated into rules. These rules, and events, are distributed amongst other processing nodes. In [72], authors propose a method of developing WSN applications that uses finite state machine as a bridge between application logics and the rules running on the REED. The emphasis in this work is on making application development easier – one of the fundamental principles of WSNM.

The aforementioned approaches push rule-based reasoning engine to the sensor nodes in order for the nodes to compute events locally, thus reducing the amount of data which is to be sent over (wireless) network i.e. reduce bandwidth; and in some cases to correct non error-free data reported by sensors, and/or to use local knowledge to assist in the process of event inference. Pushing application logic to the sensor nodes has advantages and disadvantages. On one hand, reducing the amount of outgoing data can preserve battery life. On the other hand, performing computationally intensive task on a resource-poor sensor node can drain its battery and prolong node's response time.

2.3 Event Detection in Wireless Sensor Network Middleware

Taylor and Penkala in [65] propose to employ semantic technologies for user programming of sensor devices. Their work shows "how an ontological concept description can be translated to an active query or command to a sensor device, coupling interpretation of a declarative ontology with device-specific wrapper code". User expresses

their query in an ontology-driven interface, which has been developed as a Protege[7] plug-in, backed up by a description logic reasoner (Pellet[8]). In this interface users can see available sensors (discovered automatically), their capabilities and form their own query. The query gets classified and executed to eventually reach sensors via use of the Ontology Transformer, which automatically translates it to a sensor-device specific form.

Taylor and Leidinger take this idea even further. In [39] and [64] they propose the use of ontologies to "specify and recognise complex events that arise as selections and correlations (including temporal correlations) of structured digital messages, typically streamed from multiple sensor networks". Ontologies (based on SSN ontology [14]) are used to define complex events of interest – via the user interface which again lists available sensors and their capabilities. These events are translated to selections and temporal combinations of streamed messages by the semantic event middleware. Event descriptions execute under the control of a commercial Complex Event Processor (CEP), which performs event detection (based on received streams and queries) and sends an alert if a specific event was recognised. An event description is made up of an alert and the definition of the event itself, called an observation. In order for an event to represent complex structures defined by the user, every complex observation is made up of multiple atomic observations. Moreover, the possibility of reusing streams of information, which are already used by previously specified complex events, is anticipated in this architecture. However, the approach is limited by the set of basic operators embedded in their CEP, and also does not support integration of analytical tools. In [39] these authors planned to integrate GSN [1] into their software – which in this thesis is considered a low-level SNM.

In [67] authors propose EPC SNM, which similarly to [39] and [64], proposes CEP technology to handle large volumes of events from distributed RFID and sensor readers. Their solution follows the relationship set way and hence is not very powerful

[7] http://protege.stanford.edu/
[8] http://clarkparsia.com/pellet/

in terms of complex rules specification. The authors define simple events and complex events which are combinations of the first. The event specification (consisting of events, conditions and actions) is interpreted by a parser, which "...transforms event specification into four processing steps: filtering, grouping, aggregation and complex event construction" [67]. The authors do not specify in what language rules are expressed, however they presented sensor events in XML format. They have adopted publish/subscribe approach to expose data to applications, however, there is no mention of semantically annotating this data.

Some authors [16] take a fairly low-level view onto the event detection problem. Rather than using a base station or other central processing, which increases communication overhead (drains sensors' batteries), the authors propose a distributed event detection system, which suits the WSN's characteristics. It focuses on correct event detection at the node level – aims to improve detection accuracy. Their solution "uses the distributed sampling of sensor nodes to optimize the accuracy of the event detection process" [16]. Sensors nodes locally recognise previously learned patterns and also exchange their data with other nodes that were triggered by the same event. This solution lies in the space of sensor nodes programming rather than WSNM.

The problem of event detection is approached in a similar way in [7]. Again, this happens at the sensor node level but the solution incorporates two stage evaluation of readings via use of feed forward neural network (FFNN) or Naive Bayes classifiers for which learning happens offline. At the first level, each node performs a classification task in order to decide upon occurrence of an event. In the second level the decisions of individual nodes are fused and a judgment is made whether to report an event or disregard it.

2.4 Activity Recognition in Wireless Sensor Network Middleware

Activity recognition via the use of sensors is a well-explored research area. Many researchers have conducted studies in order to extract activity patterns from various data sources. As already presented in Chapter 1, there are many means of classification for such systems. Because the work presented in this thesis does not lie within the strand of visual computing, nor aims to work at the level of body sensor networks, the state-of-the-art for the similar class of problems is reviewed. However, in the Chen and Khalil's [13] classification of AR approaches (*vision-based* and *sensor-based*), it is hard to fit our work into one subcategory of the sensor-based approaches. Work presented in this thesis does not assume that all sensors are worn by subjects (*wearable sensor-based* approach), nor that only surrounding objects are equipped with sensing devices (*object-based* AR). Both these classes of the sensor-based AR have serious downsides.

The biggest drawback of the wearable *sensor-based* solutions is the inconvenience of having to wear either multiple devices, or sensors that are big in size, or both. All this makes the subject feel uncomfortable and definitely violates the definition of ubiquitous computing being achievable via use of seamless technology. As various studies have shown [49] [38] [50] [8] [28] body worn sensors can be very effective for the purpose of activity recognition however, they are not always acceptable from the subject's perspective. Moreover, human's behavior is often influenced by the presence of other humans or by the consciousness of being watched – of which wearable sensors, if not seamlessly integrated, can be a strong remainder. Secondly, as pointed out in [13], "many activities in real world situations involve complex physical motions and complex interactions with the environment. Sensor observations from wearable sensors alone may not be able to differentiate activities involving simple physical movements, e.g. making tea and making coffee". Therefore, while wearable *sensor-based* solutions

achieve high activity recognition rates:

Bao and Intille – Using decision tree classifiers, achieved recognition rates of between 80% – 95% for 20 activities, with the overall recognition rate of 84.26% [8],

Parkka et al. – Conducted a study in which 7 activities (Lie, Row, ExtBike, Sit/Stand, Run, Nordic walk and Walk) were classified using three different approaches: custom decision tree classifier (total classification accuracy of 82%), automatically generated decision tree (86%) and artificial neural network (82%) [49],

they violate basic principles of ubiquitous computing: calmness (technology "which informs but doesn't demand our focus or attention"[9]) and comfort.

On the other hand, object-based AR, even though nowadays financially affordable, is very laborious in terms of setting up the environment. Labeling of objects of everyday life, such as: mugs, boxes with tea, coffee, milk, etc. imposes a huge overhead and workload to the system administrator, when one of the most desirable feature of AR sensor-based systems is their low maintenance factor and ability to work unsupervised. Moreover, if there are many subjects in the monitored environment, it becomes hard to infer who the subject of the action was. The authors of [29] use Dynamic Bayesian Network (DBN) framework for AR from interactions with objects. In their setup, a nurse performs a drip injection procedure and the purpose of the system is to prevent the cause of medical accidents and incidents. Again, this is an example of how uncalm the technology can be made by researchers. In their lab setup, the nurse had an RFID reader attached to her hand and all the objects she interacted with were tagged with RFID tags – some of them carried multiple tags to allow for precise detection of interaction. In [41] hybrid discriminative/generative approach with hidden Markov model (HMM) is used for *object-based* AR. Authors of this work however, did not attach sensors to various object, but prototyped a wristband type device consisting of: camera, microphone, accelerometer, illuminometer and digital compass. Moreover, they

[9] http://www.ubiq.com/hypertext/weiser/calmtech/calmtech.htm

claim that these sensors can be embedded into a wristwatch, which would solve the problem of users having to wear uncomfortable sensors. The basis for AR recognition is formed by the visual feature extraction. The biggest downside of this approach (and of other vision-based approaches) is the use of camera, which invades users' privacy and drains battery very quickly as images are continuously sent over the air to the host PC. Moreover, as with every machine-learning approach, this HMM approach to AR requires training data, which has to be "acquired in each user's environment because these sensor data are environment dependent" [41]. Moreover, the training data needs to be annotated, which is a tedious task. Therefore, these AR systems usually do not perform well straight out of the laboratory. Training data is hard to acquire and hence these solutions are not very scalable.

Most of the AR systems based on supervised *machine-learning* techniques have a major downside. They rely on the application of supervised learning algorithms that, in order to perform well, need to be trained with a sufficiently large amount of labeled data i.e. scalability and practicality issues. The COSAR system [55], tries to overcome these limitations through the integration of statistical and ontological reasoning. This technique relies on "modeling context data in ontologies and using derived semantic relationships expressing the feasibility of performing a given activity in a given context". This work was taken even further [56] and its authors have proven the superiority of their approach over purely statistical methods. They have implemented their system using COTS sensors and an Android-based smartphone, which apart from hosting the main activity recognition module, provided its own sensor readings (accelerometer data). The combination of statistical and symbolic reasoning, combined with the use of ontologies yielded impressive results, however there are still areas in which the proposed approach needs to be improved, e.g. consideration of data other than just location for ontological reasoning (duration, temporal relationships). As the next step authors consider the extension of their ontology and investigation of probabilistic frameworks (such as HMM) to replace their historical prediction procedure [56].

To overcome the limitations of many *logic-based* approaches which do not handle missing activities and noise; and *probabilistic-based* approaches which require training data and do not cope with conflicts well; [20] proposes "a rule-based recognition system for hierarchically-organised activities that returns only logically consistent scenarios". Their solution deals with noise, uncertainty, resolves conflicts and reports logically consistent scenarios. Moreover, their system takes parameters which define the reported level of abstraction and detail. At its lowest level, the system takes as inputs atomic activities (sensor data). Complex activities are inferred with the aid of hierarchical rules, which map temporal and spatial conditions between atomic activities. Uncertainty is achieved by listing optional sub-activities of inferred complex activities – the more of these optional conditions are met the higher the confidence factor. The detection of pairs of conflicting activities is performed on the list of all computed complex events. The resolution of the conflicts "is converted into a weighted partial maximum satisfiability problem such that any solution to it corresponds to a set of identified activities (scenario) that do not conflict" [20]. Weights in this solution correspond to the preferred level of detail and to atomic facts' confidence, temporal duration and number of sub-events. This system was tested and achieved 100% accuracy, precision and recall for noise levels up to 10%; as well as high values of these qualities for higher proportions of noise in the data. However, in this work there is no mention of the low-level sensor infrastructure – was any SNM used? It is also unclear whether produced activities carry any semantics i.e. adhere to some ontologies. The approach in [17] uses the Semantic Web Rule Language (SWRL) and ontologies (to model context and assist in the process of AR), applied to the example of smart classrooms. The biggest limitation of this work was the "inability to recognise more than one activity at a time, or the activities of more than one person" [17]. These are substantial limitations, as AR systems should be capable of monitoring multiple subjects performing many activities at the same time – need to be scalable.

Referring back to the classification based on the recognition algorithms [13]: *machine learning techniques* and *logical modeling & reasoning* need summarising. The *ma-*

chine learning techniques include "both supervised and unsupervised learning methods, which primarily use probabilistic and statistical reasoning." [13]. Amongst supervised learning techniques the following leading algorithms are applied: Hidden Markov Models (HMMs), dynamic and naive Bayes networks, decision trees, nearest neighbour and support vector machines (SVMs) [13]. These approaches are able to handle noise, uncertainty and incomplete sensor data but often require a large amount of labeled training and test data (supervised) or a computation of hand-crafted probabilistic parameters for every activity (unsupervised). Logical modeling & reasoning techniques have a downside of not dealing with missing data and noise [6]. Moreover, these techniques struggle to represent fuzziness and uncertainty. Probability-based AR solutions, such as HMM, suffer scalability issues, as they are dependent on the sequence of data produced by sensors. Hence, a large group of sensors producing a very large number of readings, affects the performance of the algorithm. Moreover, many systems (not only probability-based) are tied to the specific sensor setting – they lack interoperability. Changing sensor layer requires probability-based approaches to be trained again, and this is an important limitation, which in a fast evolving sensor domain (cheaper and better sensors replace older ones) is not acceptable. In general, supervised machine learning techniques require training of the system, so when new sensors are introduced to the setting, their data needs to be carefully labeled and the system trained. Unsupervised machine learning techniques try to construct recognition models from unlabelled data but they require manual assignment of probability to each activity and a definition of a stochastic model that can update these values based on new observations.

The general trend noticeable in most approaches to the problem of AR, is for researchers to either simulate sensor data or to test it with a specific type of sensor network middleware (or even just hard-wired sensors). Clearly, the AR layer needs to be logically separated from the sensor layer, which in turn need to be open to various SNM, so that changes at this layer do not affect the working of the recognition module. Simple but powerful abstractions are needed to logically separate the two layers and to ab-

stract away sensor's heterogeneity. Models need to be simple and generic to suit any type of sensors used. An ideal solution would require minimum programming effort to account for sensor replacements or extensions to the original setting e.g. by providing a wrapper to interact with sensors, and leaving the AR logic untouched. An ideal solution should also not limit the user in the use of sensors i.e. be only suited for either *wearable sensors-based* approach or *object-based* AR. The reality is that we are surrounded by sensors embedded into surfaces of everyday life, but also (especially with increasing popularity of smartphones) we tend to wear sensors close to our body. Vision-based approaches rely heavily on visual data. These solutions are applicable to many domains, such as investigating suspicious activities in public places. However, in some domains where the use of cameras is forbidden (violates privacy, etc.), they cannot be used e.g. in the context of assisted living or hospital monitoring.

2.5 Discussion & Conclusion

Section 2.2.5 of this chapter identified problems with low-level and high-level SNM. Both these classes of SNM are closed at either end and do not form complete solutions to the problem of AM. Rule-based WSNM (Section 2.2.6) usually work at the node-level and do not form solutions complex enough to aid application development sufficiently. Moreover, many of these efforts lack semantics, which is very important for interoperability and ability for machines to process sensor data efficiently. Approaches focused on event detection (Section 2.3) are often limited by the set of basic operators embedded in their CEP and are unable to capture complex logic – limited by the expressiveness of their event detection unit. Similarly to the rule-based approaches, some of these solutions work at the node level to discover events and often do not incorporate any semantics.

A good WSNM solution to the problem of AM should overcome these limitations. Firstly, it has to produce semantically rich sensor-derived data. Secondly, it needs to

be open at the top (to new requirements/domains) and at the bottom (to new sensors) – as opposed to ad-hoc solutions presented in this chapter. Moreover, it ought not to be limited to the number of activities it can recognise or to the number of people it can track (as in [17]). It should follow Mark Weiser's principles of calm technology, rather than be at the centre of user's attention. Furthermore, a sound solution should not be bound to either of the sub-categories of the *sensor-based* approach to AR (*wearable sensor-based* or *object-based*). Ideal SNM should also be scalable and not require laborious training, computing probabilities by hand, or re-training the system when new sensors are introduced. Finally, SNM should be field-tested to prove its usefulness and applicability to real problems – simulations do not always expose all limitations of the system.

par A WSNM funded on rules and semantic layered Event model can help to overcome these limitations. Firstly, it addresses the problem of semantics by exposing sensor data directly linked to its ontological definition. The layered Event model also ensures its loose coupling, which adds flexibility in terms of new sensors and new requirements (non ad-hoc solution). Only models and rules which are related to new sensors/requirements need to be added and the rest of the system remains intact. Moreover, rule-based systems are expressive enough to capture complex relations between facts. Rules are applicable to all data in the WM and do not limit the system to only track one person or recognise activities of only one entity at the time. Rule approach allows great degree of flexibility and is not limited to any of the sub-categories of *sensor-based* approach to AR – sensors can be worn by users and they can also be attached to various objects in the environment. This approach, as opposed to *machine learning techniques*, does not require laborious training or computation of hand-crafted probabilistic parameters. Designing and testing a WSNM funded on rules can be done in collaboration with domain experts. Rules are of simple format and can be easily understood even by somebody with no programming knowledge. Hence, they facilitate injection of human knowledge into the system. It is also anticipated that presence of rules and semantic models helps in validation and verification of such system by cross

checking its components. Maintainability of a WSNM is highly desirable, especially that sensor networks are non-static and neither are requirements of sensing applications. However, none of the solutions found in the literature that are mentioned in this thesis target this problem.

In this chapter we have reviewed the background work relevant to this thesis. In Section 2.2.1 the semantic sensor web work informs the development of a semantic event model in Chapters 3 and 4. Following the identification of low- and high-level wireless sensor network middleware in Section 2.2.3 and Section 2.2.4 we attempt to create a kind of middleware that bridges the gap, which we introduce in Chapter 3. Chapters 4 and 5 demonstrate the effectiveness of a rule-based approach – drawing on previous work in Section 2.2.6; and present a new approach to activity recognition (drawing on the work in Section 2.4) based on a layered Event model, which we introduce in Chapter 4.

Chapter 3

Proposed Architecture

This chapter introduces the proposed model, layered architecture and a platform, which combines low-level wireless sensor network middleware and rule-based system approaches. It discusses various design and implementation choices made, the platform's degrees of flexibility, and intended benefits of the proposed architecture.

3.1 Model

This model represents the structure of events, which "...are primarily linguistic or cognitive in nature. That is, the world does not really contain events. Rather, events are the way by which agents classify certain useful and relevant patterns of change" [3]. These patterns of change are characterised by a set of properties, such as: subjects and objects involved, time period in which they took place, location at which they occurred, and sub-events if they are not atomic. These common characteristics of an event make up the properties of the *Event model*, which is described in more detail in Section 4.2.1.1. Events which are atomic i.e. do not contain sub-events or consist of sub-events of the same type are called low-level events (LLE)s, and other (complex) events can be thought of as high-level events (HLE)s. *Abstraction support* is an important challenge for WSNM [45] and this *Event model* captures concepts relevant to the domain of activity monitoring.

3.2 System Architecture

The conceptual system architecture in Fig.3.1 was formulated to improve the existing low-level WSN middleware by addressing the basic objectives of such software i.e. to provide better support for "...the development, maintenance, deployment, and execution of sensing-based applications" [58]. The layered approach presented in this section separates applications, the Rule-Based Semantic Sensing Platform (RBS2P) and sensors. The RBS2P is divided onto three processing layers, each of which deals with a different set of sensor-data processing and aggregation tasks.

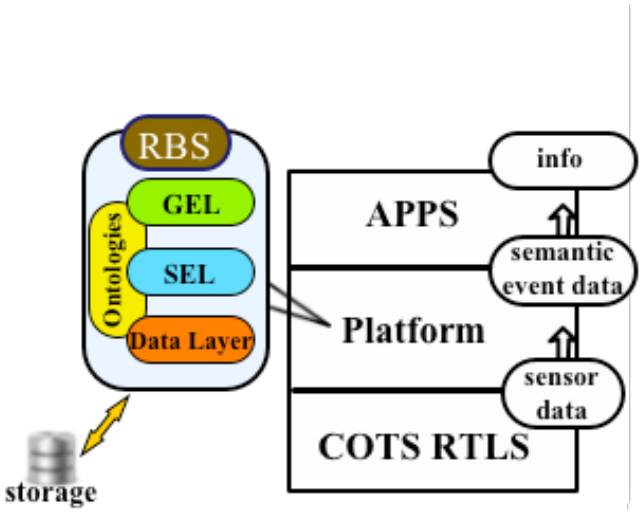


Figure 3.1: Architecture Diagram

3.2.1 Commercial Off-The-Shelf Real Time Location Systems

Commercial off-the-shelf (COTS) Real-Time Location (RTL) technologies sit at the bottom of the architecture, as represented by Fig.3.1. These include open RTL systems or sensors that can provide information on object locations within some (pre-equipped) environment. Openness in this context means that such a system or device needs to provide an Application Programming Interface (API), through which its data can be acquired. There are many sensors and RTL systems on the market which are closed –

stopping developers from incorporating them into, for example, sensor networks. Such practices are in place so that companies can sell their products in bundles, effectively disallowing their products to be easily extended and stopping others from (re-)using their devices in other systems or in a different context. Rather than 'being built for purpose', the aim of the proposed architecture it to provide flexibility to users in the selection of sensors. In this layered architecture the Data Layer, further described in the next section, is not bound to the particular source of information. As soon as it is able to receive sensor data via a software wrapper, whether in a push or a pull manner, the above layers can make effective use of the data. In short, any COTS RTL system (e.g. RFCODE's RTLS based on active RFID tags[1]) or sensors/sensor bundles (e.g. Fitbit[2]) can contribute to the system, providing they expose an open API.

3.2.2 Platform

As shown in Fig. 3.1 the platform is made up of three data processing layers. Events – based on their type – are handled in an appropriate layer. Problem-specific events are handled in the Goal Event Layer (GEL) and data acquired from COTS RTLS is handled in the lowest Data Layer (DL). Sensor specific logic resides in the Sensor Event Layer (SEL) and deals with error corrections and data redundancy. The data aggregation logic is expressed in the form of rules and each layer has its own rules for handling routines applicable to that layer. Due to their closeness to physical sensors rules in the DL and SEL layers should be formulated by an IT expert familiar with sensing technology, while rules in the GEL layer should be written in consultation with domain experts. The platform consists of a reliable storage component and consists of ontologies which describe events in each layer. Detailed explanation of the platform can be found in Section 3.3

Figure 3.2 visualises the concept of using rules to support decision-making based on

[1] http://www.rfcode.com/

[2] http://www.fitbit.com/, Greengoose[3]

low-level data sources. Atomic facts that the sensors provide (LLE) together with assignment information i.e. static mappings, are fused to form more complex conclusions (HLEs), which in turn get aggregated to form even more complex conclusions (HLEs), and so on. At the end of this process, the inferred facts are complex enough to aid the decision-making. In Fig. 3.2 low-level information such as: reader's location, tag discovery by this reader, current time and the fact that Mrs. Jones has Tag_123; are turned into facts of higher semantic enrichment (e.g. Mrs. Jones registered by Reader_1) in the process of data aggregation. The chain of inference continues until final conclusion is reached that 'Mrs. Jones took 10% less time to walk the corridor'. Domain experts e.g. physiotherapists can work with such complex information to better monitor progress of their patients and make informed decisions based on automatically captured and inferred evidence.

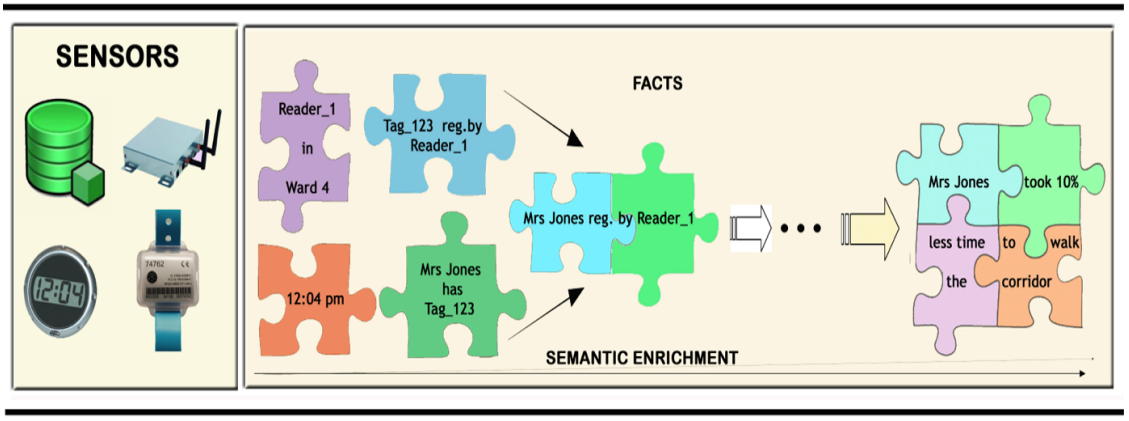


Figure 3.2: Reasoning Process: from sensor data to meaningful information

3.2.3 Applications

It is highly desirable for a SNM to ease application development, so that applications can be implemented quickly without any concerns about handling of low-level protocols. The application developer does not have to be an expert in sensor technology as the data produced by SNM does not differ from other data available on the Internet or

offered by various services. In this proposed architecture, the applications can make effective use of the semantically enriched data produced by the platform. Because all facts that are exposed comply with the *Event model*, all the data is uniform and can be easily parsed and analysed by various applications. The principles that govern the Semantic Web should also be applied to the data produced by SNMs – hence in this architecture applications can access semantically rich goal event data. Sensors, in fact, measure phenomena and conditions of the real world and hence should be supported by the semantic descriptions of these concepts in order to eliminate any ambiguities.

3.3 Platform's Layers

The proposed platform aims to provide a linkage between the COTS RTLS, which incorporate various sensors, and sensing-based applications. In this platform, the sensing task is divided between the three layers: data layer (DL), sensor events layer (SEL) and goal events layer (GEL). The lowest layer (DL) interfaces with sensors, the middle layer (SEL) deals with erroneous sensors' data and aims to shrink the rich volume of data, and finally the upper layer (GEL) performs the set (goal) sensing task via the execution of the encoded domain-specific logic.

The aforementioned separation of the sensing task onto the three layers easies the development, maintenance, deployment and execution of sensing-based applications, as each layer deals with a smaller portion of the problem i.e. its partitioned onto more manageable chunks. For example, the resolution of the sensing task of measuring patient walking speed in hospital environment and discovering involvement of walking aids is a very complex problem. To solve this problem with the aid of technology, many factors need to be consider and the list of requirements becomes very long. From sensors and WSN middleware selection, through mapping requirements in software, to testing. However, what if the sensors break or are no longer allowed in the hospital environment – does it mean the entire process needs to be repeated again? Using a

layered approach helps to separate physical hardware components (sensors) from the application logic. Therefore, situations where sensors feeding the system change, or the algorithm for completing the sensing task gets modified, or new requirements/goals are introduced, can be handled well in a modular approach. Even though the three layers provide logical separation, their common ground and the leading abstraction i.e. the *Event model*, provides consistency across the modules.

Sensors and sensor-based systems are often connected to the Internet and produce huge volumes of data. In order to contribute meaningful information to the web and realise the Semantic Web vision, they need to produce semantically rich data. No system can do so without the mapping of concepts they ought to express. Hence the representation of knowledge i.e. ontologies need to be present in a complete WSN middleware system. The platform's design principles are in line with this view and hence ontologies form an integral part of its architecture.

In comparison to the existing low-level SNM (GSN, ITA Sensor Fabric, SWE:SensorML), which simply provides sensor data, the proposed solution offers an easy way for data fusion (via rules), which is explicit to selected low-level SNM. Openness at the bottom layer makes this architecture more adaptable to given problem and more responsive to changing requirements – unlike high-level SNM (SS, SAM, SWE:SPS) are locked to particular low-level SNM. The *Event model* abstracts away sensors' heterogeneity by providing a common template for all events produced by sensors. Use of ontologies ensures that produced events are well-described and available for machines to process.

3.3.1 Data Layer

This layer interfaces with low-level SNM(s) and static data sources – Fig.3.3. The low-level SNM is capable of delivering sensor readings, however it is not complex enough to fulfil all the ten challenges for WSNM identified in [45], nor make application de-

velopment easy. The openness of this layer to various low-level SNM ensures that any sensors compatible with supported SNM can contribute to the system. Moreover, the data fusion logic in this architecture is explicit to the components in this layer, therefore enables aggregation of various data sources. The main purpose of this layer is to translate and inject static information and sensor data to the Sensor Event Layer in the form of facts.

Input: Physical sensors' signals and static DB data.
Processing: Translate inputs into facts, merge sensor readings with static information.
Output: LLEs and static DB's mappings in the form of facts.

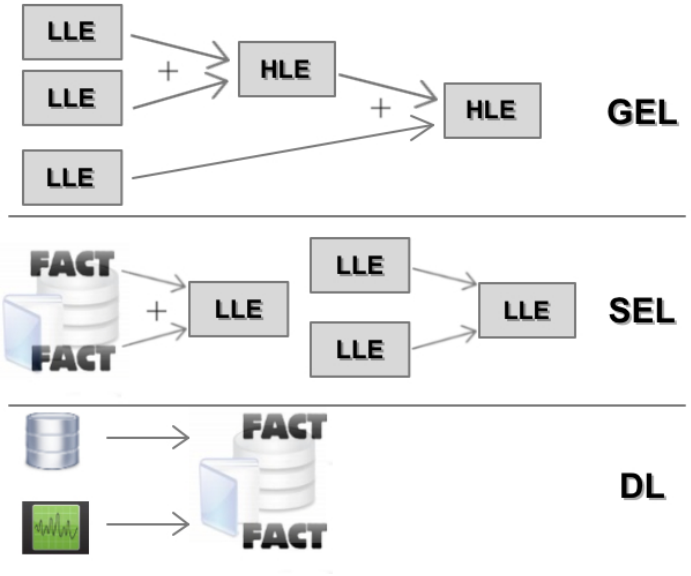


Figure 3.3: Data Processing in the Platform's Layers

3.3.2 Sensor Event Layer

The Sensor Event Layer (SEL) fuses static mappings and sensor-derived facts that are injected by the Data Layer to form low-level events – as represented by Figure 3.3. It abstracts away the network heterogeneity by representing this inferred data in a uniform format – compliant with the *Event model*. It is advisable that rules in

this layer are written by somebody familiar with the specific sensors used in the given setting, as this is where the data can be compressed and freed from any errors. For example, if the RTL system produces ObjectInMotion data every 2 seconds, a good practice would be to have an ObjectInMotionEvent with the start and finish times, rather than 30 ObjectInMotionEvents for every minute of object's motion. Also some rules of thumb can be used to correct the erroneous data e.g. if the hardware reports an object at location X for some period of time, then is not able to find its location for 2 seconds, and then again reports it at location X – it would be sensible to infer that this object was at location X for the whole time. Designing rules in the SEL layer also demands some knowledge of the environment in which the data is being collected, as this enables the rule's inventor to incorporate this knowledge into the error correction or data redundancy rules.

Input: Facts injected by the Data Layer.
Processing: Merge sensors' facts either with other sensors' facts or static mappings. Can also include rules for correcting erroneous sensors' data and for collapsing multiple events of the same type into one event.
Output: Low-Level Events (LLE).

3.3.3 Goal Events Layer

The Goal Events Layer (GEL) reasons over events injected by the SEL layer. The LLEs derived in the SEL layer are the 'building blocks' in the process of inferring HLE of different complexity. It is desirable to reuse existing LLEs and HLEs when designing rules in this layer in order to avoid unnecessary computations – as shown in Fig.3.3. Section 4.3 gives practical examples of how this is achieved in the pilot study described in Chapter 5. This layer could equally be called the 'expert's layer' as it forms a container into which experts, in the particular domain of interest, can apply their rules (in the current implementation rules are captured in CLIPS format

compatible with Jess). Domain experts should not modify the preceding two layers, as they perform fusions and injections of sensors' readings into the KB and translate facts into LLEs.

Input: LLE produced by the Sensor Event Layer, static mappings.
Processing: Transform LLEs into High-Level Events (HLEs). Reuses HLEs to form HLEs of greater complexity/meaning.
Output: High-level Events.

3.3.4 Designing the Platform for Flexibility

The platform's design was driven by the idea that all components of the system should be highly independent of each other. Decoupling the logic and processing amongst various layers eases system development as changes can be made to any of the subcomponents without having an effect on other subcomponents. Hence the proposed system architecture would allow a fair degree of flexibility or, in other words, degrees of freedom to the user in the following areas:

Sensors: The sensing platform should not be bound to the specific types of sensors or SNM but should rather be open i.e. extensible and support agility. Posing such constraints onto any sensing platform limits its use and reuse in other scenarios or domains. Such a practice is against the *adaptability* criterion of the ten challenges for sensor network middleware [45]. For any SNM it is highly desirable for it to work with a wide range of sensors, as building systems fit for purpose is not the way forward. Hence the proposed architecture remains open to any low-level SNM which can provide the data either in the push or pull manner in some parsable format. Consequently, the choice of sensors is on the user's side and if they wish to incorporate new sensors to the system, they may do so providing that these devices are not 'closed' i.e. are interfaceable. Very important is the

decoupling of physical hardware i.e. sensors from software. Although hardware failures may limit the system's functionality, the software would still execute, regardless of the performance of physical sensors.

Rules: The rules are spread across the three event processing layers. The logic encoded in the form of rules is transparent and easily understandable as it is just a mapping of 'IF -> THEN' statements. The data aggregation logic is explicit to the SNM (handled by the Data Layer) as it lives in the tiers above. This approach meets the *data fusion* challenge for SM [45] as it allows for facts' fusion and hence inference of new facts about the real world. All rules persist in one container which increases the transparency of the *data fusion* logic, as opposed to spreading this logic across the various low-level SNM which feed the Data Layer. The SEL layer holds sensor specific rules – which should be programmed by the system's administrator due to their technical nature. On the other side, the GEL layer holds the 'goal events' for which rules should be designed in cooperation with the expert in this particular domain, as they may lie outside of the administrator's understanding. Moreover, if the front-end rule editor was to be implemented, the experts could be managing the rules in the GEL layer on their own.

Ontologies: The assumption made here is that every fact/event name that the system deals with, corresponds to a class in the provided ontologies. The platform is divided onto a number of sub-layers as it is visualised by the Fig.3.1 and similarly the ontologies are organised into layers. The set of concepts represented by the system can be extended by specifying new or extending existing models in any of the layers. Again, the layered ontological approach provides a degree of independence for every set of concepts in each layer and hence makes it easier to modify, if the need arises, without affecting concepts in other layers. Capturing concepts in the form of ontology allows for human's expert knowledge to be encoded into the system, therefore addresses a very important challenge listed in

[45] called the *application knowledge*.

Apart from the aforementioned degrees of freedom in the choice of sensors, rules and ontologies, the platform is not bound to particular software components. The COTS RTLS layer can consist of many types of sensors handled by different low-level SNM. Similarly, the platform is not bound to the specific rule-based system (RBS) and is able to work with various RBSs. Finally, the system is not bound to any particular domain as both rules and ontologies can be developed in cooperation with field experts. This flexibility of the platform is illustrated by the two pilot studies presented in Chapter 5.

3.4 Implementation of the Platform

The RBS2P interfaces with low-level SNM (GSN [1]) via HTTP requests. However, other SNMs such as: ITA Sensor Fabric, SWE compliant middleware, or cloud-based services (such as Xively) could replace or could be used together with GSN. Moreover, various static mappings and inferred facts are stored in the MySQL DB, which can be interfaced through Java's JDBC API. The core of the system i.e. the RBS accepts rules and facts in the CLIPS format. Some features of the system, such as the CSV export, are serialised via use of scripts. The ontologies that describe the events that the system processes are available in OBO [4] and OWL[5] formats in Appendix B.

The entire platform has been implemented in Java due to its platform independence and the availability of libraries. The Rule-Based System (in the current implementation it is Jess[6]) supports the evolution of the sensor data from low-level events to high-level events by applying aggregation rules specified in its knowledge base. High flexibility of the system is achieved by allowing different RBSs to be used instead of Jess e.g. SweetRules[7]. The more complex the HLEs the higher their semantic enrichment.

[4] http://www.obofoundry.org/
[5] http://www.w3schools.com/rdf/rdf_owl.asp
[6] http://www.jessrules.com/
[7] http://sweetrules.projects.semwebcentral.org/

Sensor-derived event data is permanently stored in the MySQL DB, in an Event table which schema is compatible with the *Event model* described in the next chapter. We assume that every fact/event name that the system deals with, uniquely matches name of the class in system ontologies.

At the other end, the semantically enriched event data is exposed to applications. Applications, in order to consume the event data, can either directly access the DB or ask for the data to be provided in an appropriate format. Because all the facts that are stored in the DB comply with the *Event model*, all the data is uniform and can be easily parsed and analysed by various applications. In short, the RBS2P takes fairly low-level sensor readings and transforms them into meaningful semantic event data, which can be 'consumed' by applications. Moreover, all the software components that make up the platform in its current form are open source, which is another important factor for users. Licensed products can be very expensive and their price may form a barrier for new users.

3.5 Benefits of the Proposed Architecture

One of the evaluation criteria for WSN middleware is how the proposed solution meets the ten identified computer science challenges for sensor network middleware [45]. In the proposed architecture *security* challenge is not tackled by the system but rather is left to the low-level sensor network middleware that feeds the system. However, with the layered architecture in place, another layer could be dedicated to dealing with the security issues. *Scalability, dynamic topology* and *QoS support* have to be handled by both the low-level SNM and the platform. The proposed system architecture helps to accomplish these addressed challenges in the following ways:

Abstraction Support: Each event (represented in RBS as a fact) is an interpretation of data. An expert, via rules, determines how data is interpreted. The event

abstraction provides a uniform model for all facts, which simplifies the task of rule' implementation and application development.

Data Fusion: Rules fuse facts (sensor and static data) to infer new facts about the real world. In the proposed architecture data fusion logic is explicit to low-level SNM and is expressed in a widely used rule format (CLIPS in the current implementation) – rather than in a new specification with which programmers may not be familiar (another spec. to learn). Fusing events of the same structure (based on the *Event model*) makes task of data fusion less complicated.

Resource Constraints: Rule approach to SNM helps to deal with restricted resources. If use of one sensor type e.g. camera is restricted in some environment, then in many cases alternative rules can be developed using other sensors.

Dynamic Topology: Dynamic configuration of WSNs often results in node failures, mobility and communication limitations. In a modular approach physical sensor failures do not affect the execution of other layers. Moreover, multiple rules can be assigned to a particular problem implementing different logic and/or making use of different sensor compositions to overcome nodes' limitations. The responsibility of handling WSN's dynamic topology also resides on the low-level SNM side, which has to deal with cases of nodes coming in and out of range i.e. handling of nodes disconnections and re-connections.

Application Knowledge: Human knowledge is encoded into an automated system, which allows for integration of domain expert's knowledge and hence makes it a part of the solution. Therefore, it allows for tailoring the platform to the user-defined needs.

Programming Paradigm: SNM should be based on simple concepts, which are uniform across many domains. Rule approach to SNM does not introduce a new standard but rather applies a technique, which has been used for decades to solve a variety of problems. Supported by the uniform *Event model*, the proposed

platform makes use of a logic programming paradigm, well suited to the problem of activity monitoring. The ability to capture (in software) human-generated procedures for solving problems is a key.

Adaptability: Applicable to many activity monitoring scenarios. Rules can be written by non-programmers and can be easily amended if the requirements change. The layered approach separates working of physical sensors from rules' execution, hence provides flexible solution, which can respond well to changes.

Scalability: The proposed platform is not limited to a specific low-level SNM, hence allows for many sources to feed it. When problem becomes large i.e. thousands of sensors feeding the platform, parallel computing can be used to spread the workload between many machines. This is possible by separating the layers between many computer units and also via use of modules in the RBS component of the platform. Dividing a large computing task onto smaller, manageable chunks is supported in this architecture. The responsibility of handling scalability also lies on the SNM side, as it needs to be capable of managing connections with large number of sensors simultaneously.

QoS: There are many QoS features to be considered: response time, availability, node's failures, explanation of reasoning, etc. In the proposed solution multiple combinations of rules and facts can often answer the same query. Solution can be explained by retracing the reasoning, as the *Event model* supports recording of evidences (sub-events make up events). Response time can be maximised with the use of modules in the RBS component and also via use of parallel computing. These qualities also depend on how low-level SNM supports QoS features.

The low-level SNMs (GSN, ITA Sensor Fabric, SWE:SensorML) introduced in chapter 2 all fail in some of the above areas, especially to address: *application knowledge, adaptability* and *abstraction support*. Even though each of these solutions uses some powerful abstraction it does not introduce a model, which is uniform for all sensors i.e.

does not impose a fixed structure which is easy to work with. Application knowledge is injectable to the middleware, however the programmer needs to learn another specification, which is specific to the chosen low-level SNM. How would one solve the problem of using three sensors, each of which is only supported by one of the named SNMs? By writing logic in three different specifications? And then how could the three sensors' data be combined to address some sensing task? These middleware, together with the high-level SNMs introduced in chapter 2(SS, SAM, SWE:SPS) are not very adaptable and do not consider many real-life scenarios, such as sensors being locked to specific middleware/vendor, or changing application's requirements. These high-level SNMs only support fusion of data for their own chosen low-level SNM and hence are locked at the bottom layer i.e. lack flexibility.

Further evaluation of the proposed platform can be done by referring back to the definition of SNM – "the main purpose of middleware for sensor networks is to support the development, maintenance, deployment, and execution of sensing-based applications" [58]. Low-level SNMs mentioned earlier, do not support these tasks very well as application development with them is time consuming. They are very close to sensors but further away from applications and hence do not easy the task of application development and maintenance. On the other hand, high-level SNMs introduced earlier are very close to applications but are locked to a specific set of sensors supported by only one low-level SNM they use. Hence development of some applications may not be possible, if desired sensors are not supported. This classification of SNM and their leading abstractions is represented by Fig.2.1.

In contrast, the presented platform is open at the top, which means that new rules can be added easily in order to extend system's functionality. It is also open at the bottom, so that the user is not limited in the choice of sensors. The data fusion logic in this layered architecture is written in the form or rules and is explicit to the low-level SNM used and therefore if multiple SNMs are used the rules remain unchanged. The three degrees of freedom in the choice of: sensors, rules and ontologies make the

system more flexible and extensible, while remaining open to the changing requirements. The layered architectures easies the system's development as each layer can be improved/extended independently of others and changes do not affect the working of other layers. Moreover, the physical hardware is decoupled from software and therefore sensor's failures do not affect software execution. Finally, there are some extra benefits that come from incorporating an RBS into any system: modularity, uniformity and naturalness [48]. In short, using a Rule-Based System with low-level SNM and ontologies adds the desirable functionality and forms a good approach to the problem of Activity Monitoring.

Apart from evaluating the proposed architecture against the ten identified challenges for SNM and against the definition of SNM, one may want to use software architecture evaluation methods such as ATAM (Architecture Tradeoff Analysis Method)[35]. This method concentrates on evaluating an architecture for suitability if terms of the quality attributes that make up system utility. Hence, the proposed architecture can be evaluated in respect to the following standard quality attributes (definitions are based on [9]):

Performance: Refers to the responsiveness of the system or the number of events processed in some interval of time

Reliability: The ability of the system to keep operating over time.

Availability: It is the proportion of time the system is up and running.

Security: The measure of the system's ability to resist unauthorised attempts at usage and denial of service while still providing its services to legitimate users

Modifiability: The ability to make changes to a system quickly and cost effectively.

Portability: The ability of the system to run under different computing environments. These environments can be hardware, software, or a combination of the two.

Functionality: The ability of the system to do the work for which it was intended.

Variability: How well the architecture can be expanded or modified to produce new architectures that differ in specific, preplanned ways. Variability mechanisms may be run-time (such as negotiating on the fly protocols), compile-time (such as setting compilation parameters to bind certain variables), build-time (such as including or excluding various components or choosing different versions of a component), or code-time mechanisms (such as coding a device driver for a new device).

Subsetability: Is the ability to support the production of a subset of the system. It enables incremental development and is a special kind of variability, mentioned above.

Conceptual integrity: It is the underlying theme or vision that unifies the design of the system at all levels. The architecture should do similar things in similar ways. Conceptual integrity is exemplified in an architecture that exhibits consistency, has a small number of data and control mechanisms, and uses a small number of patterns throughout to get the job done.

Some of the above attributes, especially the first four, can only be measured once the architecture is implemented. However, the architecture can take these factors under consideration. *Performance* in RBS2P's architecture can be improved (if needed) via use of modules (in the rule-based system) or by distributing the system's layers across multiple computers. Such layered architecture increases system's *reliability* and *availability*, as various layers can operate independently, which is particularly useful in case of components' failures. *Security* aspect is beyond the scope of this system and hence is not considered. *Modifiability* is one of the key principles the proposed architecture was based on - openness at the bottom to new sensors and openness at the top to new requirements/adaptability to various domains. *Portability* relates not only to the architecture but also to the implementation choices made and hence is measurable on the implemented system. Again, this quality is also important in the RBS2P implementation and hence OS-independent software components were favoured in its implement-

ation. Assessment of the architecture's *functionality* can be best accomplished either via simulation or by conducting reliability and validity studies. The proposed system architecture supports *variability* – more layers can be added if needed or existing layers can be modified with minimal effect on other layers – and *subsetability* – the system can be built incrementally, layer-by-layer, starting from the bottom. *Conceptual integrity* is achieved via the *Event model* which bridges all the system layers together.

This chapter introduced the model, conceptual architecture and platform used in the remainder of this thesis. Implementation choices were presented and justified against the desirable features of WSN middleware. Finally, benefits of the proposed platform's architecture were discussed. Having defined main concepts and leading design principles, models behind the implemented system can be presented in the next chapter, together with detailed explanation of layers' functions and examples of rules implemented in each layer of the platform.

Chapter 4

Modeling & Implementation

This chapter discusses the motivation, modeling choices made and models that were created to represent various concepts within the platform. Functions of the platform's layers are explained together with advantages of the layered approach. The chapter also provides an insight into the semantic encoding of the events inferred by the system and some example pseudocode of rules that the system implements.

4.1 Motivation

A single sensor provides only partial information on the actual physical condition measured, e.g. an acoustic sensor only records audio signals. For an application to reason over sensor data, raw sensor readings have to be captured and often aggregated to form a more complete picture of the measured real-world phenomenon. Sensor network middleware aids this process. However, existing SNM do not give the user – who can be an expert in some area that is not computer science – an opportunity to easily specify data aggregation logic themselves. Supplementing existing low-level SNM with a rule-based system can address this problem in the domain of activity monitoring (AM).

4.2 Events

As introduced in Section 3.1 everything that happens around us can be classified as events. An arrival of a person or a car, at a particular location and time, is an event. Brushing teeth or showering; shopping with friends or a trip to a cinema; or even a toast popping out of the toaster – these are all examples of events. An *event* "consists of some temporal and spatial boundaries subjectively imposed on the flux of reality or imagination, that we wish to treat as an entity for the purposes of making statements about it. In particular, we may wish to make statements that relate people, places, or things to an event."[1]

Basically, our life is made up of a sequence of events of different complexities that either we generate or witness. Sensors, which typically try to measure some existing condition, are a medium for recording conditions of some occurring events. A single temperature reading is an event. This single event can contribute to discovery of a fire event – if the temperature measured over a period of time suddenly reaches 100 degrees Celsius. Humans are able to observe events through their senses; software systems, on the other hand, are capable of inferring events (with some certainty) based on the information they are fed with – typically via use of sensors. Section 4.2.1.1 of this chapter describes the *Event model* (EM) and its properties in more details.

The characteristics of an event are fairly intuitive, as every event ought to: have a name by which humans classify it, occur in a certain place and time, involve a person(s) and/or an object(s) and can be made up of *sub-events*. For example, a talk by a keynote speaker at a conference is an event but so is the entire conference. Hence we classify events as either low-level events (*LLEs*) and high-level events (*HLEs*). LLEs are atomic i.e. do not contain sub-events or consist of sub-events of the same type, and other (complex) events can be thought of as high-level events (HLE)s. This classification is intuitive and very easy to grasp but above all it helps to understand the process of events' composition. Moreover, in an implementation of a system capable of dedu-

[1] http://linkedevents.org/ontology/

cing events, properties of an event such as its ID or references to supporting evidences (provenance) is beneficial. The Rule-Based Semantic Sensing Platform (RBS2P) performs this process of events inference and supports provenance as a way of explaining its reasoning.

Examples of LLEs and HLEs presented in this section are authentic and come from the rehabilitation domain i.e. the two case studies presented in Chapter 5. These events (facts in rule community terminology) are represented in CLIPS format, and have the following structure. Each fact is enclosed in parentheses. Fact name is followed by a set of slots enclosed in parentheses. Each slot represents one property of a fact i.e. name of a property followed by a single space and its value. For example, `(Person (age 27))` represents a `Person` fact, which in its `age` slot carries the value of 27.

4.2.1 Layered Event Models

This section describes the layered *Event models*, where each subsequent model 'builds up' on the preceding one. The *Event model* makes up the lowest layer on which all the other models depend. The medium layer is the *Real-Time Location (RTL)* model and the 'high' layer is the *problem-specific* model. While the *Event model* is domain independent and forms the foundation of the sensing system, the other two layers are tied to a different extent to the domain for which the system is built. The RTL model can be re-used across multiple domains, which deal with the subclasses of the activity monitoring problem, while the problem-specific model is tied closely to the application area. Such a modular approach increases the reusability of the system and eases its maintenance. In conjunction with a rule-based system, it allows human expert knowledge to be encoded into the system, therefore addresses the *application knowledge* challenge listed in Section 2.1.

4.2.1.1 Event Model

The key concept for all the models presented in the reminder of this section is the generic *Event model*. In every domain events share common characteristics. Every event has to happen at some point in space and time and can involve either people or objects, or both. The *Event model* in Fig.4.1 was inspired by the *Event ontology* (EO)[2] and the LODE ontology[3] and is further discussed in the reminder of this section.

Figure 4.1: The Event Model

The *Event ontology*, which was developed in the Centre for Digital Music in Queen Mary, University of London has the following properties:

place Relates an event to a spatial object.

time Relates an event to a time object, classifying a time region (either instantaneous or having an extent). ...

agent Relates an event to an active agent (a person, a computer, ...).

factor Relates an event to a passive factor (a tool, an instrument, ...).

product Relates an event to something produced during the event (a sound, a pie, ...).

sub_event This property provides a way to split a complex event (for example, a performance involving several musicians) into simpler ones (one event per musician). [53]

[2] http://motools.sourceforge.net/event/event.html
[3] http://linkedevents.org/ontology/

Even though the EO was developed for a specific purpose [53], it has been proven useful in a wide range of context such as: talks in a conference, description of a concert or chords played in a Jazz piece.

The LODE ontology presented in [59] is described as "...an ontology for publishing descriptions of historical events as Linked Data, and for mapping between other event-related vocabularies and ontologies". In this ontology, the Event class has the following properties:

atPlace [The value of this property is] a named or relatively specified place that is where an event happened.

atTime [The value of this property is] an abstract instant or interval of time that is when an event happened.

circa [The value of this property is] an interval of time that can be precisely described using calendar dates and clock times.

illustrate [The value of this property is] an event illustrated by some thing (typically a media object).

inSpace [The value of this property is] an abstract region of space (e.g. a geospatial point or region) that is where an event happened.

involved [The value of this property is] a (physical, social, or mental) object involved in an event.

involvedAgent [The value of this property is] an agent involved in an event. [53]

The LODE ontology consists of many relevant concepts, such as: atPlace, atTime, involved and involvedAgent properties, however it does not support the composition of events i.e. sub_event property in the EO.

The *Event model* in Fig.4.1 is derived from the two ontologies discussed above and has the following properties:

atPlace Informs where an event took place i.e. geographic location(s) or some named location(s).

atTime Specifies at what time (can be a single value or timespan) an event took place.

hasID Labels an event with a unique ID by which it can be referenced.

evidencedBy Provides references to evidencing sub-events.

involved Physical object(s) involved in an event.

involvedAgent Lists agent(s) involved in an event.

The above properties of the *Event model* make it well-suited for AM scenarios – as these are mostly concerned with entities involved, places where activities occurred, time in which they happened and evidences supporting conclusions. However, in order to be suited for any sensor system it would need to include properties, which describe sensor's value and unit. The URL to all layered event models (implemented in OBO and OWL formats) can be found in Appendix B. The key property of the EM is the `evidencedBy` relation, which in the *Event ontology*, is called the `sub_event`. The purpose of the `sub_ event` property in the EO is to split complex events into simpler ones e.g. a performance of several musicians into one event per musician. In our case, the function of the `evidencedBy` property is to link inferred HLE to their supporting evidences i.e. LLEs that make up the higher-level event. This event composition aids the evolution of LLEs to HLEs and also enables traceability of evidences. Its main difference from the `sub_event` relation is the fact that the concluding HLEs can, on the first look, have no semantic similarities to the LLEs that formed them.

In order to support the traceability of evidence an event (in the EM) has `hasID` property. Every event needs to be labeled with a unique ID by which it can be referenced/identified. Referencing sub-events has many advantages over including their entire data in complex events. Firstly, it reduces the memory requirements for the system, as only a unique id of sub-events is stored in the `evidencedBy` slot. Secondly,

there is no data duplication in the system, which also reduces the memory consumption. Finally, it facilitates data re-use and hence a LLE that evidences some HLE can become an evidence for another complex event.

The `atTime` property in the EM is equivalent in meaning to the `time` property in the *Event ontology* and hence the value of this property is an instantaneous time value or time interval when the event occurred. Other properties of the EM, namely: `atPlace`, `involved` and `involvedAgent` are equivalent in meaning to the corresponding properties in the LODE ontology.

According to Sheth and Perry [61] a semantically rich sensor network needs to provide three-dimensional information (spatial, temporal, and thematic) essential for discovering and analysing sensor data. They also share the view that events are fundamental for realising the Semantic Web vision as they relate entities in space and time.

4.2.1.2 Real-Time Location Model

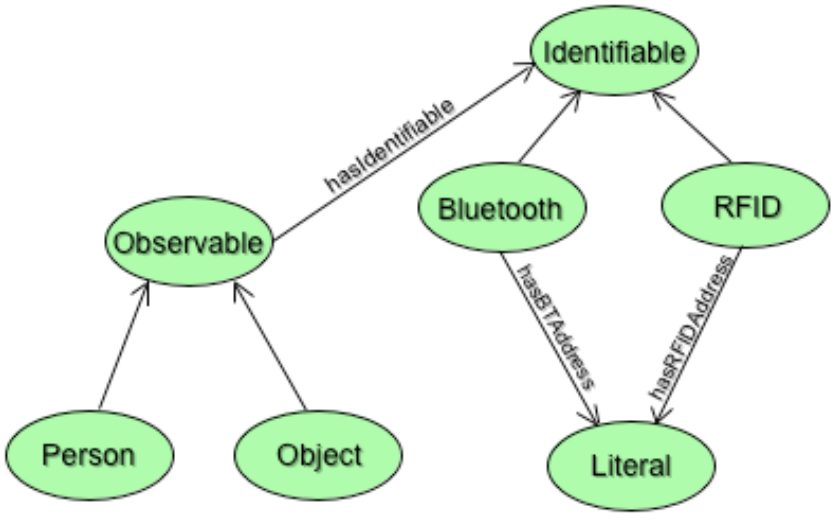


Figure 4.2: The RTL Model

The *real-time location* (RTL) model is illustrated by Fig.4.2. This model is somewhat similar to the *semantic sensor network's* (SSN) model [14]. In SSN, *Observation* is

observedBy a *Sensor*. In the *RTL model Observable* such as *Person* or *Object* (which are referenced in the *Event model*) *hasIdentifiable* of type *Identifiable*. The *RTL model* is very minimalistic and only includes *Bluetooth* (BT) and *RFID Identifiables* and their respective properties i.e. *hasBTAddress* and *hasRFIDAddress*, which values are simply Literals. However, there are other forms of identifying objects/people such as: visual, ultrasound, break beam sensors, etc., which could be added to this model. The key message is that for an entity to be considered an *Observable* it needs to be identifiable via one of the subclasses of the *Identifiable* class – analogically to SSN model where observations are observed by sensors.

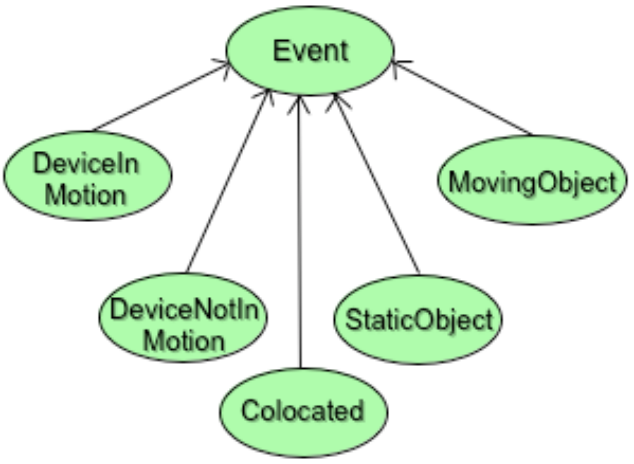


Figure 4.3: The RTL Classes

Currently there are five classes in the *RTL model*, which are subclassing the *Event* class. Fig.4.3 illustrates this relation. *DeviceInMotion* and *DeviceNotInMotion* relate to the concept of *Identifiables*. Simply if something is of an *Identifiable* type e.g. RFID tag or BT-enabled device, its RTL data is described via use of the two aforementioned classes (LLEs). However, if a person or an object is wearing one of the *Identifiables* it can be thought of in terms of *StaticObject* or *MovingObject* (LLEs). Therefore a fusion of a LLE *DeviceInMotion* with some assignment information (mapping of person's/object's identity to an *Identifiable* they wear) would result in an inference of a *MovingObject*

(LLE). This process of fact aggregation is further presented in Secion 4.3 with the support of pseudocode.

The next step would be to cover the problem-specific model present in the GEL layer. However, due to the fact that model in this layer is domain-dependent, it does not belong to the empty system. On the other hand, `Event` and `RTL` models are parts of the platform and form an integral part of the SNM solution in the area of activity monitoring.

4.2.2 Low-Level Events

Low-level events, as the name suggests, are those that are atomic i.e. not composed of any sub-events (derived directly from sensor readings), or are composed of sub-events of the same type. Practically, there is no need to keep their provenance, as they are based directly on sensor readings and simply if the sensors are reliable it follows that the events they produce are also valid. Here is an example of an LLE that the system processes:

```
(DeviceInMotion (atPlace "100")(involvedAgent "nil")(involved "IRCODE00098127")
(evidencedBy )(atTime 1337078615758)(id "gen4578794"))
```

From the example above, we can see that apart from the empty `evidencedBy` slot, the `DeviceInMotion` has a "nil" value in the `involvedAgent` slot. This is another characteristic of low-level events. Due to their atomic nature, they can either describe a person (`involvedAgent`) or a thing (`involved`) that is a subject of the event, and not both at the same time. The `involved` property has the identifiable's address – in this case RFID tag number. The `atPlace` and `atTime` slots are non-empty as, by its definition, every event occurs in a certain place, at a certain time. Additionally, the `id` slot of every event enables its identification and traceability.

4.2.3 High-Level Events

High-level events, as opposed to LLEs, are complex i.e. made up of/derived from other events. The 'building blocks' of a HLE is any combination of other HLEs and LLEs – as visualised by Fig 3.3. Therefore, an already inferred HLE can contribute in the process of inference of higher-level events. This notion of facts' reusability is very important, as it shortens the rules and helps to avoid unnecessary computations. Rules that are involved in the inference of HLEs should be developed by/in collaboration with experts in the particular field. By encoding human's expert knowledge into the system, we make sure that the system performs in the way described by domain experts. Below is an example of an HLE that the system produces:

```
(Colocated (atPlace "100")(involvedAgent "patient_2" "staff_1")(involved "nil")
(evidencedBy "gcrgen35534" "gcrgen27757")
(atTime 1337078615758 1337080599868)(id gen35545)))
```

The `Colocated` event simply captures the co-existence of two entities in space and time – Fig.4.8. Therefore, the `atPlace` slot only consists of one value, which is translatable to meaningful location name if such mapping exists in the DB. On the other hand, the `atTime` slot is complex i.e. consists of two time values, as it defines an overlapping time for the presence of `patient_2` and `staff_1` at location `"100"`. This event has a "nil" value in the `involved` slot. However, it has two values in the `involvedAgent` slot, which suggests that this is a complex event. Moreover, there are two entries in the `evidencedBy` slot, which is another indication that this fact is an HLE and through these two IDs the tracking of premises is possible. Therefore, if the user is not happy with the conclusions, their provenance is traceable i.e. explanation of reasoning available. Rather than being a 'black box' the system is transparent and hence enables fine-tuning of its inference units.

4.2.4 Using the Event Models in the System Layers

4.2.4.1 Data Layer

The Data Layer can be thought of as the 'data harvesting' layer of the system. It is the only point at which both sources of information: static and dynamic, find their way into the system.

Moreover, it has the functionality to start and stop the data collection process and also to export the gathered data to the supported format.

The purpose of the Data Layer is to translate all the forms of information that are relevant into facts and to inject these facts into the Working Memory (WM) of the rule-based system (RBS). There are two sources of information that feed this layer: database and SNM. The DB holds static assignments of all the observables to their device address and other relevant information, such as default location. The information carried by the observables is used, for example, to unify the `DeviceInMotion` facts with the owner of this device. As represented by the Sequence Diagram in Fig.4.4, once all the observables are acquired from the DB and get injected into the WM, the RBS' engine is run so that any incoming events e.g. *DeviceInMotion* can trigger rules to fire.

The SNM loop in the Fig.4.4 pulls the SNM's data. In this loop, the sensor network middleware gets periodically queried for the most recent data. This data is then added to the WM of the RBS component, which triggers the relevant rules to fire. The last step in this loop is to put this process of data pulling to sleep for the specified sampling rate time less the loop interaction's processing time – in order to ensure that the processing time has no effect on the sampling frequency.

The Storage option in the Fig.4.4 provides reliable persistence. The function of it is to periodically back up the data that persists in the WM of the RBS. In this way, in the case of some unforeseen circumstances such as the power cut, only a very little fraction of data is lost. It also frees up the WM by deleting all the 'expired' events, i.e. events for which their `atTime` value is more than the SNM's sampling rate. Events that are more recent have to remain in the WM as rules in this layer apply the data redundancy techniques on newly arrived data. The code inside the Storage option is executed as frequently as the storing rate, so that the data backup does not happen too frequently and hence does not bring the unnecessary overhead.

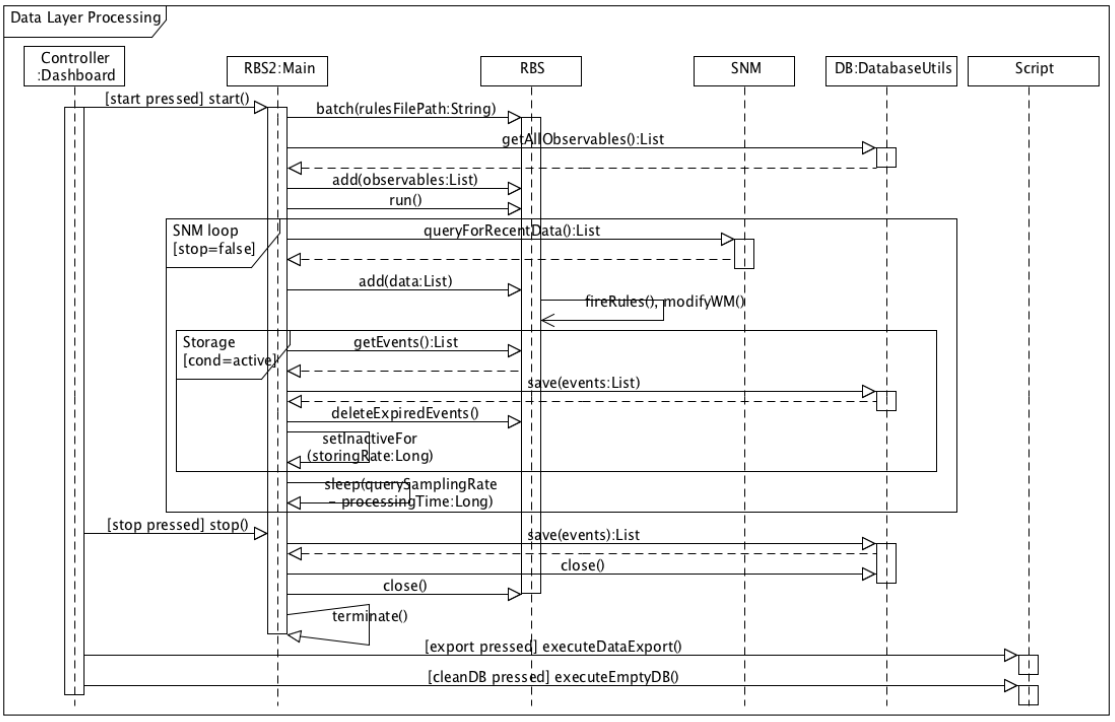


Figure 4.4: Sequence Diagram: The Data Layer

4.2.4.2 Sensor Event Layer

If platform's layers are distributed (e.g. data is collected in an 'offline' mode) then the Sensor Event Layer (SEL) is responsible for loading up the data from a database (or from other form of data storage e.g. cloud or file), so that rules which belong to this layer can be applied to this data. This layer's rules serve a purpose of fusing events with other events or static mappings. Moreover, they can provide logic for correcting erroneous data if such need arises.

Firstly, as it can be seen in Fig.4.5, the list of all observables that persist in the DB is acquired and gets injected into the WM of the RBS. This step is optional, as the RBS2 system can either work in the 'offline' or 'online' mode. In the 'online' mode there is no purpose in pulling the list of observables to the system again, as this is something that takes place in the Data Layer. On the other hand, in the 'offline' mode, such a need arises, as only the event data gets stored in the DL layer of the system. In some cases, the details of the observables that the system is tracking may change over time, hence it is desirable that the static assignments that persist in

the DB are not fixed and can be changed/updated if such need arises.

Secondly, Event data for each of the observables is pulled from the available data storage. In this process the window should be specified, as days worth of data might be available. This event data gets injected back to the WM. Again; if the system works in the 'online' mode then there is no need to put this information back into the engine, as it would already be there. Once all the relevant data is loaded into the RBS, rules are loaded into the KB and the engine is run. It is important that these steps are performed in this sequence so that every piece of the information is considered in the inference process. Finally, it is optional to save the entire engine to a file so that all the data is backed up before going through the next layer. Worth adding is that both event data and observables are not just dumped to the engine but instead are allocated into the appropriate modules, which correspond to their ontological models. This modular approach to the problem helps to tackle *scalability* challenge for SNM 2.1.

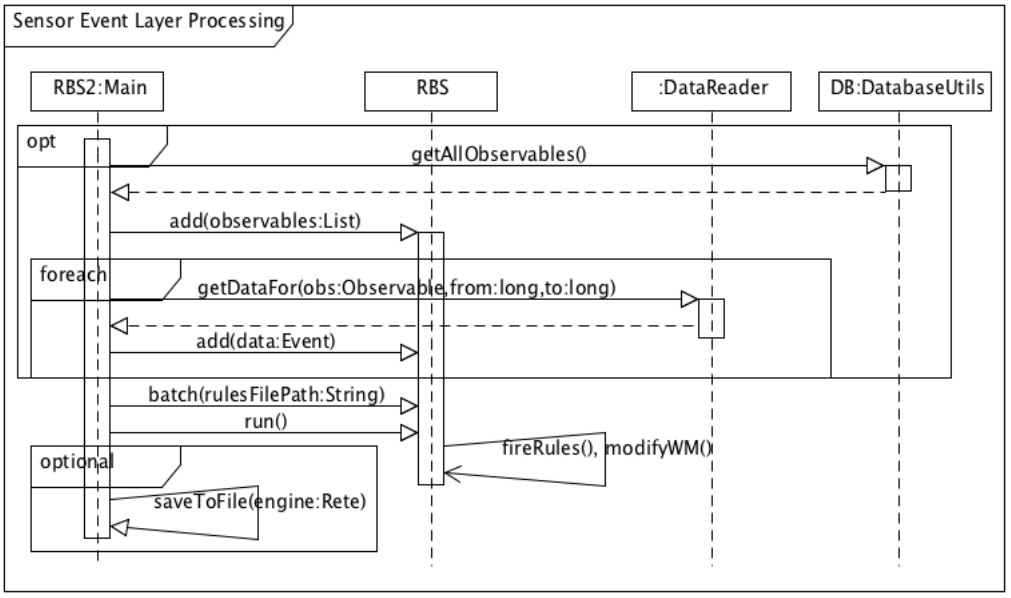


Figure 4.5: Sequence Diagram: The Sensor Event Layer

4.2.4.3 Goal Event Layer

The GEL layer is the 'highest' layer of the system. It typically consists of rules, which are specific to the domain of interest. It can either work disjointly with the SEL layer by loading

its data via available storage option, or it can simply be applied after the SEL's processing is over. The main function of this layer is to infer HLE from LLE injected by the layer below. GEL layer would typically be composed of expert's rules, which can equivalently be called goal rules – as they formulate the end products/goals of the system. Allowing experts to work on the rules' formulation or to formulate rules on their own, addresses the *abstraction support & application knowledge* challenges for SNM listed in 2.1 as experts provide the logic for data interpretation and inference. Moreover, all layers of the platform, via application of their rules, address the *data fusion* objective for sensor systems.

As it is illustrated by Fig.4.6, apart from applying expert's rules to the data it deals with, the GEL layer is also responsible for exposing the events that persist in this layer. All these events are queryable and should be delivered in an easily consumable semantically rich format.

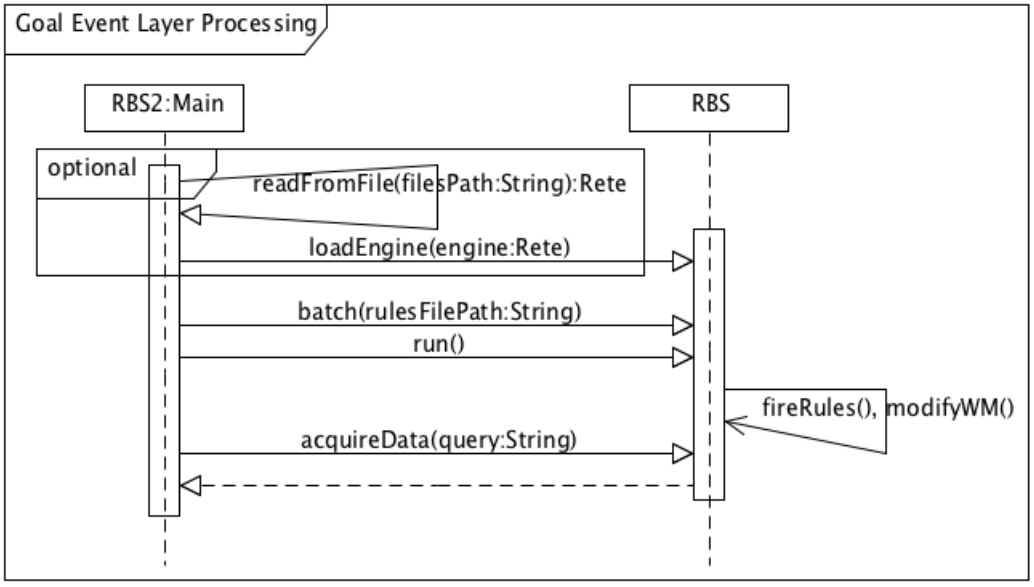


Figure 4.6: Sequence Diagram: The Goal Event Layer

4.2.5 Semantic Encoding

Fig.4.7 represents the concept of transforming sensors' and static data into consumable, semantic HLE and LLE. At the bottom of this pyramid are all the sensors' and static data, which

volume is usually huge. 'Step 1' label can be interpreted as the function of the DL layer – transformation and translation of data into LLE. 'Step 2' label represents the process of fact aggregation performed in SEL and GEL layers. Finally, the last step is the *semantic encoding* of HLE and LLE. With every step in this process of data aggregation, distributed amongst the three layers of the system, grows events' complexity and their semantic enrichment. Data at the top of the pyramid is the end product of the system (goal events) and this is where semantic events are ready to be exposed to applications via API. Moreover, the further up in this pyramid the lower the volume of the data, as data fusion and data redundancy strategies are mapped into rules. The produced data can be modelled in e.g. RDF or OWL and serialised either in XML or in JSON format (smaller message size).

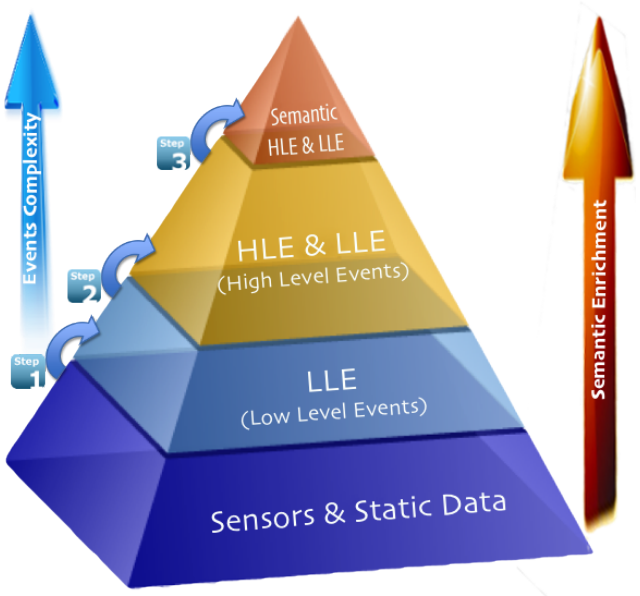


Figure 4.7: Semantic Enrichment of Events

4.3 Implementation

The purpose of this section is to provide the reader with an insight into rules, which form an integral part of the platform. The empty system would be populated with the presented rules, as they provide building blocks for any domain concerned with activity monitoring e.g. rehabilitation, surveillance, etc. Rules are presented with the aid of pseudocode, which provides enough

details to reveal the complexity of the problem, yet hides implementation specific details, as in principle any rule-based system can be used in platform's implementation. These rules have been implemented (in Jess[4]) and hence have their grounds in reality. The RTL system used for testing these rules is the RFCODE[5] active RFID solution incorporating active RFID tags and RFID receivers.

Rule-based system, embedded in the RBS2P, supports the evolution of sensor data from LLEs to HLEs by applying aggregation rules specified in the Knowledge Base – fulfils the *data fusion* requirement listed in Section 2.1. The inference rules are spread across the three layers of the system, depending on the origin and the complexity of events they deal with. The closer to the sensors (bottom layer) the more sensor-specific the rules are as they deal with low-level sensor readings. On the other hand, the rules that are closer to the application layer are more complex, contain field expert's knowledge and make use of the events inferred in the layers beneath.

The notion of events re-use across multiple layers was one of the leading principles when designing and implementing the RBS2P. This approach is somewhat similar to the Semantic Stream's (SS) approach [68], where rules can be thought of as equivalents of semantic services – take zero or more input streams and produce one or more output streams. The main difference is the fact that rules produce events that do not form a stream but rather get asserted into the WM without any ordering – a bag structure as opposed to a stream made up of the same events arranged into a non-decreasing sequence.

[4]http://www.jessrules.com/
[5]http://www.rfcode.com/

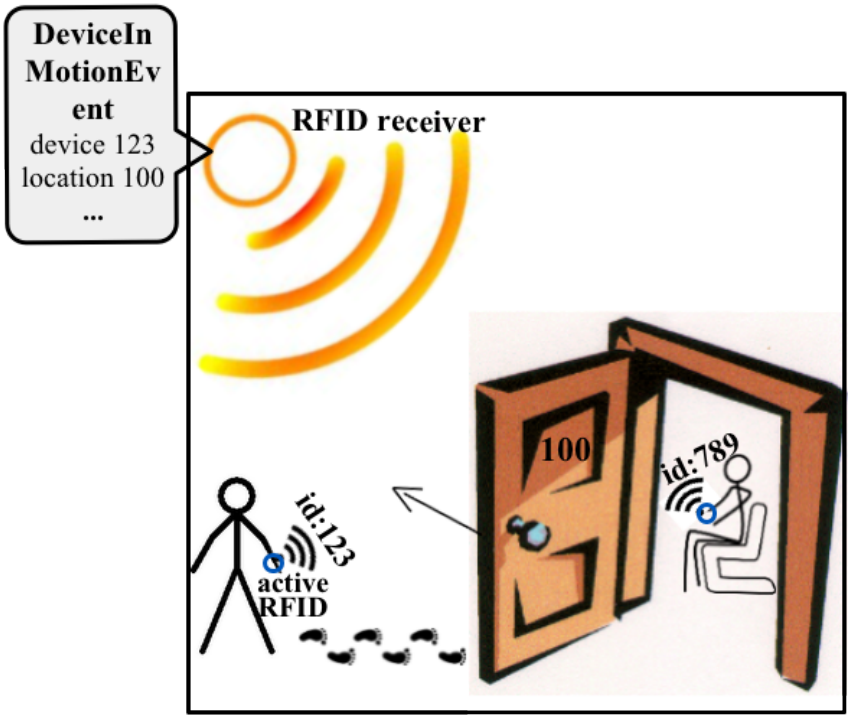


Figure 4.8: Active RFID sensing and event generation

With the rule-based approach to activity monitoring, one does not need to resolve/consider all possible combinations of events that happen concurrently. Let us consider a scenario where some patient undergoes a physiotherapy treatment. During this session, a patient can perform a variety of activities/events – they can be walking, sitting, performing some sport exercises, etc. The Physiotherapy rule does not need to be informed of any of these activities. The fact that the physiotherapy session is made up of these shorter events does not affect or complicate its logic in any way. It is natural that some events are made up of sub-events, however this does not entail that all sub-events of an event need to be discovered for this event to be inferred. In short, in some cases LLEs are needed to infer HLEs. On the other hand, not all LLEs need to be considered/discovered for a HLE to be inferred. Moreover, in the example above the physiotherapy session can be interrupted by another event e.g. a speech therapists walking into the room. The fact that some event overlaps in time with the physiotherapy session does not have to terminate this session. In the rule-based approach, the physiotherapy session would continue, unless one implements a rule acting differently on this type of scenario. Therefore, the rule-based approach to activity monitoring provides a flexible solution adaptable to many

scenarios. How various scenarios are handled depends on the specific requirements and field expert's opinion.

The reminder of this section looks into rules that are distributed across all system layers. These rules are not project-specific and apply to all AM scenarios. They form basis, i.e. produce facts, that are very likely to be used in the GEL layer of various AM projects. The two projects undertaken during the course of this PhD, that are described in Chapter 5, both make use of the events inferred by the rules described below. All rules in this section are represented with the aid of pseudocode and we assume that every fact/event name that the system deals with, uniquely matches name of the class in the *Event* and *RTL models*. Some rules are supplemented with example facts they produce (CLIPS format discussed earlier), in which the value of `atTime` slot is expressed in milliseconds as either one value (for snapshot events) or with two values (for events that spam some period of time). Moreover, the `atPlace` slot can be resolved against the mappings of location codes to their names, which can persist in the DB as another static facts feeding the system. The full Jess code of rules can be found in Appendix C.

4.3.1 Data Layer Rules

The Data Layer (DL) consists of a set of rules that apply to the 'live' data. All the rules in this layer are loaded into the KB before any facts get asserted. As the data is pulled from the low-level SNM and gets injected into the RBS's WM in the form of facts, these rules fire. Their main objective is to reduce the data volume where possible either by merging multiple facts of different type, examples for which can be `Rule_1a, Rule_1b, Rule_1c and Rule_1d`; or by fusing facts of the same type – `Rule_2a and Rule_2b`. Hence the name of this layer, as it is mostly concerned with injecting data into the WM and reducing its volume.

The RTLS equipment used in this project, as explained in Chapter 5, apart from location information also provides motion status for RFID tags. Therefore there are two types of facts representing RTLS' data: `DeviceInMotionEvent` and `DeviceNotInMotionEvent`. The first, as it is demonstrated by Rule_1a below, is merged with `Person` (and `Equipment` in Rule_1b) facts to assert `MovingObject` facts to the Working Memory (WM). Analogically, the same happens for `DeviceNotInMotionEvent`, which is translated to `StaticObject` facts – Rule_1c and

Rule_1d, which have been omitted due to their similarity to Rule_1a and Rule_1b. The distinction between `Equipment` and `Person` is made to allow for appropriate slots in the *Event model* to be filled in i.e. either `involved` – in case of things, or `involvedAgent` – for humans. Additionally, these rules hide the physical addresses of the RFID tags attached to objects and people, which helps to integrate data coming from various RTL systems. The asserted fact is also given a unique id by which it can be referenced. Due to the volume of data coming in the form of `Device(Not)InMotionEvents` it is feasible to drop these facts once they are no longer used. By doing so, the ability to track evidences is lost, however it is not needed in the DL layer, as these are the only sources of information and simply need to be trusted. Otherwise, if these evidences were to be kept, instead of reducing the volume of data, these rules would cause the opposite effect.

```
Rule_1a
IF
    DeviceInMotionEvent with some deviceAddress
AND Person with matching deviceAddress and some name
THEN
    generate unique ID
    assert MovingObject for that person and set its ID
    remove DeviceInMotionEvent
END

sample output fact:
 (MovingObject (atPlace "100")(involvedAgent "patient_2")(involved "nil")
 (evidencedBy )(atTime 1337078615758)(id "gen4578794"))
```

Rule_1a is a good example of a fusion rule, which based on sensor-derived data and static mappings asserts a certain conclusion to the WM. Another two rules that exist in the KB of the DL layer are: Rule_2a and Rule_2b. These rules, on the other hand, collapse similar 'snapshot' events into one longer event with some time interval – a good example of a data redundancy rule. Again, due to the analogy of the two rules only one is presented below. Rule_2a aggregates `MovingObjects` that happened at the same location and involved the same agents/objects. The two `MovingObjects` are only aggregated if the time of the second `MovingObject` is grater than the time (or finish time if the fact is already an aggregation of other facts) of the first `MovingObject`. Another test is made to check if the difference between the times of the two `MovingObjects` is not greater than 3 seconds, so that only traces that are no more than 3 seconds apart are merged. The RTL system's sampling rate is 2 seconds; therefore one extra second is added to ensure that the processing time does not influence the outcomes. This parameter is adjustable depending on the characteristics of a given RTL system used. If we imagine that a person or an object can remain in one place for hours, it is more feasible to record their entry and leaving time rather than storing huge volumes of data in 2 seconds intervals that is redundant.

```
Rule_2a
IF
    MovingObject
AND MovingObject at the same location with the same involved(Agent)
AND the latter MovingObject is more recent
AND the times of the two MovingObjects are no more than 3 seconds apart
THEN
    remove the second MovingObject
    IF
        the first MovingObject has 2 timestamps
    THEN
        update its finish time
    ELSE
        remove the first MovingObject
        generate unique ID
        assert a new MovingObject with that ID and 2 timestamps
        (first MovingObject's time and second MovingObject's time)
END

sample output facts:
 (MovingObject (atPlace "100")(involvedAgent "patient_2")(involved "nil")
 (evidencedBy )(atTime 1337078615758 1337080599868)(id "gen4578794"))

 (MovingObject (atPlace "100")(involvedAgent "staff_1")(involved "nil")
 (evidencedBy )(atTime 1337078439752 1337081567768)(id "gen4565477"))
```

In the conclusion of the Rule_2a the more recent `MovingObject` is deleted and then two cases are considered. If atTime property of the earlier `MovingObject` is a list – and hence has two time elements: from and to – its finish time is updated. In the opposite case, where both traces have a single element in the `atTime` slot, the first `MovingObject` is removed, id for the new fact generated and a new `MovingObject` is stored. The `atTime` attribute of the new fact now consists of two elements, which are the start and the finish times of this event. In all the rules presented so far (and the ones omitted due to their analogical nature), the value of the `evidencedBy` attribute of asserted events is left blank, as the evidences for these are removed in

order to reduce the memory consumption. By doing so, the ability to track supporting evidences is lost, however it is not mandatory in the DL, as these are the only sources of information and need to be trusted.

4.3.2 Sensor Event Layer Rules

The SEL layer consists of a set of rules, which run on completed events. Objectives of the rules in this layer are to correct erroneous sensors' data and to merge any facts that can be reduced. It is advisable that rules in this layer are written by somebody whom has in-depth knowledge of the physical sensors incorporated into the system, so that abnormal hardware behaviors or incomplete sensor readings can be dealt with appropriately. Therefore, if the RTL technology has any shortcoming resulting either from the environment in which the system is deployed, or its physical limitations e.g. range, the SEL layer would handle it appropriately.

Let's imagine a scenario where the RTL system is reporting its location in the following order: location_1 for 30sec, lack of location for 4sec, location_1 for 1min, lack of location for 6sec and location_1 for 1min. Human's analysis of the problem would be to assume that the person wearing the tag was at location_1 room for the entire time (2min 40sec) as it is not feasible to assume that they walked out of the room and came back within 4-6sec. Moreover, if the probability-of-failing-detection is small it needs to be assumed and accepted that the loss of location information is only due to the imperfectness of the RTL system and therefore by applying some common sense reasoning the data can be 'cleaned up' neatly. The probability of failing detection is different for every environment – it depends on the topology of the environment, and therefore needs to be calculated for every single case separately. It is simply the proportion of time the tag spends reporting lack of location or being outside of range, over the total time. It can be calculated for any portion of time, however to get a good picture of the deployment environment it is advisable to calculate it over a 24 hours period. The pseudocode for the so-called 'sandwich rule', which hides its filling (loss of location) in between its outer layers, is as follows:

```
Sandwich_Rule
IF
    MovingObject or StaticObject (O1) at some location other than blank location
AND Moving or StaticObject (O2) at blank location with the same
    involved(Agent) value as O1
AND MovingObject or StaticObject (O3) at the same location as O1 with the same
    involved/involvedAgent value as O1
AND O3 happens immediately after O2 and O2 is immediately after O1
AND ProbabilityOfFailingDetection is less than 0.1
THEN
    remove O2 and O3
    update O1 finish time with O3 finish time
    assert that Sandwich_Rule is an
    EvidenceFor the updated event
END
```

The above rule consolidates the events, which were fragmented due to the imperfectness of the sensing (RTL) system. It looks for three events of the same type, occurring one after another, out of which the middle one (O2) does not contain any location information and O1 and O3 have matching locations. If such a sequence of events exists and the *ProbabilityOfFailingDetection* is sufficiently low then events O2 and O3 are deleted and the finish time of O1 is amended. Having this rule at the *Moving/StaticObject* level ensures that the data is permanently corrected before any of the GEL rules fire. Therefore, none of the Goal Events should suffer from the presented fragmentation problem. However, this rule only deals with cases where a single blank location falls in between two events with matching location. What about cases where there are two or more Static/MovingObject events in between – as demonstrated by Fig.4.9. Moreover, due to the complexity of this rule, number of conditions that have to be met and the volume of *MovingObject* & *StaticObject* that usually persist in the WM, this rule was tested to work unacceptably slowly – needs several hours to compute all the possible combinations. Therefore, it can serve as an example of a badly performing rule and its structure should be avoided. An alternative solution has been created and is discussed below.

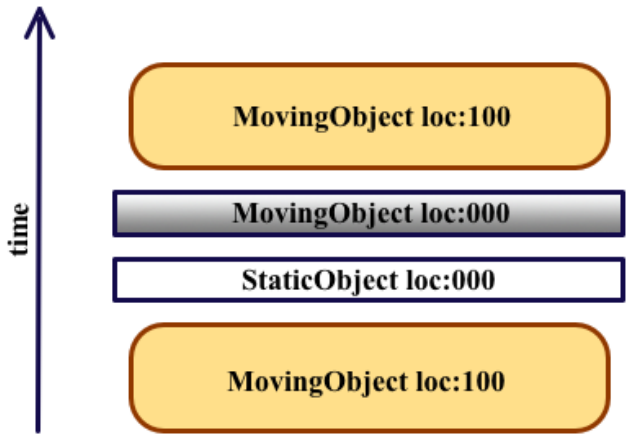


Figure 4.9: The “Sandwich" Problem

`Rule_3a-d` is a pseudocode for rules 3a to 3d. All these four rules could be implemented as one using the ‘or’ conditional element: (`MovingObject` or `StaticObject`). However, the rule complexity analysis and performance tests have shown that 4 rules with no ‘or’ conditions perform much faster than a single rule with two ‘or’ conditional elements. More on rules complexity analysis in Section 5.2.7.

```
Rule_3a-d
IF
    MovingObject or StaticObject (O1) at some location other than blank location
AND MovingObject or StaticObject (O2) at blank location with the same
    involved(Agent) value as O1
AND O2 happens immediately after O1
AND ProbabilityOfFailingDetection is less than 0.1
THEN
    assert LocationCorrectionTracker for that Person/Object,
    at O1's location, for O1's id at O2 finish time
END
```

In short, `Rule_3a-d` looks for two subsequent `MovingObject` or `StaticObject` events, for

the same entity, that occurred one after the other. The first event takes place at any location other than blank location. The second event takes place at the blank location. If the `ProbabilityOfFailingDetection` is sufficiently small then `LocationCorrectionTracker` event is asserted for that entity, at the non-blank location, for O1's id and O2's finish time. In this way, the `LocationCorrectionTracker` fact can be matched with another Static/MovingObject without a need to reference fact O1. The `Rule_3e-f` does exactly this.

```
Rule_3e-f
IF
    LocationCorrectionTracker (O1) at some location
AND MovingObject or StaticObject (O2) at blank location with the same
    involved(Agent) value as O1
AND O2 happens immediately after O1
THEN
    modify LocationCorrectionTracker by adding O2's id to its list of ids,
    and by updating its finish time.
END
```

The head of the above rule looks for `MovingObject` or `StaticObject` events that took place at the blank location, immediately after the time recorded for the `LocationCorrectionTracker` fact. In this process, the `LocationCorrectionTracker` gets updated with O2's id and finish time – i.e. its tail gets extended. Finally, the `Rule_3g` performs the location correction algorithm for all the events referenced by the `LocationCorrectionTracker`.

```
Rule_3g
IF
    LocationCorrectionTracker (O1) at some location
AND MovingObject or StaticObject (O2) at the same location
    as O1, with the same involved(Agent) value as O1
AND O2 happens immediately after O1
THEN
    foreach id in O1 modify that fact by replacing its
    atPlace slot with O1's location and add EvidenceFor
    fact for each id.
    remove O1
END
```

In the head of the rule above, `LocationCorrectionTracker` fact is matched with `MovingObject` or `StaticObject` that took place at the same location for the same entity, immediately after O1. If these conditions are satisfied, then for each of the ids stored by the `LocationCorrectionTracker`, the referenced fact is updated with the correct location and an `EvidenceFor` fact is asserted, reflecting that this rule has modified this fact.

Despite the fact that so many rules are required to perform a fairly simple task of correcting the missing location, the rules discussed above form a good alternative to the slowly performing `Sandwich_Rule`. Moreover, these rules are able to handle cases where multiple events with no location information form the sandwich's 'filling'. On the other hand, the original `Sandwich_Rule` was only capable of working with one 'filling' in between the outer layers. To sum up, the later approach performs faster and handles all the possible scenarios. The correct order of rules' execution is achieved through the use of *salience* – rule's priority. Rules with higher salience fire first and in the aforementioned case the salience should be set in such a way that the rules fire in the following order: `Rule_3a-d`, `Rule_3e-f` and `Rule_3g`. However, the correction of the atPlace value does introduce the fragmentation problem, as adjacent events of the same type, with matching slots are not collapsed. The solution to this problem has already been discussed for the DL layer and `Rule_2a` and `Rule_2b` needs to be loaded into the SEL layer in order to merge adjacent events of the same type.

To summarise, rules in the SEL layer serve the purpose of data: fusion, redundancy and error

correction. They ensure that the data quality is high and data volume low – the data recorded by the DL layer gets compressed even more in order to speed up the rules' execution time in the next layer. Rules in this layer should be written by somebody with an expertise in the RTL system deployed. Moreover, as in the example of the Sandwich_Rule, rules of thumb such as: closed-world assumption or the fact that no object can be observed in two places simultaneously, can often assist in the creation of clever rules.

4.3.3 Goal Event Layer Rules

The events produced by the SEL layer feed the Goal Event Layer (GEL), which reasons over completed events. Due to the fact that this work is scoped to activity monitoring scenarios, this layer can be already pre-populated with the three rules described in the reminder of this section. These rules are not specific to any particular domain, such as the rehabilitation, but are rather generic to the problem of AM and form a solid basis for the development of the domain-specific rules. Therefore, the system is not fixed and exposes one very important extension point at the GEL layer where experts can publish their rules – meets the *adaptability* criterion for SNM listed in Section 2.1.

The below transformation rule `Rule_4a` (and `Rule_4b` which deals with `MovingObjects`) looks for `StaticObject` events and transforms them into `Event` facts – process which hides away the heterogeneity of the `MovingObject` and `StaticObject` facts. This time the `evidencedBy` slot in the asserted `Event` is referencing the relevant source event, which can be looked up for more details (e.g. motion information) if needed. Similarly to SEL's rules, a unique `id` is generated and assigned to that `Event`. Worth noticing is the `EvidenceFor` fact, which associates rule's name with the event's `factID`.

```
Rule_4a
IF
    StaticObject
THEN
    generate unique ID.
    assert an Event with that ID and all the attributes of the StaticObject
    assert that Rule_4a is an EvidenceFor the Event created
END

sample output facts:
 (Event (atPlace "100")(involvedAgent "patient_2")(involved "nil")
 (evidencedBy "gen4578794")(atTime 1337078615758 1337080599868)
 (id "gcrgen35534"))

 (Event (atPlace "100")(involvedAgent "staff_1")(involved "nil")
 (evidencedBy "gen4565477")(atTime 1337078439752 1337081567768))
 (id "gcrgen27757")
```

The `Rule_4c` presented below takes the outputs of the aforementioned `Rule_4a\b` and merges Events, which took place one after another, at the same location, for the same entity. The evidencing events E1 and E2 are kept whereas a new Event with the `atTime` value spanning the time of these two events is asserted in the conclusion of this rule. Therefore, if an object is recorded at the same location via the use of `MovingObject` and `StaticObject` events, the motion information can be hidden in order to produce one, long Event for the entire duration spent at this location. Some goal rules will use only the location information and hence collapsing multiple events that took place at the same location is sensible. On the other hand, other goal rules may need the motion information and hence none of the evidences gets removed from the WM.

```
Rule_4c
IF
    Event E1 at some location, for some entity
AND Event E2 at the same location, for the same entity
AND E2 is immediately after E1
THEN
    generate unique ID.
    assert an Event with that ID and all the attributes of E1 and E2,
    and atTime value being the start time of E1 and finish time of E2.
    assert that Rule_4c is an EvidenceFor the Event created
END

sample output facts:
 (Event (atPlace "100")(involvedAgent "patient_2")(involved "nil")
 (evidencedBy "gen4578794")(atTime 1337078615758 1337080599868))
 (id "gcrgen35534")

 (Event (atPlace "100")(involvedAgent "staff_1")(involved "nil")
 (evidencedBy "gen4565477")(atTime 1337078439752 1337081567768))
 (id "gcrgen27757")
```

The last generic goal rule `Colocated_Rule` aggregates events based on their spatial and temporal co-existence. This is a good example of a rule that is triggered by two HLEs to conclude more complex HLE. Obviously the evidencing HLEs are recorded through the `evidencedBy` property, which is common for all the RTL classes represented by Fig.4.3. The head of this rule searches for two `Events` representing two different entities, that took place at the same location and overlap in time. The implementation of this rule calls the `overlaps` function which has been implemented to check if the two time periods overlap. Some rule's conditions are non-trivial and this is where functions become useful. Upon matching all conditions, the `Colocated_Rule` generates unique id for the inferred `Colocated` event, which has its slots filled with the union of the individual Events' slots. Therefore, `atPlace` slot has one value in it as both Events took place at the same location but the `involvedAgent` slot has two values as the `Colocated` event is inferred for two entities.

```
Colocated_Rule
IF
    There are two Events for different entities with different IDs
AND Both Events occurred at the same location
AND Their timespans overlap
THEN
   generate unique ID
   assert a Colocated event for that location, involving the two entities,
   for the overlapping timespan, with the generated ID
   assert that Colocated_Rule is an EvidenceFor the Colocated event created
END

sample output fact:
 (Colocated (atPlace "100")(involvedAgent "patient_2" "staff_1")
 (involved "nil" "nil")(evidencedBy "gcrgen35534" "gcrgen27757")
 (atTime 1337078615758 1337080599868)(id gen35545)))
```

The Goal Events Layer also consists of domain specific rules that can be developed in co-operation with experts from the given field. These rules are responsible for generating the HLE of interest. In the rehabilitation context, the physiotherapist may want to know, for example, how much physiotherapy time each patient gets in the course of the week. Having already defined rules for composing `Colocated` events, the physiotherapy rule can fire whenever a patient and a physiotherapist are recorded in the physiotherapy room together for some period of time. Examples of GEL rules that are domain-specific can be found in the two case studies described in Chapter 5.

The process of chaining events e.g. '`ObjectInMotion -> MovingObject -> Event -> Colocated -> ...`' happens across different layers of the system – Fig.4.10. With the growing events complexity grows their semantic enrichment. The key particle in this chain is the `Event` fact. The fact that events of different types are expressed as an `Event` fact facilitates the use of generic rules – only one rule needs to be written to consider all relevant sensor data. Therefore, whenever a new sensor is introduced to the system, the KB remains unchanged while already making use of the new sensor data. Moreover, this process of event chaining has another important advantage: facts' re-use. For example, in order to infer `Physiotherapy`

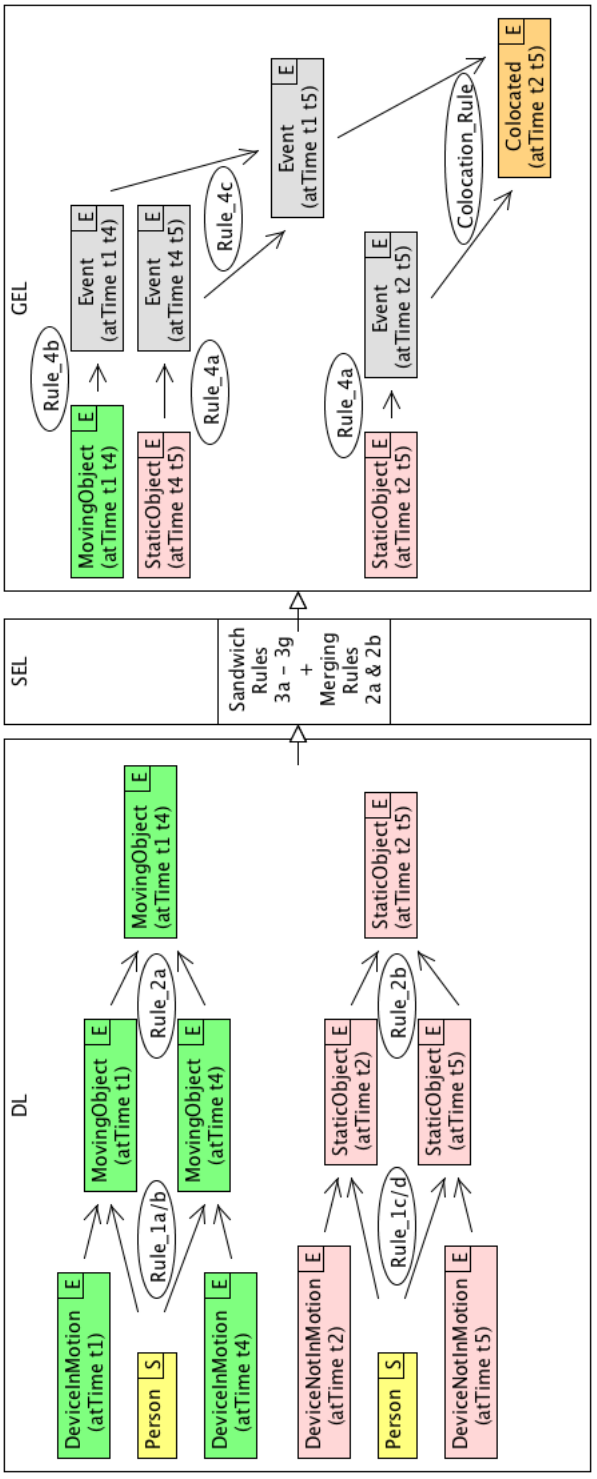


Figure 4.10: Event Chaining Across Layers

events, only one new rule needs to be provided which makes use of the `Colocated` events and other facts. Therefore, the chain of events presented in Fig.4.10 does not need to be formed again but, instead, can be re-used to infer `Physiotherapy` events.

4.4 Persistence

A reliable storage component is a desirable feature of any system as in many cases lost data is very hard and often even impossible to recreate. One of the key requirements in the implementation of this architecture was to use available, free of charge components. Hence, the implementation of a separate storage layer that does not make use of the available technologies would violate this principle e.g. in-file storage. Therefore, various storage mechanisms have been considered against these criteria. Cloud storage e.g. ownCloud[6] was not taken into account due to two reasons. Firstly, it relies on the availability and bandwidth of the Internet connection. Secondly, one might not want to store their sensitive information on the cloud. NoSQL approaches, even though very efficient for some problems, often lack standard interfaces and lack the full ACID transaction support. Therefore, the choice have been made to use MySQL database (DB), as it provides reliable storage. Other benefits of this open source product are: scalability, flexibility, high performance, strong data protection and data integrity[7].

MySQL DB in the proposed architecture serves two purposes. Firstly, it stores static mappings which feed the Data Layer of the system. Secondly, it permanently stores events produced by the platform, effectively freeing up the working memory.

4.5 Advantages of Layered Architecture

For every system its design choices have to be justifiable. These choices are driven by their anticipated benefits, however for any idea to be objectively assessed it needs to be implemented so that it can produce some tangible evidence. This evidence and the implementation of the

[6] http://owncloud.org/

[7] http://www.mysql.com/why-mysql/topreasons.html

platform is presented in Chapter 5. The anticipated benefits of the layered architecture are listed below.

Firstly, system layers (DL, SEL and GEL) can be distributed over multiple physical tiers in order to overcome physical limitations of a single computer (memory and processing power). Distribution of layers amongst multiple machines, has many potential benefits which have not been explored in this project: performance improvement, increased scalability and fault tolerance [62].

Secondly, layered approach to any system lowers the coupling of its components. In the RBS2P, the coupling of its layers is so loose that they can be separated across multiple computers – as described in the paragraph above and as demonstrated on the example of the RSU project in Chapter 5. The low coupling of the system entails more benefits, such as:

- the easiness of adding new functionality to the system,
- improved orientation in the code and its testability,
- easier implementation, maintenance and deployment.

Hence, if this platform was to serve data in many formats only one, easily identifiable layer would need to be extended without any concerns about breaking the other parts of the system. This change would be easy to test and easy to deploy to a working system. Moreover, adding security mechanisms to the platform would probably involve a modification of each layer or adding an extra layer on the top – both approaches would be easier to handle in layered architecture.

4.6 Disadvantages of Layered Architecture

Obviously, every design choice carries both pros and cons. Having named all the advantages of the layered architecture one has to consider its possible shortcomings. The following list names the possible negative effects of a layered architecture [4] on the RBS2P:

Negative impact on performance. There is an extra overhead of passing through layers instead of calling a component/getting the data directly. However, this does not apply to the two tests cases discussed in this thesis. The data collection environments are disconnected from the Internet and hence it is necessary to write the data to a file (CSV format in current implementation), so that it can be read-in by the GEL layer on another machine.

The use of layers adds complexity to simple applications. Most likely, if a simple application was to be 'built-for-purpose' to solve a particular problem it would have taken less time to develop it rather than to implement this layered platform and to interface the application with it. Having this platform already in place, could increase the complexity of a simple application but it would definitely make its development quicker and easier. Overall, this platform has not been 'built-for-purpose' but rather to ease the application development in the domain of activity monitoring.

Changes to lower level interfaces may percolate to higher levels. This statement applies to the core concepts of the system. If, for example, the *Event model* was to be changed in one of the layers, then this would surely affect all the other layers. The system would still execute but to ensure its working correctly, other layers would have had to be tested and updated accordingly. Therefore, this problem only applies to the core concepts that are implemented by the system (models that are used to interface between the layers) and only if these get modified it may impose an avalanche effect on the other layers. However, the RBS2P is coupled so loosely that this danger is minimal.

Development of user-intensive applications can take longer if the layering prevents access to some relevant data/routines. The RBS2P exposes all the relevant data to the user and clearly identifies its access points.

This chapter discussed the motivation for designing and implementing the RBS2P. It explained modeling choices made and introduced layered *Event models*, which form platform's foundations. Examples of LLEs and HLEs were shown and the use of the *Event model* in the system layers was presented. Moreover, rules belonging to the empty system were discussed with the

aid of pseudocode, as they form an integral part of the solution to the problem of activity monitoring. Finally, this chapter concluded with the discussion of advantages and disadvantages of layered architecture. Having introduced the platform, the next chapter presents the reader with the two pilot studies in which RBS2P has been used.

Chapter 5

Pilot Studies

This chapter provides the reader with an insight into the RTL technology used in two pilot studies. For the first study domain knowledge and rehabilitation rules are introduced. Furthermore, reliability and validity (further divided into script-based validation and observation-based validation) studies are described in detail with respect to their: requirements, selected approaches, results and conclusions. This chapter concludes with an insight into the Home Monitoring second pilot study.

5.1 Platform Evaluation

It is desirable for any software system/platform to be evaluated in order to determine the quality of the solution and judge on the extent to which it solves the given problem. One can evaluate software platform in a number of ways. Running a feasibility study answers a very important question – is it possible? Knowing that the chosen solution is feasible, one may want to run a pilot study to prove that it is appropriate/works for the given problem. If two (or more) techniques are to be compared, a comparative study helps to judge which technique is more suitable. Another useful evaluation method is the observational method. Through collection of data derived from direct observations, one is able to compare these against software recordings. Therefore, the software system can be confronted against the ground truth. Expressing a problem in a formal model also helps to evaluate software. One is then able to judge whether the mathematical abstraction of a certain problem is implementable in software and produces a sound solution. Finally, simulation can evaluate many important characteristics of a software system – scalability, QoS, response time, etc. Even though simulation is cheaper, easier to

apply, and quickly produces tangible evidences, it only tests the scenarios accounted for by the programmer. In cases where testing software in a real environment is difficult or impossible (such as testing Curiosity rover on Mars), it is however a useful technique.

A number of evaluation approaches have been used in this research. Initially, a short feasibility study took place before the two pilot studies presented in this thesis were conducted. The aim of the feasibility study was to judge whether the proposed platform architecture can be implemented and produce desirable results in a timely fashion. Subsequently, the platform has been evaluated in the two pilot studies to prove its applicability to the real-world problem. In the first pilot study (Regional Stroke Unit) reliability and validity studies have been conducted. In both these studies, observational study method has been used to create a ground truth record against which system recorded data was confronted. Reliability study helped to assess whether the platform was capable of producing consistent results. On the other hand, validity study helped to judge on the strength of the platform's outcomes, or in other words, the extent to which the system measured what it was said/designed to measure. Having satisfied the reliability and validity studies the project progressed into the longitudinal study. However, the results of this study are not reported on in this thesis, as they belong to our collaborators from the School of Healthcare Sciences, Cardiff University.

Solving a real problem has been chosen over running a simulation study. The two pilot studies revealed many problems (staff conformance issues, physical sensor failures, issues of going outside of RTLS' range, 'sandwich' problem), which even the best-designed simulation would have troubles accounting for. Sensors are deployed in various environments to solve real problems and hence need to be field-tested to prove their usefulness.

5.2 Regional Stroke Unit Pilot Study

The Regional Stroke Unit (RSU) pilot study was carried out in collaboration with the School of Healthcare Sciences, Cardiff University and its aim was to test unobtrusive RTL technology in hospital setting. Researchers and doctors would like to know how active stroke survivors are throughout their stay in the unit. Devices such as mobile phones or video cameras are not allowed on ward, and hence unobtrusive RTL technology was chosen for rehabilitation monit-

oring. Small size of sensor tags and acceptable range of operating frequency made piloting of this technology possible.

5.2.1 Real-Time Location Systems and RFID Technologies in Hospitals

Real-time location systems (RTLS) have found their way into hospitals to solve a wide variety of problems. In [21] Fisher and Monahan evaluated existing RTLS's in hospital context. Leading RTLS technologies used in the reviewed projects were: RFID, active RFID, ZigBee, ultrasound, infrared and ultra-wideband. These technologies were applied for the following purposes:

- patient ID in surgery,
- asset tracking,
- patient tracking,
- personnel tracking,
- temperature monitoring,
- patient ID for delivering medicine,
- patient ID in emergency department.

Fuhrer and Guinard in [24] review projects using RFID technologies in hospitals. All of the considered solutions incorporate RFID technologies to solve the following problems:

- patient identification,
- blood tracking,
- smart operating theatres,
- tracking equipment, patients, staff and documents,

- avoiding theft of medical equipment,
- anti-counterfeiting,
- smart medicine cabinets.

The list of possible applications is quite comprehensive, which only shows the need for RFID technology in modern hospitals. Many of these problems relate to the prevention of human error and patient/asset/personnel tracking. Projects tackling tracking of equipment, patients, staff and documents (listed in [24]) did not consider discovering interactions between these entities. Also projects listed in [21] did not aim to discover activities of patients inside the hospital or to automatically create patients' physical recovery records e.g. in the context of rehabilitation. The volume of publications in this domain shows that there is a huge amount of interest in all the aforementioned applications of RTLS's and with fast-evolving technology, new projects come to live frequently.

5.2.2 Hardware Setup & Characteristics

Hardware used in this study is made up of the off-the-shelf RTLS (sensors and receivers) and the computational unit (PC) hosting the RBS2P and RTLS software. The active RFID sensors were acquired from the US-based RF Code company[1] – offering commercial solutions mainly for asset management, environmental & power monitoring. A typical RTLS offered by this company is made up of two hardware and one software component:

RFID Active Tags – report their status in certain time intervals. Models differ in size, visual qualities and the type of features they measure. Active RFID is the type of technology in which tags report their signals via radio waves over the air, and hence no direct contact is required between the tags and the reader (passive RFID). The payload gets sent over the 433MHz band, which falls within the industrial, scientific and medical (ISM) radio band.

[1] http://www.rfcode.com/

Figure 5.1: Active IR-enabled RFID Tags

Reader – "is a dual-channel radio receiver tuned to 433.92 MHz. The readers are programmed, calibrated and dedicated to interpreting and reporting the radio frequency messages emitted by RF Code tags. Tag transmissions are processed in real-time to quickly locate and identify tagged assets or personnel in defined areas. New models are compatible with wired and wireless networks for rapid integration into an organisation's IT infrastructure"[1] .

Figure 5.2: RFID Reader

Software: Zone Manager – "a real-time location engine designed specifically for use with RF Code's asset tracking and wire-free environmental monitoring solutions. Zone Manager is designed to be integrated with one or more business applications via its open

[1] http://www.rfcode.com/

application programming interface (API). Zone Manager handles all of the direct hardware interfaces for RF Code readers and tags"[1].

Room Locator – transmits an IR pulse pattern containing a unique 3- or 4-digit location code to enable room-level accuracy localisation for IR-enabled tags.

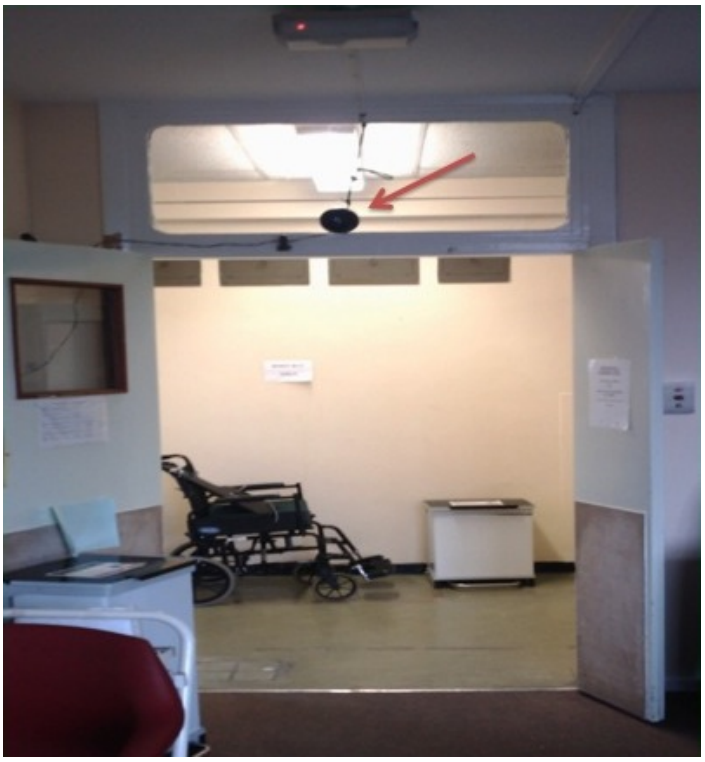


Figure 5.3: Room Locator in the RSU

Typically, the reader is equipped with two helical antennas (range up to 45m) but its range can be extended via the use of omni-angle or YAGI antennas. In the pilot study the M200 Fixed Reader was used with two omni-angle antennas, which provide the range of up to 91 meters. The antennas were connected to the reader using specially manufactured cables (screened) in order to reduce the radio noise. Antennas were mounted to the ceiling on both ends of the RSU's corridor to provide coverage for the entire unit's floor. The M160-i wristband tags were chosen for the study as they are of small size and provide motion status, IR capabilities (needs IR transmitter called the room locator) and the highest transmission range (up to 91m) out of all available tags.

[1] http://www.rfcode.com/

During lab and field testing the deficiencies of the chosen RTL technology were revealed. Presented in this chapter reliability and validity studies contributed to finding the limitations of the selected sensors. None of the discovered shortcomings were mentioned in the technical specification of the RTL system, hence came rather by surprise. However, with the help of the RBS2P some of these deficiencies were overcome by applying rules to the DL and SEL layers.

Retscher and Fu [54] [23] explored the problem of indoor navigation with RFID. There are two types of setup to tackle this problem. One is to place active RFID tags at known locations in the surrounding environment and equipping the user with a portable RFID reader. Alternative solution is to place RFID readers at fixed locations and tagging objects with active RFID tags. Authors of [54] [23] used the first approach as they argue it is cheaper (active RFID tags are much cheaper than RFID readers). Therefore, they developed and investigated three approaches to the problem :

Cell-based positioning: The location of the reader is described by a cell identified by the maximum read range to a tag – Cell-of-Origin (CoO) approach.

Trilateration using ranges to the surrounding RFID tags deduced from received SSI: Relies on the received SSI (Signal Strength Indicator) to distance conversion model. User's location is calculated based on the SSI received by the portable RFID reader for RFID tags present in a room. Trilateration is a similar concept to triangulation. To triangulate the signal one needs to measure tag's/object's signal strength at two different points in a 2D space in order to calculate the object's position within this space. Trilateration, on the other hand, involves three receivers placed at different positions and based on their received SSI calculates object's position in the 3D plane.

RFID location fingerprinting: Made up of two phases: off-line and on-line positioning. Off-line phase creates a database of received SSI values to the surrounding RFID tags at known locations - done by site survey. In the on-line phase nearest neighbour in signal space (NNSS) algorithm can be employed to estimate RFID reader's location.

The authors' conclusions on these approaches were as follows. Cell-based positioning method is only well suited for application where accuracy is not very important. Higher positioning accuracies are achievable with the other two methods, however these approaches need to go

through the calibration phase. As a solution they came up with a technique called RFID time-based CoO, in which "...the estimated trajectory of the user is corrected using the RFID time-based CoO position solution whenever the RFID reader detects the signal strength above a certain threshold from the nearest RFID tag in the surroundings. If several peaks in signal strength from one tag above the certain threshold are detected very closely to each other, their mean time value will be selected to obtain the estimated time point for updating the INS (Inertial Navigation System)"[54]. The accuracy achieved with this approach was on average about 0.8m with a maximum deviation from the ground truth of around 1.7m.

Due to the nature of the target environment (hospital) and its characteristics, cell-based approach has been ruled out at the start. Firstly it was not feasible to ask patients to wear portable RFID reader due to their size. Moreover, such an approach would contradict the rule of seamless technology. On the other hand, cell-based approach could be accomplished using the opposite setup – user wearing an RFID tag and rooms equipped with RFID readers. However, big range of RFID readers (up to 100m) would not provide the required granularity and would result, at most, in a room level localisation, which can be accomplished via use of room locators (explained later on in this section). Having ruled out the possibility of a patient wearing a portable RFID reader, the other two approaches were taken into consideration assuming patients wearing tags only. Tests have been designed and conducted in such a way that the recorded data would either serve as basis for defining a received SSI to distance propagation/conversion model (trilateration approach), or it could be used to to create a database of SSI values in the off-line phase of RFID location fingerprinting. Tests were carried out at the School of Computer Science & Informatics (Cardiff University)

The scenario was as follows. One tag (labeled Tag1) was placed on a chair and its signal strength, available via the Zone Manager's API, was recorded. The chair was moved around the room (9 x 7 meters) in 1 (vertically) and 2 (horizontally) meters intervals and the experiment was repeated 4 times on 4 different days. At each chair's position, tag signal was recorded for 1 minute. The four antennas (1A, 1B, 2A and 2B) used in this experiment were positioned in each corner of the room underneath the ceiling – as represented by Fig. 5.4. Moreover, two reference tags were placed in the room to test if their SSI would vary over time and if so, whether this variation could be used to correct Tag1 readings.

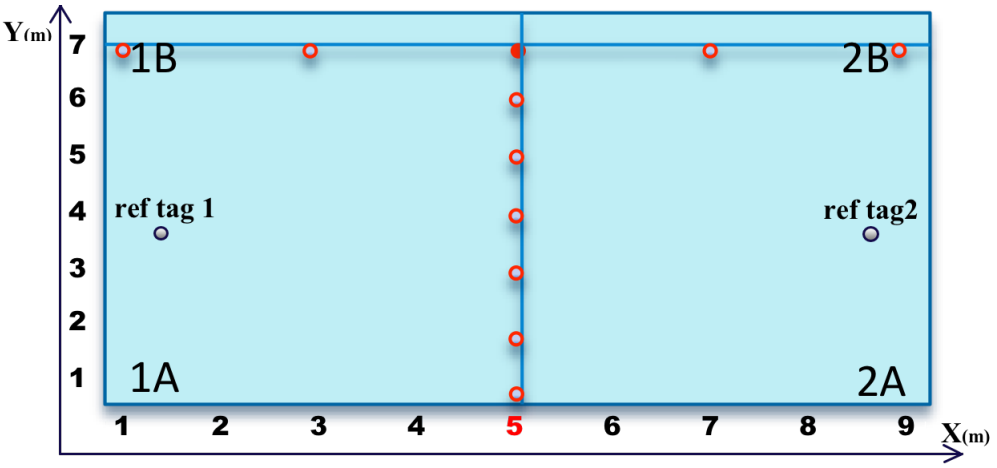


Figure 5.4: SSI change over distance reliability test

Fig. 5.5 visualises only a proportion of the results. The X-axis represents the distance, while on the Y-axis are the SSI readings of the antennas. The values displayed for Tag1 and for reference tags are averages over the period of recording (1min for every position). As the figure shows, the reference tags' signal was fairly static, while Tag1's SSI varied over the distance with no pattern. The final conclusions were rather disappointing:

- Tag signal strength (measured in dBs) did not drop with distance but varied without any pattern.
- Test results show that distance -> SSI is a non-monotonic function.
- SSI readings for all tags were inconsistent in relation to themselves (SSI varied when tags were stationary).
- Changes/fluctuations in the reference tags' signals did not correlate with SSI changes for Tag1.
- The same tests performed at different times, with the same setup gave different results.

To find explanation for these conclusions, two factors have to be considered: the environment in which the tests were carried out and the equipment used. The environment could contribute to the problem in the form of interference. Because the RFID equipment operates at the free-of-charge ISM band, interference with other equipment operating at similar frequency is likely.

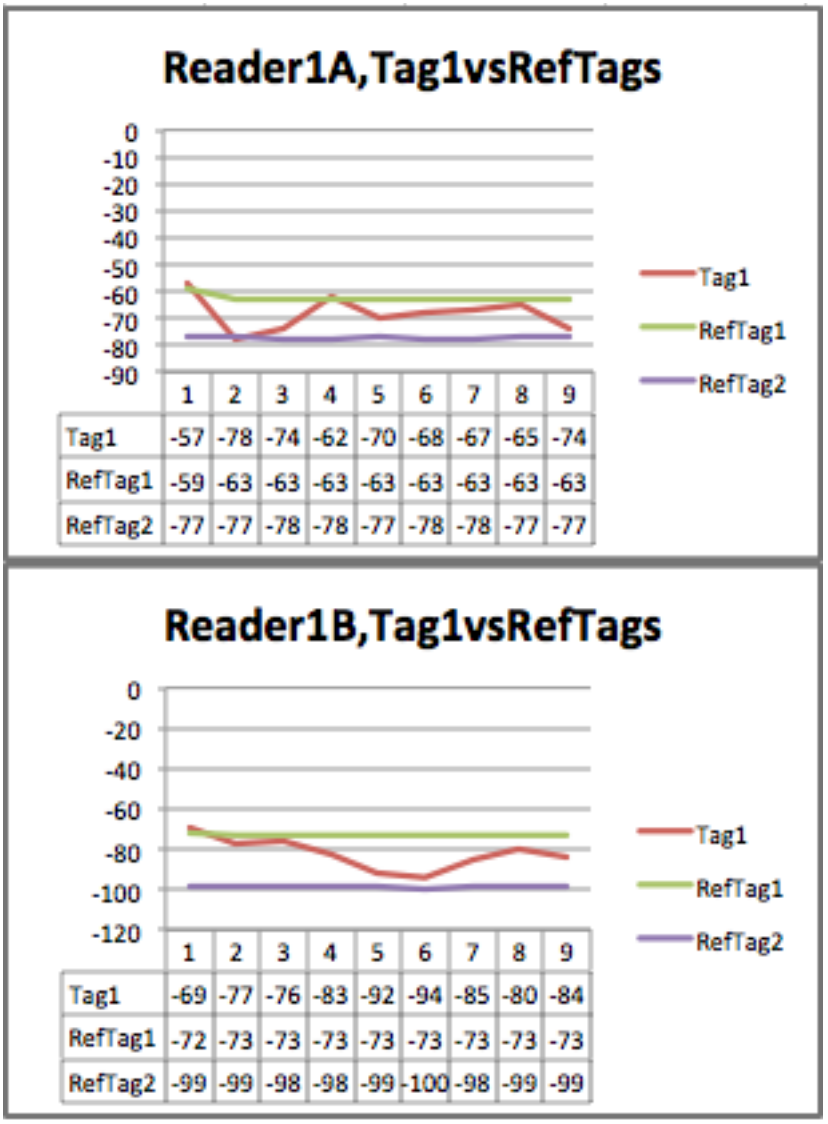


Figure 5.5: SSI change over distance for Tag1

Moreover, this Ultra High Frequency (UHF) band suffers from the line-of-sight propagation, therefore physical obstacles, which were not present in the experiment setup but will be in the clinical setting, would make the SSI propagation even more unpredictable. Authors of [54] also identified that high variability in received SSI is dues to obstacles and moving objects in the environment. Finally, the antennas cable loss (around 2.5dB every 10 meters) also contributes to the problem, however due to the fixed cable length it would not bias the results. Further investigation of the tag's architecture revealed that tag's signal distribution is non-uniformed (doughnut's shape) and hence its orientation in space affects the SSI readings. Due to all the

aforementioned problems it appeared to be impossible to calculate in-room SSI propagation model and hence the 3D trilateration idea had to be abandoned. For similar reasons, the RFID location fingerprinting approach also had to be abandoned. The consequences of this meant that tracking of objects with the in-room level accuracy was impossible and consequently closed the door to the inference of in-room activities which do not involve interactions with tagged objects. Therefore, the focus shifted from precise recognition of a tracked object's actions e.g. tooth brushing, into activity recognition at the stroke unit level e.g. recognition of physiotherapy sessions, which take place at a particular location.

The room-level localisation of active RFID tags is achieved via use of the room locators (RLs), which broadcast a pre-set location code via an Infrared (IR) signal (Figure 5.6). The IR-enabled active RFID wristband tags report their IR location code (if present, otherwise report location '000') to the RFID reader. For this a direct line-of-sight between the tag and the room locator is required, however our experiments revealed that the IR signal is reflected by many objects in the environment and therefore a direct line-of-sight is not mandatory.

5.2.3 Data Storage

The system's storage is made up of multiple tables. The *Events* table permanently stores events inferred by the system. Observables i.e. static mappings, in the RSU setting are stored in the following tables:

patients_store - Stores patients' psudo-names to device address mappings. In other words, this tables keeps track of wearable sensors assigned to patients.

staff_store - Has the same function as the patients_store but deals with staff and their roles.

equipment_store - Equivalent to the previous two stores but for rehabilitation equipment used on the unit i.e. stores their type and association with patients (if such exists).

room_store - Maps room location codes to the room names.

5.2.4 Motivation & Requirements

The first of the two pilot studies was conducted in the domain of post-stroke rehabilitation in the Regional Stroke Unit (RSU) in Cardiff. This unit, "part of Cardiff and Vale University Local Health Board, is an integrated rehabilitation service provided by a team of multi-skilled health care and allied professionals that offer effective, modern and coordinated care. It is a unit dedicated solely for the rehabilitation of adults recovering from a stroke"[1]. The motivation for this study was to prove the usefulness and applicability of the RBS2P to a real-life problem, while testing its reliability and validity. One of the leading concepts behind the implementation of the platform was to enable experts from the application field to collaborate in the process of rules implementation and fine tuning – so that their knowledge can be used to specify the logic for information inference. A collaboration has been established with the School of Healthcare Sciences Cardiff University and the UK National Health Service (NHS) in order to carry out the project. The study started in February 2012 and finished in May 2013 and was funded by NISCHR[2]. The purpose of the project, presented to all the collaborators was as follows:

> The aim of the project is to provide information on patients' activity, location and movement speeds within controlled environment equipped with RFID technology. In this environment patients, recovering after a serious medical condition (e.g. stroke), are monitored without human assistance in order to assess their fitness/mobility.

And the research hypothesis for this project was:

> Patients' conditions can be efficiently and effectively evaluated by healthcare professionals (HCP), who have access to their mobility data. HCPs and physiotherapists in particular, can monitor their patients and compare their mobility against existing norms to decide whether their physical condition is satisfactory to discharge them from the hospital.

Due to the 1-to-many relation between medical staff and patients, the 24/7 human observation of all patients is impossible. It is also not feasible from the financial point of view. Nurses,

[1] http://www.rsucardiff.org.uk/

[2] http://www.wales.nhs.uk/sites3/page.cfm?orgid=952&pid=51982

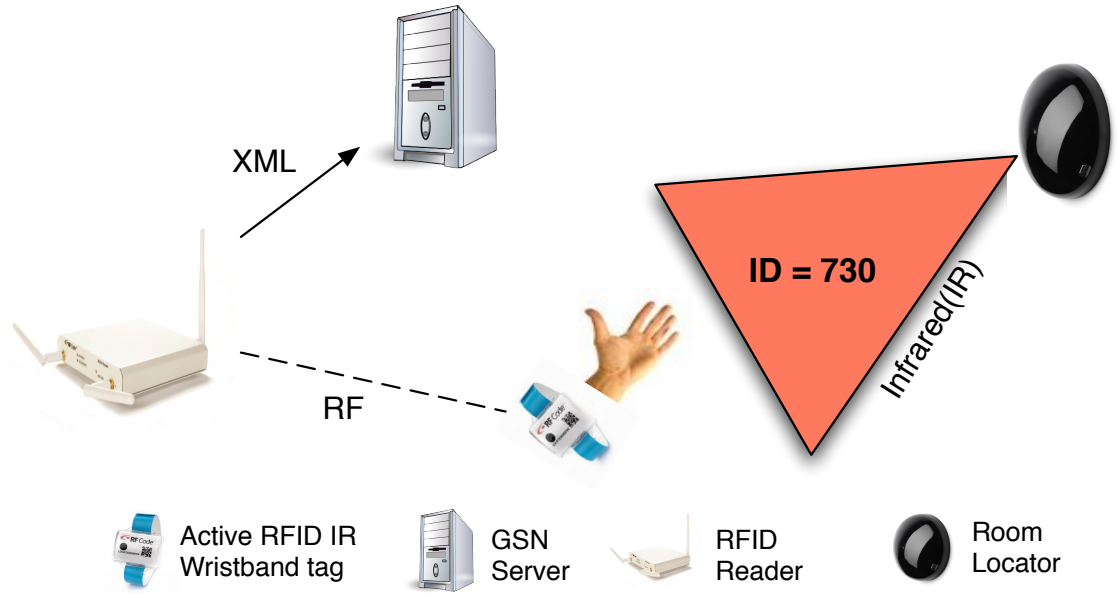


Figure 5.6: Localisation with the Room Locator (example location 730)

doctors and other HCPs have many everyday duties which they have to fulfill and patients' physical assessment takes place only occasionally – which provides just a snapshot of the state of a patients' mobility. Knowledge of this state is very important to healthcare professionals, so that they can provide the right treatment leading to faster recovery. Therefore there is a big gap in patient records that needs filling, so that healthcare specialists can be fully aware of where patients are on their 'road to recovery'. A solution to this problem, which enables NHS to provide better care to the stroke patients, is to implement a 24/7, unobtrusive, patient activity monitoring system within the RSU.

For this study RFID technology was chosen due to its unobtrusive characteristic and many interesting features. Other technology, such as cameras or mobile phones, were not allowed on the unit due to privacy and interference considerations respectively. The RFCODE[3] solution was selected, as an off-the-shelf product, which offers wide variety of components and operates at allowed frequency – as already described in the previous section. Fig.5.6 shows components of the system and the concept of localisation with room locators.

RFID active tags are of small size (slightly bigger than a watch) and hence can be worn on a wristband, allowing monitored subjects to forget about their existence. Humans behave dif-

[3]http://www.rfcode.com/

ferently when they are aware of being observed, so it was crucial for this study to seamlessly integrate technology into the unit's environment.

The following list represents the project requirements agreed on at its start:

1. Detect patient (wearing RFID tag) in a room equipped with RFID reader and with a room locator.
2. Detect patient's proximity and/or synchronised movement to other patients/doctors.
3. Detect the fact that the patient is using some (walking) aid.
4. Adjust/filter collected data to account for factors such as:
    - Radio noise,
    - Signal strength loss due to obstacles,
    - Sudden loss of signal broadcast by room locator (due to line-of-sight limitation of this technology).
5. Detecting events of the following kinds:
    - A given patient is in room X.
    - A given patient has walked from one room to another (the system can provide information on the distance travelled, and average walking speed and whether any walking aid is used).
    - During the course of the day, a given patient has spent X hours lying in their bed, Y hours moving within the ward, and has seen a specific healthcare professional at some point.
6. Permanent storage of collected information, which minimises the effects of power shortages and accidental switch offs.
7. Provide statistical data about some individual, such as: weekly/monthly totals of distance walked, time spent in bed, number of interactions with certain healthcare professionals.

The main assumption for all the studies presented in this chapter is that the system is capable of capturing the RTLS data and permanently storing it for subsequent analysis. The environment in which the data collection took place, did not allow for outgoing Internet connections, hence only the DL layer needed to be loaded onto the on-site machine. This layer is responsible for data redundancy and low-level data aggregation, so that the pre-processed data can be reliably stored in the MySQL table for further analysis. The data could be taken out of the collection environment by the means of portable storage, as all the data which persists in the DB can be effortlessly exported to CSV files.

5.2.5 Regional Stroke Unit Environment

The RSU and any other clinical environment are usually governed by a set of **operating protocols** and restrictions, which can be incorporated into the RBS2 system in the form of rules. For example, the information that visiting hours are between 4-6pm allows to disregard independent activities performed by patients, on the basis that visitors could assist patients and they are not tagged. The unit has its own schedule, which includes: visiting hours, protected meal times (publicly available information) and treatment sessions (individual for every patient). In a 'closed world assumption' what is not currently known to be true is false – which in the sensor world means that the complete knowledge about the environment is assumed and negation as failure applies. In the RSU environment, where 'not quite closed world assumption' holds (patients, HCPs and equipment are tagged but external visitors are not), schedule information is very valuable as it allows to infer a possible involvement of an un-tagged party. Moreover, negation as failure technique can be used in the development of rules for time periods where no un-tagged parties are present. The unit also introduces some technical limitations: connecting to the Internet via hospital's network is banned and the use of cameras is not allowed. These restrictions have two immediate consequences: RBS2 needs to be disconnected from the Internet and can only be accessed locally; and validating the system against camera readings is not acceptable.

Apart from RSU's policies regarding the use of various equipment, another area of concern was participation of the units staff in this study. After several meetings explaining the technology and the research study, staffs' major concern was that the system was designed to track their

activities. Some individuals refused to wear tags, worrying that the management could use RTLS data for performance management reasons. To encourage them to participate, every profession of interest (nurses, doctors, physiotherapists, etc.) had their own 'bag' of tags, from which every day they would randomly chose a tag to wear. Eventually, over time, the staff participated fully in the study but there were occasions where tags were not worn. Our collaborator (School of Healthcare Sciences, Cardiff University) Arshi Iqbal was overlooking the stroke unit for compliance with set protocols.

Before conducting the reliability and validity studies various tests were carried out in order to test the RTLS' coverage. Firstly, tags were placed at all possible locations within the unit in order to determine system's coverage. As a result, the positioning of antennas within the unit had to be changed, which resulted in unit-wide coverage. Secondly, the coverage of room locators was tested in order to find out if the IR signals propagated to the corridor or overlapped with the neighbouring locations. Based on these tests an RL coverage map was constructed for future reference.

The platform has been deployed 24/7 in the Regional Stroke Unit and continuously tested with over 170 RFID tags over the course of one year. Every 2 minutes 'expired' events were automatically dumped to the DB to free up the working memory and to permanently store collected data. The implications of this approach are that in the event of power shortage, accidental switch off (happened once), or any other system failure, only two minutes worth of data would be lost. Moreover, the system also offers the CSV export function, which exported all the events from the Event table to a CSV file on the local file system and cleared the source table. In this way the collected data was easily transferrable, via the use of portable storage, to another machine for the analysis/further processing and also for the purpose of backup. It is not necessary that two distinct machines have to be used for the data collection and data processing, however the reality of the Internet-disconnected environment implies that the data needs to be transferrable out of the data collection environment. To illustrate the size of the data the system is dealing with one has to multiply the number of sensors deployed in the environment times their reporting frequency. Therefore, for 172 RFID tags, in the worst case, the system produces 172[tags] x 30[broadcasts per minute] = 5160 [events per minute]. Therefore a maximum of 10320 events get stored in the DB every 2 minutes and the DB holds up to 24[hrs] x 60[min]

x 5160[events per minute] = 7.43 million[events a day] and over 2.71 billion[events per year]. This worst-case scenario calculations demonstrate the volume of data which the system has to withstand and imposes some concrete requirements on the storage capability of the platform. Taking into consideration all the aforementioned factors and constraints the chosen storage component (MySQL) seemed appropriate.

5.2.6 Reliability Study

For any system to be trusted it has to be assessed for its reliability and validity. Reliability is the judgment on consistency of overall results i.e. whether replication of an experiment yields comparable results. Unit tests may show that certain components perform as required, however the entire system may not work as expected. Simulation – another technique useful in system development and testing – does not account for all possible scenarios. Therefore, a good way to measure the extent to which the system consistently measures some concept is to validate its performance in a real case study setting.

The reliability study took place at the RSU and involved 10 post-stroke patients, a physiotherapy student, the data collector, and a set scenario to execute. Ten patients were selected for this study – 5 male and 5 female – in order to get a statistically important representative sample (not biased by gender). The system-collected data was then analysed in order to judge on RTLS' consistency in reporting movement and location changes for patients wearing multiple tags.

5.2.6.1 Approach

Patients, assisted by one physiotherapy student, were asked to execute a pre-defined scenario (31-step script). This script was designed by our collaborators from the School of Healthcare Sciences, and purposely included instructions for patients to move between locations i.e. from their own room, to day room, to physiotherapy room. The scenario, captured in the form of an iPad application, was read out by the data collector. Patients had to respond to the instructions read and the data collector logged the time of execution of each of the tasks on the application provided (Fig. 5.7). Due to the fact that patients were asked to wear 3 tags on their wrist and 3 tags on their ankle, these tags' data could be compared against each other to test the reliability

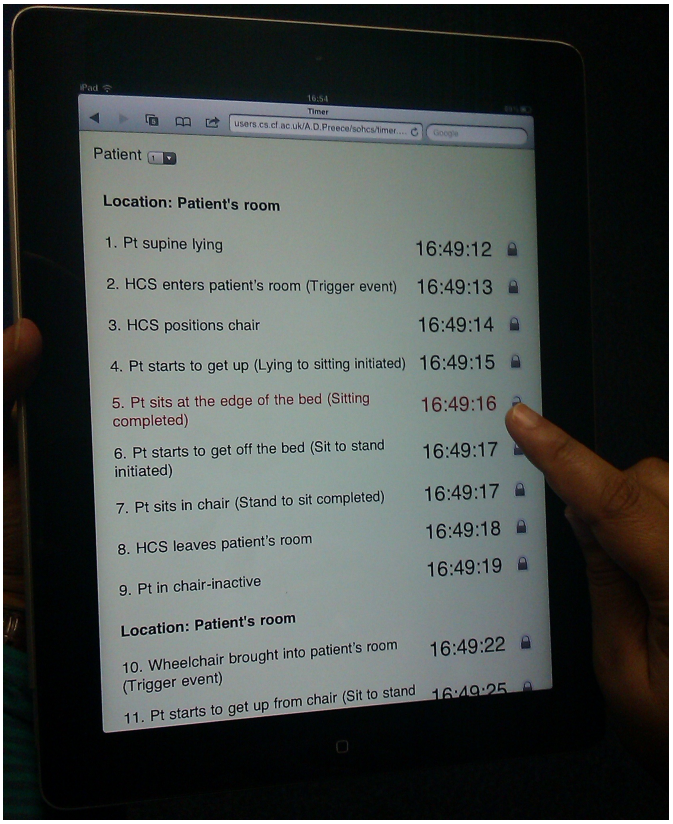


Figure 5.7: iPad Application used to capture “ground truth"

of the tags in reporting motion periods and acquiring location codes. The data collected for the reliability study was also planned to be used for the first phase of the system’s validation.

5.2.6.2 Requirements

The following is the list of requirements for the reliability study:

- Stroke patients wear three tags on a wrist band around their ankle and around their wrist.
- Walking aids and chairs are tagged.
- Locations (rooms) of interest are equipped with RLs.
- All the tags are within the antennas’ range.
- The system has storage abilities to permanently save the relevant RTLS data for further analysis.

- Patient is accompanied by a final year physiotherapy student (also tagged) for safety reasons.
- Patient, as instructed by the data collector, must move between locations.

In the experiment, the network consisted of 1 Linux PC running the platform together with an instance of the GSN server, PHP server and the Zone Manager; wireless router, 1 RFID reader with 2 omni-directional antennas and 2 room locators placed in the day room and physiotherapy room. To clarify, the software/hardware components used in the experiment have the following functions:

Linux PC: Runs the platform and the 3 servers.

Wireless Router: Forms a local network for the iPad to connect to and also connects the PC to the RFID Reader.

GSN Server: Connects to sensors, logs their readings and exposes them via a web server.

Zone Manager: Exposes an API to query for sensors' readings.

PHP Server: Hosts the PHP code for the iPad's script and handles the submission of data captured via the iPad.

5.2.6.3 Results & Discussion

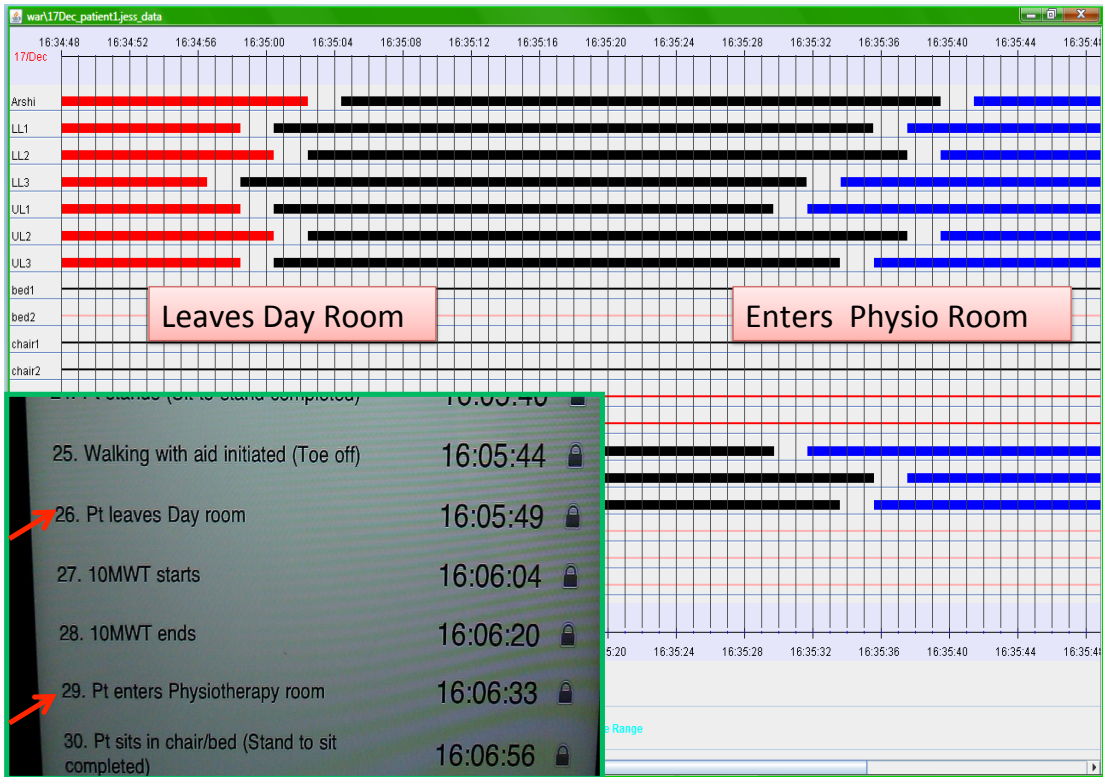


Figure 5.8: System Data & Ground Truth

For each patient the data collected by the system was analysed with the aid of a Java application implemented for this purpose. Fig. 5.8 represents the analytical application designed to display the collected data and the iPad's display of the PHP script. The analytical application displays the data for all tags against the time axis. Thick lines represent motion periods, where thin lines indicate no movement of tags. Different colours were used to colour-code the location recorded by each tag. Red stands for the day room, blue for the physiotherapy room and black indicates the lack of the IR location code (i.e. location 000). From the top, the displayed tags were attached to: the data collector, LL1 (Lower Limb 1), LL2, LL3, UL1 (Upper Limb 1), UL2, UL3, etc. The three UL tags (wrist) were confronted against each other, and the same for the LL tags (ankle), in order to establish the degree of reliability for the sensor-data recorded by the platform.

Table 5.1: Results of Reliability Study: ICC [32]

Time taken to walk from day room to physiotherapy room		
Patient Tags	ICC (3,3)(95%CI)	SEM[a]
Wrist	0.98 (0.99 to 0.94)	7 sec
Ankle	0.90 (0.97 to 0.76)	16 sec

[a]Standard Error of Measurement over a period of 60 seconds

The data displayed on Fig. 5.8, at the first look, suggests that LL and UL tags' data aligns nicely and is reliable. Detailed statistical analysis was carried out for every patient's data and the results of the study were presented in the Physiotherapy Research Society Conference [32] and in the Welsh Stroke Conference [30]. The statistics were calculated by our collaborator from the School of Healthcare Sciences, PhD candidate Arshi Iqbal. They were prepared for upper limb and lower limb tags (3 tags were worn on each) for the time reported to leave the day room and to enter the physiotherapy room (result of patients following the iPad's script). The overall ICC (Interclass Correlation Coefficient) score of 0.98 (wrist) and 0.90 (ankle) were calculated (Table 5.1) and based on the fact that they are both >= 0.90, the system showed **excellent reliability** in measuring the time taken by patients to walk between the two locations. Stronger correlation was achieved for the wrist tag, which suggest that upper limb is a better choice for tag's placement. Moreover, the Pearson's Correlations Coefficients (PCC) were calculated for patient tags against staff tag and for patient tags against the walking-aid tag. PCC for both is 0.98 (Table 5.2), which demonstrates that the system is **capable of reliably detecting** the assistance of walking aids or HCPs for the walking events.

Table 5.2: Results of Reliability Study: PCC [32]

Time taken to walk from day room to physiotherapy room	PCC
Patient tags and staff tags	0.98[a]
Patient tags and walking-aid tags	0.98[a]

[a]Significant correlation (2-tailed)

5.2.7 Recommendations for Rule Development

To give practical recommendations as to creating fast and efficient rules, one needs to take into consideration how RBSs handle rules and factors affecting the runtime. Most RBSs use the

Rete algorithm or a modified version of it. The complexity of the simple rules finding facts algorithm is RF^P, where R is the number of rules, F is the total number of facts, and P is the average number of patterns per rule [22]. The performance on the first cycle of the Rete algorithm is the same for both algorithms, as Rete has to do pattern matching for every fact in the working memory. Typically, in the subsequent iterations the working memory changes slowly. Rete uses the information about previous iterations to outperform the naive algorithm – only new or deleted facts are tested and only against the subset of the rules that may actually match. This results in the following Rete runtime complexity: $R'F'^{P'}$, where R' is a number less than R, F' is the number of facts that change on each iteration, and P' is a number greater than one but less than the average number of patterns per rule.

If one wants an RBS to perform fast they need to keep the number of rules to minimum. However, the algorithm performance is only directly proportional to the number of rules, and hence if the number of rules doubles, the performance time will double too. In the case of other factors, number of facts F in WM is something that programmer has very little influence on. Therefore, unnecessary facts should not be produced. F' also is almost beyond developer's control as if certain conditions are met the rule fires and modifies the WM. Unnecessary WM modifications should be avoided, which is usually the aim of the developer. The last factor to be considered is P – the average number of patterns per rule. This is probably the most influential factor in the equation and the only exponential one. It is also the factor on which the developer has the most influence on when developing rules, and therefore needs to be highlighted.

The following recommendations are based on best practices found in the literature [22] and find their explanation in the runtime complexity analysis. For the RBS to perform fast one needs to adhere to the following recommendations:

Keep the number of patterns for each rule low It has been tested and is derived directly from the runtime complexity analysis that RBS performs faster when the number of patterns per rule are small. In general, it is more efficient to write two rules with two patterns than one rule with three patters. In the first case the overall complexity is (R=2, P=2) $2F^2$ and in the later (R=1, P=3) it is F^3, so for F>2 the first case is more efficient. Normally RBS have to deal with thousands of facts and hence having more rules of smaller complexity is better. A good example for this can be the Sandwich rule which eventually

was implemented by multiple rules of lower complexity.

The 'or' conditional element Presence of this element on the rule's LHS increases rule's number of patterns. Again, as in the earlier case, it is more efficient to have two rules instead of one with the 'or' condition.

Use of functions on rule's LHS Computationally expensive functions used within the 'test' conditional element are also inefficient. It forces the RBS to evaluate the given function instead of matching a fact from WM against it – as this is usually the case with the rule's LHS. It has been tested that a rule with a function call (especially for complex functions) on its RHS is more efficient than the one calling it on its LHS.

Facts that are in fewer quantitates need to be listed first If the rule tests for, for example, `Person` and `MovingObject` facts and there are fewer instances of the first, then it should be listed first on the rule's LHS. Therefore, the network will be constructed with fewer branches at the top level, which usually limits the number of branches at the lower levels.

Use rule's priority In Jess the priority of a rule is referred to as its *salience* This attribute specifies rule priority and higher priority rules get executed first. Therefore, to get correct results, and in some cases to increase RBS efficiency, one should consider the order in which rules ought to run.

Use of modules Whenever possible, the problem should be separated into unrelated sub-problems, which can be solved in separation from each other. This reduces the problem's complexity, as rules only apply to the proportion of facts in working memory and apply only to relevant modules. Moreover, such a separation helps developer to test the system and to isolate relevant concepts.

5.2.8 Domain Knowledge & Rehabilitation Rules

5.2.8.1 Rehabilitation Ontology

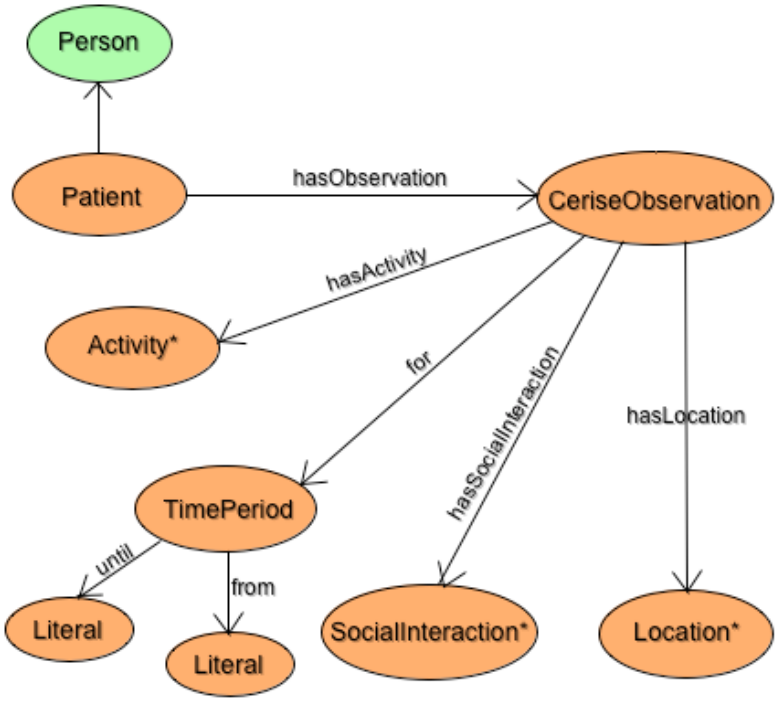


Figure 5.9: The Rehabilitation Ontology

The Rehabilitation ontology has been developed for the purpose of the two pilot studies. It is based on the concepts included in the CERISE [15] tool – a standardised physiotherapy measure, which records patient's activity, location and social interaction in 10min intervals (CERISE form in Appendix A.1). We have ontologised the CERISE model so that an automated sensor system can strictly adhere to this well-established measure and therefore its data can be compared against hand-recorded ground truths. This ontology has been captured in the Open Biological and Biomedical Ontologies (OBO) format via the use of the OBO-Edit tool[4]. This particular tool was chosen to express the rehabilitation model, as it was designed to be used in the medical context. Moreover, it is very simple to use and has an OWL export tool, which allows for automatic translation to the Web Ontology Language format. Figure 5.9 visualises the *Rehabilitation* ontology. `Patient` class, which is a subclass

[4]http://oboedit.org/

of a `Person` (Fig. 4.2), has `hasObservation` property. The range of this property is the `CeriseObservation` class which has 4 properties. The `hasActivity` property describes the `Activity` the patient was performing when the measurement took place. The `hasLocation` property simply points to the location where the observation took place i.e. can be extracted from the *Activity's* `atPlace` property. The `hasSocialInteraction` property describes the `SocialInteraction` the patient had e.g. with *Nurses, Doctors, Therapists* (all these are subclasses of the `SocialInteraction` and can be extracted from *Activity's* `involvedAgent` slot). Finally, the `CeriseObservation` is for a particular time interval, which is captured by the `for` property (observations are made in 10min intervals, however the *Activity's* `atTime` describes the start and finish time of some particular *Activity* that took place within this 10min slot). As it is visualised by Fig. 5.10, *Activity* is a subclass of *Hospital_Activity(CERISE)*, which in turn is a subclass of the *Event* class. The *Activity* class is further specialised into two categories: *Therapeutic* and *NonTherapeutic* activities, which are further specialised into a number of subclasses. Using this Rehabilitation ontology, one is able to express that, for example, Mrs. Jones had a physiotherapy session, with a therapists, in the physiotherapy room, at some point in time.

An asterisk in classes' names in Fig. 5.9 denote that these classes have subclasses, which have been omitted for brevity. However, the full ontology can be found in Appendix B.

5.2.8.2 Goal Events Layer Rules

The Goal Events Layer (GEL) consists of domain specific rules that can be developed in co-operation with experts from the given field. These rules are responsible for generating the HLEs of interest, in the case of the pilot study, rehabilitation events. Rules described in this section produce CERISE activities highlighted in bold in Fig. 5.10. In some rules an attribute 'salience' is used to define rule's priority. Higher priority rules get executed first. In order to improve the system's performance and to maintain the data in categories to which they belong, the use of modules is facilitated by the rules presented in this section. Modules precede the fact's name and are separated from it via the two colons notation e.g. MAIN::Colocated – means that the fact *Colocated* is to be found in module "MAIN". Rules presented in the reminder of this section make use of the `Patient`, `Equipment`, `Staff` and `Room` facts which persist in the DB and

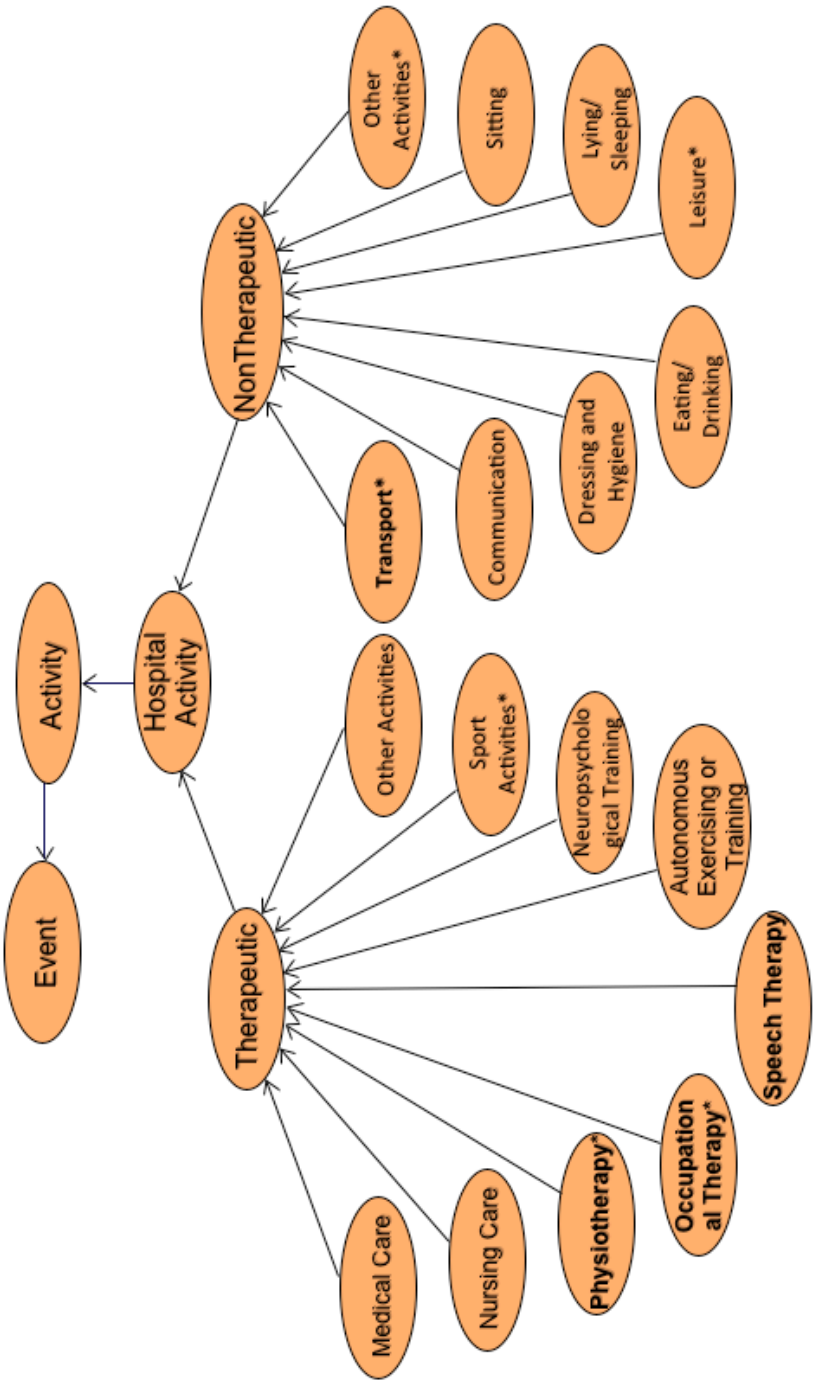


Figure 5.10: The Hospital Activity Classes

carry detailed information about these parties. Moreover, each rule finishes with a call to the `(store <string or symbol> <expression>)` function, which use is required for system verification – more on this in Chapter 6.

Due to the fact that some GEL rules (e.g. Physiotherapy) for correct working need to know patients' rooms, the RTL data is analysed in order to discover/update these. This is done mostly because patients do get moved around the ward and these moves may not necessarily be captured by the system's administrator. Therefore, rather than relying on this information to be supplied by unit's staff, it is automatically discovered. For every day that is analysed, patient's location is looked up at 7am in the morning and if it is "000" (no RL signal) then a backup strategy fires i.e. their location is looked up at 8:30am. However, if this also fails then patient's bed location is checked at 8:30am on that day. The three strategies for patient's location discovery are not coincidental. Patients at such early hours should, in principle, be in their beds. However, the user who runs the analysis gets also asked to confirm patients' locations before the GEL rules are fired. Therefore, even if a bed reports "000", the user still has a chance to input the missing information. Moreover, it cannot be assumed that patients are always in their rooms at such early times and therefore a manual check and corrections are beneficial.

All `Hospital Activities` are kinds of therapy and hence are captured by the model in Fig.5.10. `Therapeutic` activities are those which are provided or prescribed by various health-care professionals e.g. `Speech Therapy` or `Sport Activities`. On the other hand, `Non Therapeutic` activities are self-driven actions, which patients perform on their own or in assistance of a nurse e.g. `Transport` between locations. All `Hospital Activities` are related to patient's movement, as increasing their mobility is the aim of the rehabilitation process. From the therapeutic point of view, it is important that patients become independent in executing these activities on their own, so that they can be discharged from the hospital and live independent life. It is normal that when patients are admitted to the stroke unit their mobility is low, but through regular therapies and exercises, assisted by HCPs at early stages of recovery, their mobility improves to such level that they no longer need assistance and become independent.

The reminder of this section describes rules for recognition of various `Hospital Activities`. It starts with the `Physiotherapy` rule which is an example of the `Therapeutic` activity. Next come subclasses of `Transport`, which is a descendant of `Non Therapeutic` activity. `Transport` is a superclass of `Walking` and `Wheelchair_Ambulation` activities. HCP's as-

sistance in any of the above activities is captured by filling the `involvedAgent` slot of events.

5.2.8.2.1 Physiotherapy Rule

In the rehabilitation context, the physiotherapist may want to know, for example, how much physiotherapy time each patient gets in the course of a week. Having already defined the rule for composing `Colocated` events in Section 4.3.3, the physiotherapy rule re-uses facts that are already in the WM.

```
(defrule Cerise_Physiotherapy_Rule
 (declare (auto-focus TRUE))
 (Room::Room (name ‘‘physio")(rlcode ?physioLoc))
 (Room::Room (name ?ptDLoc)(rlcode ?defLoc))
 (Patients::Patient (name ?ptName)(defaultLocation ?defLoc)(deviceAddress ?ptRFAddress))
 (Staff::Staff (name ?staffName)(role "physio")(defaultLocation ?staffDLoc)(deviceAddress
   ?staffRFAddress))
 (MAIN::Colocated (atPlace ?loc1&:(member$ ?loc1 (create$ ?physioLoc ?defLoc)))(involved $?things)
  (involvedAgent $?agents&:(eq $?agents  (create$ ?ptName ?staffName)))(atTime $?time)(id ?id)
  (evidencedBy $?evidences))
  =>
 (bind ?id2 (gensym*))
 (assert (CERISE::Physiotherapy (atPlace ?loc1)(atTime $?time)(involved $?things)
  (involvedAgent $?agents)(id ?id2)(evidencedBy ?id)))
 (assert (CERISE::EvidenceFor (rule "Rule_Cerise_Physiotherapy")(factID ?id2)))
 (store Cerise_Physiotherapy_Rule "Cerise_Physiotherapy_Rule")
)

sample output fact:
Physiotherapy (atPlace "100")(atTime (1337078615758 1337080599868))(involved ("nil" "nil"))
 (involvedAgent ("patient_2" "staff_1"))(id cgen146)(evidencedBy gen35545)
```

The Cerise_Physiotherapy_Rule, represented in CLIPS syntax, fires when **patient and physiotherapist are co-located in patient's room or in physiotherapy room**. In depth analysis of the outcomes of the above rule revealed that there was a huge number of physiotherapy activities recognised for each patient. All these activities were a couple of minutes long and, compared against manually collected data, were correct. However, instead of 30 physiotherapy events each lasting around 1-2min it would be more feasible to infer one physiotherapy event with the duration of 1 hour. The fragmentation of the results was due to the system occasionally reporting the lack of location for the involved entities. Whether it is the tag's orientation against the room locator, objects in between a tag and an RL, or the fact that the patient covered their tag with a pillow, the system should be able to deal with these situations. Therefore, the so called 'Sandwich_Rule' was introduced to the system's SEL layer to solve this problem in the layer which precedes the GEL layer. The Sandwich_Rule (or rather rules 3a-g), which

was already described in Chapter 4.3.2, ensures that the lowest 'building blocks' are not fragmented. The fact that this rule lives in the SEL layer ensures that none of the *Events* inferred in the GEL layer would ever suffer from the fragmentation problem. Therefore this one rule, injected into the appropriate layer, solves the problem permanently. Consequently, whoever is implementing rules in the GEL layer, does not even have to be aware of the shortcomings of the physical sensors – as some of these are already dealt with.

The Cerise_Occupational_Therapy rule takes almost identical shape to the Physiotherapy rule, as already mentioned in the previous section. Therefore, for brevity, it is not analysed in detail in this section.

5.2.8.2.2 (Assisted)Walking Rules

Walking is very important in the context of rehabilitation. Healthcare professionals are in need to know how much time their patients spent walking, how fast they walk and what distance they cover everyday. Moreover, knowing if the walk was assisted by some walking aid or not is a good indication of patient's ability to walk independently. The reliability study has proven that the system is able to reliably detect the use of walking aids for patients walking and hence it is possible to implement rules which accomplish this task. The rules introduced below are governed by a small set of assumptions, which have been established in collaboration with domain experts. For a patient to be considered walking, they need to transfer between different locations and spend a maximum of 3 minutes at this location. If the transfer only involves moving between two locations then the time limit does not apply e.g. a patient can leave their room and go to the day room to watch TV – even though short, this walk is also to be considered. Worth noting is the fact that these rules have been developed in cooperation with healthcare specialists through a number of iteration cycles. Their knowledge of the rehabilitation domain assisted in the process of rules' refinement, which lead to the successful validation of the rules presented in the reminder of this section.

The Relocation_Rule is the starting point for inferring `Walking` events. This rule fires when **patient is subsequently observed at two different locations and spends at most 3min at the later**. Upon satisfying these conditions, the rule fires and asserts a `RelocationEvent` and an

`EvidenceFor` facts to the Patients module. The meaning and importance of the `EvidenceFor` fact has already been discussed in the preceding chapter. The asserted `RelocationEvent` inherits most of its slots' values from the `MovingObjects` facts and records both their locations in its `atPlace` property. The `atTime` slot is populated with finish times of the earlier and later `MovingObjects` i.e. the start of `RelocationEvent` is when the person leaves the earlier location and its finish time is when they leave the later location. Worth noticing is the fact that this rule only restricts the time spent at the later location and hence produces `RelocationEvents` for `MovingObjects`, first of which was of any length and the later of maximum 3 minutes i.e. <any, short>.

```
(defrule Relocation_Rule
 (declare (auto-focus TRUE)(salience 100))
 (Patients::Patient (name ?agent)(defaultLocation ?location)(deviceAddress ?ptRFAddress)
  (shortName ?shortName))
 (Patients::MovingObject (atPlace ?location1&~"999")(involved "nil")(involvedAgent ?agent)
  (atTime $?time1)(id ?id1)(evidencedBy $?ev1))
 (Patients::MovingObject (atPlace ?location2&~"999"|?location1)(involved "nil")
  (involvedAgent ?agent)(atTime $?time2)(id ?id2&~?id1)(evidencedBy $?ev2))
 (test (= (nth$ 1 $?time2) (nth$ (length$ $?time1) $?time1)))
 (test (< (- (nth$ 1 $?time2) (nth$ (length$ $?time1) $?time1)) 3000))
 (test (< (- (nth$ (length$ $?time2) $?time2) (nth$ 1 $?time2)) 180000))
  =>
 (bind ?id (str-cat RL (gensym*)))
 (assert (Patients::RelocationEvent (atPlace (create$ ?location1 ?location2))(involved "nil")
  (involvedAgent ?agent)(atTime (nth$ (length$ $?time1) $?time1)(nth$ (length$ $?time2) $?time2))
  (id ?id)(evidencedBy ?id1 ?id2)))
 (assert (Patients::EvidenceFor (rule "Rule_Relocation")(factID ?id)))
 (store Relocation_Rule "Relocation_Rule")
)
```

The Relocation_Merge_Rule fires when **RelocationEvent A finishes at the same location and time where/when the RelocationEvent B starts**. The reasons for these conditions are as follows. The preceding rule (Relocation_Rule) produces `RelocationEvents` which are made up of two locations. Let's consider the following example. An object passes through the following locations: locA (t1 till t2), locB (t3 till t4), locC (t5 till t6). The Relocation_Rule produces two `RelocationEvents`: first event is for locations pair locA and locB from t2 till t4; the second event is for locations pair locB and locC from t4 till t6. In order to merge these two events into a longer one, apart from the first condition which matches the last and first location, the two time periods need to be tested for the equality of their start and finish times (i.e t4 = t4) and also they need to happen in order i.e. one after the other. Therefore, when unifying `RelocationEvents` made up of `MovingObjects` of the duration <any,short> + <any,short>, the 'any' value of the later must be 'short' i.e. maximum 3 minutes long due to the fact that it is matched against the 'short' value of the earlier (locations and times need to match). Upon

firing, the below rule modifies the first `RelocationEvent` by adding subsequent locations to the `atPlace` slot and by updating its finish time. Also, the `evidencedBy` property gets updated and the `EvidenceFor` fact is asserted. Finally, the second event is retracted as it is no longer needed. If this event persisted in the memory, it would be considered by the next rule, which would result in having overlapping events in the WM.

```
(defrule Relocation_Merge_Rule
 (declare (auto-focus TRUE)(salience 10))
 ?fact1 <- (Patients::RelocationEvent (atPlace $?locations1)(involved "nil")(involvedAgent ?agent)
  (atTime $?time1)(id ?id1)(evidencedBy $?ev1))
 ?fact2 <- (Patients::RelocationEvent (atPlace $?locations2)(involved "nil")(involvedAgent ?agent)
  (atTime $?time2)(id ?id2&~?id1)(evidencedBy $?ev2))
 (test (eq* (nth$ (length$ $?locations1) $?locations1) (nth$ 1 $?locations2)))
 (test (= (nth$ 1 $?time2) (nth$ (length$ $?time1) $?time1)))
 (test (> (nth$ (length$ $?time2) $?time2) (nth$ (length$ $?time1) $?time1) ))
  =>
 (modify ?fact1 (atTime (replace$ ?time1 2 2 (nth$ (length$ ?time2) ?time2)))
  (atPlace ?locations1 (delete$ ?locations2 1 1)) (evidencedBy ?ev1 ?id2))
 (assert (Patients::EvidenceFor (rule "Rule_RelocationMerge")(factID ?id1)))
 (retract ?fact2)
 (store RelocationMerge_Rule "RelocationMerge_Rule")
)
```

Once all `RelocationEvent` pairs are merged the RelocationExt_Rule fires. This rule looks for **immediate patient movement (max 3min long) at location different to the last location recorded in RelocationEvent**. When the rule fires, the `RelocationEvent` fact gets updated with the relevant information: new location is appended, finish time is updated and the new evidence added to the `evidencedBy` slot. In short, this rule adds a tail to the `RelocationEvent` i.e. a continuation of a walk.

```
(defrule RelocationExt_Rule
 (declare (auto-focus TRUE)(salience 5))
 ?fact1 <- (Patients::RelocationEvent (atPlace $?locations1)(involved "nil")(involvedAgent ?agent)
  (atTime $?time1)(id ?id1)(evidencedBy $?ev1))
 (Patients::MovingObject (atPlace ?location2)(involved "nil")(involvedAgent ?agent)
  (atTime $?time2)(id ?id2)(evidencedBy $?ev2))
 (test (neq (nth$ (length$ ?locations1) ?locations1) ?location2 ))
 (test (>= (nth$ 1 $?time2) (nth$ (length$ $?time1) $?time1)))
 (test (< (- (nth$ 1 $?time2) (nth$ (length$ $?time1) $?time1)) 3000))
 (test (< (- (nth$ (length$ $?time2) $?time2) (nth$ 1 $?time2)) 180000))
  =>
 (modify ?fact1 (atPlace ?locations1 ?location2)(atTime (replace$ ?time1 2 2
  (nth$ (length$ ?time2) ?time2))) (evidencedBy ?ev1 ?id2))
 (assert (Patients::EvidenceFor (rule "Rule_Relocation")(factID ?id1)))
 (store RelocationExt_Rule "RelocationExt_Rule")
)
```

Due to the fact that the Relocation_Rule imposes a maximum duration condition of 3min on the later `MovingObject`, there is one more case which needs to be considered. This case is the pair of `MovingObjects` out of which the first one if of any duration and the later over 3min

i.e. <any,long>. Because a pair of events which creates such a `RelocationEvent` does not meet the criteria of the maximum duration of 3min at the subsequent location, it should not be considered for merging with any other `RelocationEvents` and hence the following rule fires last.

```
(defrule RelocationPairsLong_Rule
 (declare (auto-focus TRUE)(salience 1))
 (Patients::Patient (name ?agent)(defaultLocation ?location)(deviceAddress ?ptRFAddress)
  (shortName ?shortName))
 (Patients::MovingObject (atPlace ?location1&~"999")(involved "nil")(involvedAgent ?agent)
  (atTime $?time1)(id ?id1)(evidencedBy $?ev1))
 (Patients::MovingObject (atPlace ?location2&~"999"|?location1)(involved "nil")
  (involvedAgent ?agent)(atTime $?time2)(id ?id2&~?id1)(evidencedBy $?ev2))
 (test (>= (nth$ 1 $?time2) (nth$ (length$ $?time1) $?time1)))
 (test (< (- (nth$ 1 $?time2) (nth$ (length$ $?time1) $?time1)) 3000))
 (test (> (- (nth$ (length$ $?time2) $?time2) (nth$ 1 $?time2)) 180000))
  =>
 (bind ?id (str-cat RL (gensym*)))
 (assert (Patients::RelocationEvent (atPlace ?location1 ?location2)(involved "nil")
  (involvedAgent ?agent)(atTime (nth$ 1 $?time1) (nth$ (length$ $?time2) $?time2))
  (id ?id)(evidencedBy ?id1 ?id2)))
 (assert (Patients::EvidenceFor (rule "Rule_Relocation")(factID ?id)))
 (store RelocationPairsLong_Rule "RelocationPairsLong_Rule")
)
```

The Walking_Rule below is the final step in asserting `Walking` events to the WM. For every `RelocationEvent` it asserts the `Walking` fact which has all the properties of th `RelocationEvent`. Moreover, the `EvidenceFor` fact gets added and the `RelocationEvent` is deleted from the WM.

```
(defrule Walking_Rule
 (declare (auto-focus TRUE)(salience -10))
 ?fact1 <- (Patients::RelocationEvent (atPlace $?locations)(involved "nil")(involvedAgent ?agent)
  (atTime $?time2)(id ?id2)(evidencedBy $?ev2))
  =>
 (assert (CERISE::EvidenceFor (rule "Rule_Walking")(factID ?id2)))
 (assert (CERISE::Walking (atPlace ?locations)(involved "nil")(involvedAgent ?agent)
  (atTime $?time2)(id ?id2)(evidencedBy ?ev2)))
 (retract ?fact1)
 (store Walking_Rule "Walking_Rule")
)
```

Finally, the Assisted_Walking_Rule fires when **patient's Walking event is matched with their allocated walking aid's Journey event**. The `Journey` facts are inferred by the same set of rules which deal with `Walking` but instead of `Patients` they consider instances of `Equipment`. For brevity, these rules are not listed here. This rule's RHS tests for two conditions. Firstly, the compute-similarity() function needs to return a number greater than 0.4 and secondly, patient's time and walking aid's time need to overlap – computed by the overlaps() function. There is one reason why these functions are not called in the head of this rule but in

its body. Calling such a computationally complex function has a huge effect on performance. Therefore, it has been observed that the performance of this rule is best when these functions are called on the rule's RHS, rather than increasing the rule's LHS Rete's network complexity [22]. If the if-statement evaluates to true, the `involved` slot of the `Walking` event is populated with equipment's name and the `EvidenceFor` fact is asserted.

```
(defrule Assisted_Walking_Rule
 (declare (auto-focus TRUE)(salience -100))
 (Equipment::Equipment (name ?aid)(type "walk_aid")(defaultLocation ?location)
  (deviceAddress ?ptRFAddress)(ownedBy ?shortName))
 (Patients::Patient (name ?agent1)(defaultLocation ?locationB)(deviceAddress ?ptRFAddressB)
  (shortName ?shortName))
 ?fact1 <- (CERISE::Walking (atPlace $?locations1)(involved "nil")(involvedAgent ?agent1)
  (atTime $?time1)(id ?id1)(evidencedBy $?ev1))
 (Equipment::Journey (atPlace $?locations2)(involved ?aid)(involvedAgent ?agent2)
  (atTime $?time2)(id ?id2)(evidencedBy $?ev2))
 =>
 (bind ?time (overlaps ?time1 ?time2))
 (if (and (> (compute-similarity ?locations1 ?locations2) 0.4) (neq 0 ?time)) then
   (assert (CERISE::EvidenceFor (rule "Rule_Assisted_Walking")(factID ?id1)))
   (modify ?fact1 (involved ?aid))
   (store Assisted_Walking_Rule "Assisted_Walking_Rule")
 )
)
```

All the above rules have been successfully validated against observed, hand-collected data – see Section 5.2.9.2. The `Walking` facts can further be used to compute other relevant information, such as: the total distance walked, or walking speed; in order to better assess patient's recovery. Interestingly, during the validation process the truth about one patient was revealed. This patient was known not to walk and had to be wheelchaired between different locations, however he was found walking with the use of a walking aid. Due to the unobtrusive nature of the RTL technology used in this project, this patient was captured behaving naturally and in the presence of HCPs they could act differently.

5.2.8.2.3 Wheelchair_Ambulation Rules

The Walking rules classify patients' appearances in various locations as `Walking` events which are either independent or assisted by assigned to patients walking aids. The Wheelchair_Ambulation rule, in the presence of additional knowledge, re-classifies some of the `Walking` events to `Wheelchair_Ambulation` events. This does not mean that the Walking rules work incorrectly. With all the information available to the Walking rules at the given time they correctly infer

Walking events. However, in the presence of new information, existing Walking events are revisited and instead of filling their involved slot with an instance of a wheelchair (which would have been a correct action) they are transformed into Wheelchair_Ambulation. The reason why the involved slot isn't just modified is due to the rehabilitation ontology. The Transport class has two sub-classes: Walking and Wheelchair_Ambulation; and hence Walking events assisted by a wheelchair become Wheelchair_Ambulation events.

The Wheelchair_Ambulation rule fires when **patient's Walking event is matched with their allocated wheelchair's Journey**. The RHS of the rule is almost identical to the Assisted_Walking rule's RHS. Again, the two sets of locations are tested for similarity and the two times need to overlap. Upon satisfying these conditions, the WheelchairAmbulation fact is asserted and the Walking fact is deleted from the WM.

```
(defrule Wheelchair_Ambulation_Rule
 (declare (auto-focus TRUE)(salience -100))
 ?fact1 <- (Equipment::Journey (atPlace $?locations1)(involved ?object1)(involvedAgent ?agent1)
  (atTime $?time1)(id ?id1)(evidencedBy $?ev1))
 ?fact2 <- (CERISE::Walking (atPlace $?locations2)(involved ?object2)(involvedAgent ?agent2)
  (atTime $?time2)(id ?id2)(evidencedBy $?ev2))
 (Equipment::Equipment (name ?object1)(type "wheelchair")(defaultLocation ?locationA)
  (deviceAddress ?ptRFAddressA)(ownedBy ?shortName))
 (Patients::Patient (name ?agent2)(defaultLocation ?locationB)(deviceAddress ?ptRFAddressB)
  (shortName ?shortName))
  =>
 (bind ?id (str-cat WLC (gensym*)))
 (bind ?time (overlaps ?time1 ?time2))
 (if (and (> (compute-similarity ?locations1 ?locations2) 0.4) (neq 0 ?time)) then
   (assert (CERISE::EvidenceFor (rule "Rule_Wheelchair_Amb")(factID ?id)))
   (assert (CERISE::WheelchairAmbulation (atPlace ?locations2)(involved ?object1)
    (involvedAgent ?agent2)(atTime ?time2)(id ?id)(evidencedBy ?id1 ?id2)))
   (retract ?fact2)
   (store Wheelchair_Ambulation_Rule "Wheelchair_Ambulation_Rule")
 )
)
```

5.2.8.2.4 Speech and Language Therapy Rules

The Speech and Language Therapy (SALT) rule fires when **patient is seen at either of the three locations leading to SALT offices, goes out of range, and after some time is seen again within the RTLS' range**. Therefore, the rule looks for instances of Patients for whom MovingObject facts exist – which took place in one of the three given locations. Then a StaticObject is needed for location equal to "999" for that patient. The "999" code means that the tracked object is out of the RTLS' range and hence "999" location is only seen for instances of the StaticObject and not MovingObject (in the absence of knowledge patient mobility records should not be increased). Then three conditions are tested. Firstly, the

start time of the StaticObject needs to be after or equal to MovingObject's finish time. Secondly, the two events need to be within 180000 millisecond (3min) of each other. Finally, the time spent outside RTLS' range needs to be over 5min. In the conclusion of this rule, the MovingObject's atPlace slot is modified with the "444" location code in order to reduce the total of out-of-range facts. Then the Speech_Therapy fact is asserted with all the slots filled with the relevant information.

```
(defrule Rule_SALT
 (declare (auto-focus TRUE)(salience -10))
 (Patients::Patient (name ?agent)(defaultLocation ?ptDLoc)(deviceAddress ?ptRFAddress)
  (shortName ?sn))
 (Patients::MovingObject (atPlace "300"|"500"|"700")(involved ?object1)(involvedAgent ?agent)
  (atTime $?time1)(id ?id1)(evidencedBy $?ev1))
 ?fact <- (Patients::StaticObject (atPlace "999")(involved ?object2)(involvedAgent ?agent)
  (atTime $?time2)(id ?id2&~?id1)(evidencedBy $?ev2))
 (test (>= (nth$ 1 $?time2) (nth$ (length$ $?time1) $?time1)))
 (test (< (- (nth$ 1 $?time2) (nth$ (length$ $?time1) $?time1)) 180000))
 (test (> (- (nth$ (length$ $?time2) $?time2) (nth$ 1 $?time2)) 300000))
  =>
 (modify ?fact (atPlace "444") )
 (bind ?id (str-cat SALT (gensym*)))
 (assert (CERISE::Speech_Therapy (atPlace "444")(atTime $?time)(involved "nil")
  (involvedAgent ?agent)(id ?id)(evidencedBy ?id1)) )
 (assert (CERISE::EvidenceFor (rule "Rule_SALT")(factID ?id)))
 (store Rule_SALT  "Rule_SALT")
 )
```

5.2.9 Validity Study

Having established that the system was reliable, the validity study was conducted to judge on its outcomes' strength. The design of this study and its results were discussed with domain experts in order to check whether the implemented system measured correct parameters.

The validity study was carried out in two phases. Phase 1 took place at the beginning of clinical tests in the RSU and used the same data set as the reliability study. The 31-step script-based data collection was designed purposely to capture enough information and of sufficient quality in order to establish the reliability and validity of the system. Again ten patients' data was analysed for this study – 5 male and 5 female. Once the system was proven to be reliable and valid, a longitudinal study started and data for all patients in the unit (apart from those who did not consent) was continuously collected 24/7 over the period of 1 year – this time using just one tag on the wrist (compulsory for consenting participants) and one on the ankle (optional if a patient agreed). The second phase of validation took place at the beginning of the longitudinal

study in order to validate the rehabilitation rules developed to date and also to provide the means by which validation of the rules developed in the future would be possible.

5.2.9.1 Phase 1: Script-based Validation

5.2.9.1.1 Requirements

Due to the fact that the same data was used for reliability and validity studies, the requirements for the system remain the same as for the reliability study with some additional ones:

- The PC was accompanied by a wireless router and it also run a PHP server.
- The iPad needed to be in the range of the locally-created network in order to load the script and to submit its data to the server.
- The data collector set the tasks and logged/timed their execution on the iPad.
- PC's and iPad's clocks either had to be synchronised or their time difference recorded before or immediately after data collection.

The 31-step long scenario, which the data collector followed and based on it gave instructions to patients and the physiotherapy student, can be found in Appendix A.2. As already explained in the aforementioned requirements, the data collector clicked each of the items when the action was initiated. This allowed for precise recording of patients' actions.

5.2.9.1.2 Approach

The approach to validate the system is very straightforward. 10 patients were selected to perform the script, while wearing three tags on the wrist and three tags on the ankle. The script was tested amongst the research group first in order to polish it to perfection and decide on the redundant or missing items. All the other objects and people involved in the scenario were also tagged. The execution times of each action were hand-recorded via the use of an iPad application to produce the ground truth for comparison against the data recorded by the system.

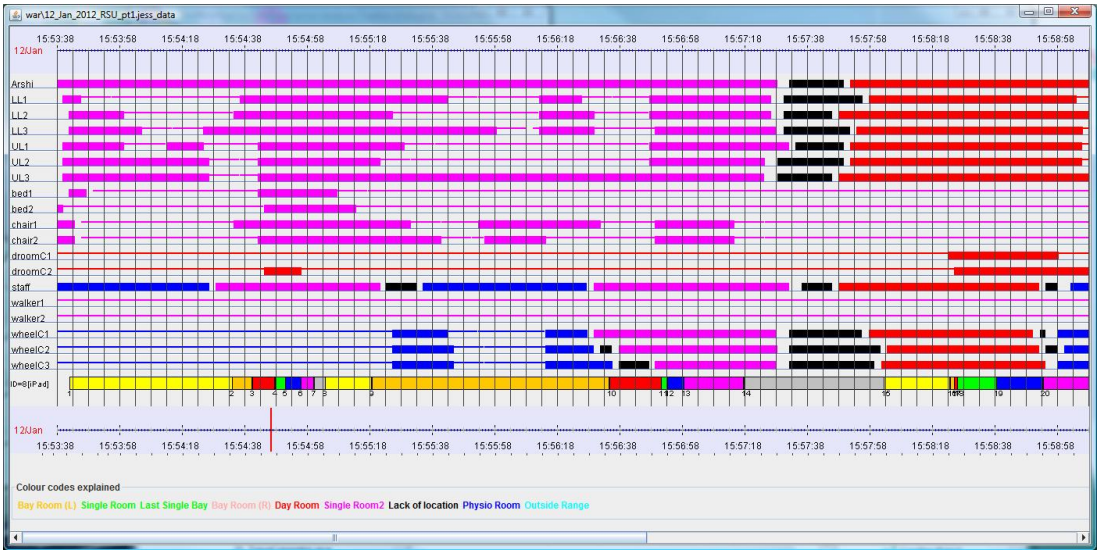


Figure 5.11: The Validation Application

The Java application written for the reliability study was extended to display iPad's data for a patient- Fig.5.11. On this figure, the iPad's data is displayed last – just above the timeline. This data has been colour coded and labeled with numbers, which correspond to the steps of the script. Due to the fact that some actions were not far apart, this application provided a function to select & magnify data for some. The detailed analysis of the tag's behaviour has also provided more in-depth understanding of the RFID technology. Based on this knowledge, further development of rehabilitation rules was much faster.

5.2.9.1.3 Results & Conclusion

During the comparison process, the time offset between the iPad's clock and the PC's clock was taken into consideration. The results were presented in the Welsh Stroke Conference [30]. The system was validated by the means of, so called, the 10-Meter Walk Test (10-MWT). The 31-step scenario includes a walk from the day room to the physiotherapy room. The distance between the two rooms was measured in order to calculate the average speed for the 10-MWT. The mean walking speed values for the system and the 10-MWT were identical, with marginally different standard deviation values - Fig. 5.3. Similarly to the reliability study, the PCC was calculated but this time for the system vs the hand recorded observations (OBS); and for the system vs 10-MWT. The PCC scores of 0.97 and 0.93 respectively, were obtained. The

Bland and Altman's 95% Limits of Agreement (LOA) were ranging from +0.34 to -0.18 and from +0.27 to -0.29 respectively (Fig.5.4), which yields a significantly strong correlation with the ground truth. Strong correlations and acceptable LOAs imply that the system is valid for walking speed measurements.

Table 5.3: Results of Validity Study (1) [30]

SSWS (m/s)	RBS2P	10-MWT	OBS
Mean (SD)	0.48 (0.34)	0.48 (0.25)	0.40 (0.22)

Table 5.4: Results of Validity Study (2) [30]

SSWS Comparison	PCC	95% LOA
RTLS vs. 10-MWT	0.93	+0.27 to -0.29
RTLS vs. OBS	0.97	+0.34 to -0.18

In addition to the 10-MWT, other actions that patients were asked to perform were also analysed and resulted in interesting findings. Firstly, the motion periods of the tags correlated well with the hand-recorded times of activities. The start times of each tag's motion period were very close to the observed movement of patients/objects. In every case, the finish times of motion periods were later than the observed, which also added to the conclusion that the tag needs time to settle its motion status. However, the motion settlement periods were not constant, varying between 30-60 seconds. Similarly for the location. All tags were found to report their previous location when in spaces with no room locator coverage for various periods of time. However, when changing from one RL location to another, this was not an issue. The results of the validity study triggered a dialog with the supplier's technical support team in order to confirm and clarify the findings. In response, the following conclusions were drawn:

- Tags have been designed to quickly report the change from stationary to motion but not the other way round. Therefore, it can pick up the movement very quickly (within 1-2sec) but it takes anywhere between 30-60 seconds to report that it is stationary again.
- The tag models used for this project (M160-i) report their signals every 2 seconds if the tag is in motion and 10 seconds otherwise. Thanks to this strategy tag's battery lasts for around 3 years.

- Tags are quick in reporting new locations (within 4 seconds) and are slower in reporting the loss of location

Therefore, a situation when the tag travels from one RL-enabled location to the other is not a problem. However, when it travels from an RL-enabled location into a location not equipped with the RL, it can take much longer to report the '000' code. The supplier's technical team gave a very good reason for this being: "if it did not do this, then the tag would more frequently report a loss of IR lock even though it was still in proximity to a locator". Even though it sounds like a sensible choice. It would be more beneficial if the tag did not do this, as this is something that software could easily correct if needed (in fact the "sandwich rule" presented in the previous chapter deals with such situations). Most of the above limitations come from the practical trade-offs between performance and battery life. Especially for sensors of such a small size, enclosed safely in a waterproof plastic enclosure, it is a reasonable thing to do. Otherwise, frequent replacements of batteries would make this type of technology not very feasible for environmental or heath monitoring, where sensors are required to operate unsupervised for long periods of time.

The most valuable lesson learned from this validity study was the discovery that the type of RTLS technology chosen was not very precise. It did expose some limitation (line-of-sight), which were possible to resolve but also it introduced some factors which were not constant i.e. tag's motion settlement time and loss of location code by the tag. Having this knowledge makes one think that precision is only obtainable to some degree. Therefore, rather than being very precise about the events of interest, such as the 10 meter walk test, one has to bare in mind that some RTL systems do not work with the precision of a stop-watch. Hence, the calculation of patient's velocity for the 10-MWT test will not be very precise, however calculating it over longer distance yields better results. As it is explained later on in this thesis, working with inaccurate information does not mean that no information is obtainable. Perhaps some information is not deducible due to lack of precision of the sensing component, however the platform is very flexible in correcting erroneous data by the means of correction rules.

5.2.9.2 Phase 2: Observation-based Validation

5.2.9.2.1 Requirements

The requirements for the RBS2P itself remain unchanged. The hardware and software configuration were the same as described in Section 5.2.6 of this chapter. This time patients were not given instructions but the aim was to unobtrusively observe and record their activities and interactions. Patients who agreed to participate in the study, were asked to wear only one RFID tag around their ankle and one tag around their wrist (on the unaffected by stroke side):

- Stroke patients wore one tag on a wrist band around their ankle and one tag around their wrist.
- Walking aids, chairs and beds were tagged.
- HCPs present on the ward wore tags (one each).
- Locations (rooms) of interest were equipped with RLs.
- All the tags were within the antennas' range.
- The system had storage abilities to permanently save the relevant data for further analysis.
- Patients were not given any instruction but were unobtrusively observed performing their daily activities.
- The PC was accompanied by a wireless router and also run a PHP server.
- The iPad needed to be in the range of the locally-created network in order to load the script and to submit its data to the server.
- PC's and iPad's clocks either have to be synchronised or the time's difference recorded before or immediately after data collection.

As to the manual recording of human observations, similarly to the Phase 1 of the validity study, an iPad was used to log all the relevant information via the use of another PHP script.

5.2.9.2.2 Approach

The next phase of the system's validation was designed to firstly validate the rehabilitation rules, and secondly to provide the means for validation of rules that were not implemented at the time. Due to the fact that the RTL sensors and the lower-layers of the system were found to be reliable and valid, the next step in validating the system was to go up the layers and move on to testing the GEL rules – in this case the rehabilitation rules. The clinical environment and the importance of applying the platform to such a serious problem, demanded that the system's data needed to be confronted with well-established clinical ways of measuring patient's activity. There are various techniques and forms which assist in this process, however there is no one standard which all the units in the UK follow. Out of all observational behaviour mapping techniques (OBMT) that are out there, the decision was made to use a concept called CERISE [15]. CERISE is an unobtrusive scoring system in which every 10 minutes an observation of each patient is made. Patients are scored in the following categories: Activity, Location and Social Interaction – pen&paper version of the CERISE form in Appendix A.1. This OBMT technique was selected due to the fact that it is widely used in stroke units in the UK and it records: location, activity and interaction for each patient with sufficient frequency; and in the absence of cameras, captures relevant information needed for system validation. For the purpose of the project, this tool has been extended to include additional information such as: Equipment and Activity Type. Moreover, rehabilitation models were developed for which diagrams are presented in Section 5.2.8.1 of this chapter.

Manual CERISE data collection, via the use of an iPad application, was performed over the course of several days alongside the running system. The data collector's task was to cyclically walk around the ward and score each patient in each of the categories i.e. record what they were doing, where they were, with whom and what equipment they used. Thanks to this, without the use of cameras (which are forbidden on the ward), a complete picture of all participating patients' activities was drawn. The data was collected for 5 patients on two consecutive days – 4pm-7pm on one day and 7am-4pm on the next day. For another set of 5 patients, the observations were made from 1pm-7pm on one day and 8am-2pm on the following day. In this way, 10 patients' observations were made. Moreover, between the two days, the OBMT data covers the entire day for each patient and hence forms a good sample (as some activities on the ward only take place in the morning and some only take place in the afternoon hours).

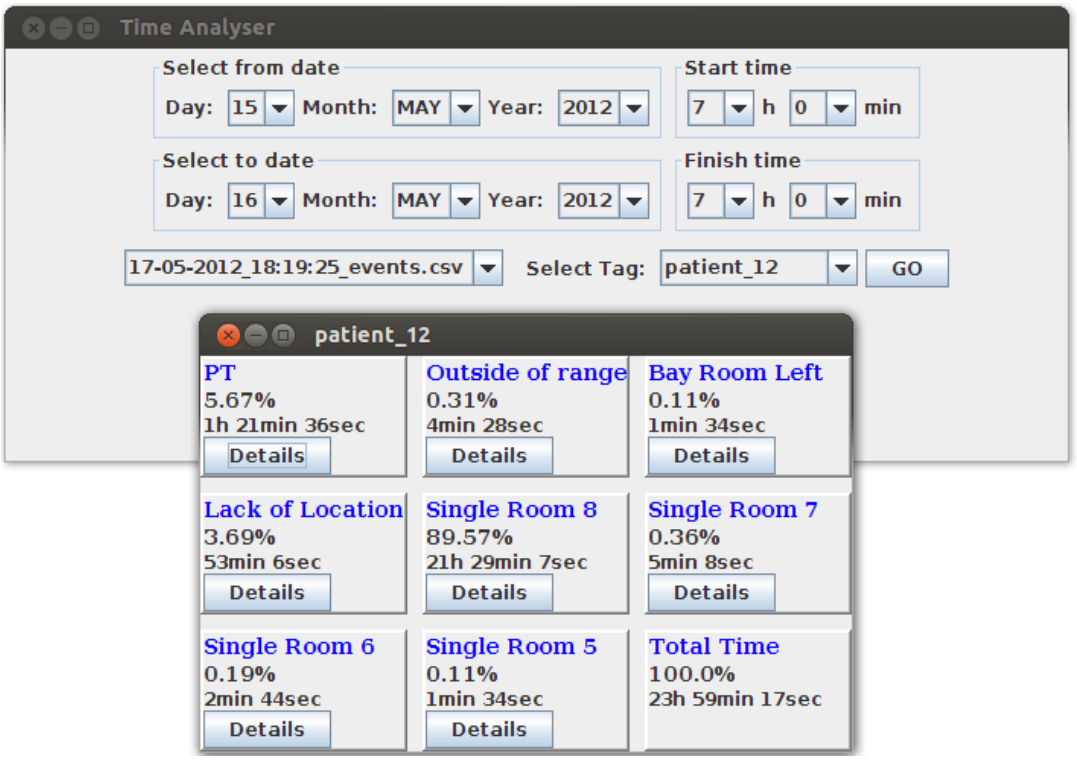


Figure 5.12: Analytical Application: Location Analysis

The validation process has two stages. Firstly, Java application in Fig.5.12 was implemented to display (for each patient) time spent at various locations, broken down further onto proportion of time spent active and inactive (in motion, not in motion readings of the RFID tags). This application enabled the comparison of OBMT data against the system recordings. The OBMT data was simply summed alongside the relevant columns as it existed in the form of CSV files. The second stage of validation involved comparing the system's inferred events against the observed ones. Approaching the problem in two stages results in validating all of the location and activity data (stage 1) and also in validating the particular rehabilitation rules' working (stage 2).

5.2.9.2.3 Results & Conclusion

The results of the 1st stage of the observation-based validation were presented in the UK Stroke Forum in December 2012 [31]. We compared time spent in the physiotherapy room and in patients' own rooms with the Cerise OBMT. The mean times reported for the 10 participating

patients were almost identical to the ones manually recorded via the CERISE technique. Over 12 hours of the OBMT data (720 minutes), the mean difference of 1 minute was found. For the mean times spent in patients' own rooms it was 570min vs 569min, and for the mean time spent in the physiotherapy room it was 49min vs 48min (System vs OBMT times). The PCC value of 0.99 (Fig.5.5) was calculated for both cases, which strongly supports the conclusion that the system is **comparable** to the observational behaviour mapping technique used in stroke units in the UK. Such a significant **correlation at the 0.01 level** proves the system to be valid for correctly measuring time spent in various locations for tagged subjects.

Table 5.5: Results of Phase 2 Validity Study [31]

	Time spent in physiotherapy room		Time spent in own room	
Method	RBS2P	OBMT	RBS2P	OBMT
Mean time in min (Std dev)	49.5 (42.8)	48.0 (39.9)	570.1 (136.3)	569.0 (139.5)
PCC	0.99**		0.99**	
95% LOA/[a]	- 6.7 to + 9.7		- 33.1 to + 34.2	

[a]Bland & Altman 95% Limits of Agreement

Some more results from this validity study were presented at the European Stroke Conference in May 2013 [33]. This time, 5 patients' data was presented in the context of the total time spent in each location, broken down further onto time spent active and inactive. These results also add to the conclusion that this system is able to continuously and accurately measure patients' active time within an appropriately equipped environment.

Table 5.6: Results of Physiotherapy Rule Validation

Patient	Observed by collector	Observed by system	Discrepancies
1	1	1	none
2	1	3	2 (6sec and 22sec long)
3	1	2	1 (3min 54sec long)
4	1	3	2 (6sec and 22sec long)
5	1	1	none
Total	5	10	5

The results of the second stage of this observation-based validity study have not been published to date. The first rule which was tested for conformance is the *Physiotherapy* rule – described in details in Section 5.2.8.2. The confrontation of the system's inferred *Physiotherapy* events against the OBMT data revealed some discrepancies. The presence of more than one

physiotherapist in the physiotherapy room triggered for multiple `Physiotherapy` events to be calculated for the same patient and for the same time period. With a help of a rule this problem was overcome. Once this issues was fixed, the validation process was performed again and gave results represented in Table 5.6. It was found that the system incorrectly reported short Physiotherapy events – due to patients passing by the physiotherapy room on their way to the adjacent occupational therapy room. However, after consulting it with our rehabilitation specialists another rule was introduced to discard events shorter than 10min – as for a session to really take place it needs to be of sufficient length. Finally, when the last rules was in place, the two datasets were confronted again, and discrepancies were no longer found.

The system was tested to correctly identify and reject `Physiotherapy` events. It was not reporting events which did not occur, and was not rejecting events which actually took place. These results form strong evidence that the system is validly recognising `Physiotherapy` events. Similar results were achieved for the *Occupational_Therapy* rule. This rule is analogical to the *Physiotherapy* rule and the only difference is that it matches events against a different location code and different HCP's profession (occupational therapist instead of physiotherapists) – hence detailed analysis of the validation results is not presented for brevity. The *Speech_Therapy* rule also gave positive results, however this rule's logic is different from the two therapies mentioned already. It looks at occurrences of patients seen at either of the three location which lead to the speech therapy office (which is outside the RTLS' coverage) and then for a subsequent occurrence of an event which indicates that the object is out of RTLS' range.

The transportation rules were also successfully validated against the OBMT data. These rules, apart from correctly discovering `Walking` events (subclass of `Transport` Fig.5.9) for patients, also precisely indicated involvement of walking aids. Moreover, the `Transport` events assisted by a wheelchair, got re-classified as/specialised into `Wheelchair_Ambulation` events (also a subclass of `Transport`). Fig.5.7 shows the results of the *Walking* rule validation. Despite the fact that the table consists of some discrepancies, the rule cannot be said to be invalid. All these discrepancies were caused by the fact that system-inferred events were not present in manually recorded observations. The team was surprised to see that the system was reporting these unobserved (by human) events. We examined raw sensor data – with the aid of the

software in Fig.5.12; and found that indeed these events were recorded. It is the conclusion of the team, including rehabilitation experts, that these events did indeed occur and were not observed by the data collector. Human observed data's granularity was 10min and in between two observations patients could unnoticeably (to the observer but not to the system) change locations. The physiotherapy experts were willing to trust the system in these unobserved by human cases that these events did occur.

Table 5.7: Results of Walking Rule Validation

Patient	Observed by collector	Observed by system	Discrepancies
1	3	4	1
2	3	3	none
3	2	2	none
4	3	4	1
5	4	6	2
Total	15	19	4

Overall, all events inferred by the system have been successfully validated against events observed by the data collector. The validity study has proven the system to be able to accurately measure time spent in various locations and to further split this time onto active and inactive, which is very important in the rehabilitation context. Moreover, the knowledge that the system can accurately measure this, enabled the implementation of the rehabilitation rules, which rely on movement and location information. The successful validation of the five rehabilitation rules listed earlier, supports the statement that the system is valid and able to replace the manual observation techniques, which are difficult to execute.

5.2.10 Discussion & Limitations

The pilot study was designed to judge on system's reliability and validity. It started off with two reliability studies and once the system was established to be reliable the focus shifted onto validating the system. The testing strategy can be considered as a bottom-up approach – as the reliability of sensors was considered first. Once it was established that the COTS sensors are reliable, then the study moved on to the validation of the RTL system and low-level functions of

the platform i.e. the application of rules in the DL and SEL layers and platform's persistence. Finally, the GEL rules were successfully validated against hand-recorded observations, which has proven the system to be valid and capable of outperforming human-based observational behaviour mapping techniques (OBMT). The pilot study lasted for a year (24/7 recording) with 172 tags attached to various objects and people. The data gathered during this period is safeguarded, in accordance to the Data Protection Act, and enables further validation of new rules in conjunction with the iPad's collected data. The work is ongoing and further rehabilitation rules are in development. The system allows for retrospective application of rules and hence this rich dataset is of a great value.

Further retrospective analysis of the data will not only allow for new rules to be implemented but may also be of use to the NHS for strategic planning and modification of treatment standards. Moreover, it can be used to compare the efficiency of this particular stroke unit (in terms of patients' average length of stay, amount of treatment they receive, etc.) against others in the UK. According to[5] there are approx.110,000 new strokes and 30,000 recurrent strokes in UK every year. Moreover, stroke is the third largest cause of death in the UK. Around £7 billion a year is spent on stroke and this figure is likely to grow in the future as every year there are more stroke survivors, for whom care is expensive. Due to the fact that OBMT techniques are not practical and rarely used, people who make important decisions, are not fed with highly accurate information of sufficient quality. The application of such system as the RBS2 in the clinical setting has a huge potential to benefit patients, healthcare professionals and the NHS.

In one case, the sensor setup did not permit for development of reliable `Lying/Sleeping` rules. RFID tags on the unit are attached not only to patient, staff and walking aid but also to patients' chairs and beds – to enable inference of interactions with objects of everyday use. During the implementation of the Lying/Sleeping rule patient, bed and chair tags' data was analysed in depth in order to discover patterns indicating getting into and getting out of the bed actions. Rules were implemented to deduce *Comovement* and *Costatic* facts which represent the correlation of patient's motion with objects they tend to use. However, due to the fact that some patients stay in shared bays and some in single rooms the presence of HCPs and tagged rehabilitation equipment (such as hoists, steadies, etc.) could not be considered. The final result

[5]http://www.strokeunitglos.nhs.uk/strokeunitgl280692.html

was rather disappointing. It was difficult to infer when patients were getting into the bed and when they were out of it – and hence difficult to conclude how much time they spent in bed. This was mainly due to the fact that beds, chairs and walking aids can be moved by anybody – nurses making beds, visitors, patients, etc. The fact that some patients were in shared rooms did not allow to use a presence of some HCP on rule's RHS. Moreover, there are so many ways in which patients can get out of/into their bed – on their own, using their walking aid, using specialised equipment operated by the nurse, etc. – that correlation of patient's movement with the presence and movement of other objects was difficult. However, it is anticipated that this rule can be implemented by approaching it in one of two ways. Firstly, checking the frequency of bed's movement against the frequency of movement of other objects that belong to the patient (chair and walking aids) could work. Secondly, upon completing all CERISE rules which infer in-room activities, this rule could work under the closed world assumption (CWA). Its logic would be as follows: if the patient is not known to be doing anything else in the particular time interval and they are in their room then they must be lying/sleeping. An alternative solution, which would not work with already collected data set, would be to equip beds with bed pressure sensor pads[6] instead of the RFID sensors. These sensors would make `Lying/Sleeping` rules very trivial to implement.

Use of other sensors such as cameras was not allowed in the stroke unit, and hence no visual data was captured either to be a part of the system or to provide 'ground truth' for validation. Hence, RTLS technology was used for data collection, which only reported on location and motion of tagged objects/subjects (motion not very accurately). Such a low-level information made the process of rule implementation and validation very challenging. Despite these limitations the implemented system, fed with very primitive information, produced useful, actionable information. In this setting, the RBS2 system was not capable of inferring the following activities:

Dressing and Hygiene: Impossible to implement without attaching tags to clothes, toothbrushes and/or combs – which is not very practical.

Autonomous Exercising or Training Not implementable for exercises which do not involve any equipment e.g. squats.

[6] http://www.quickmedical.com/smart-caregiver-bed-sensor-pads.html

Communication Not achievable with RFID technology. Possible solution would involve microphones (voice recognition) or cameras – both sensors not allowed in hospitals.

Leisure For the same reasons as above.

Apart from limitations coming from the sensor technology used, there were other factors which limited system's capabilities. Firstly, for privacy reasons, toilets were not equipped with room locators which does not make the stroke unit a fully 'closed world'. Secondly, visitors and visiting staff were not tagged and hence a presence of 'ghosts' needed to be considered. Moreover, one patient was under a threat of committing a suicide and hence the room locator from their room had to be removed – something which did not stop the data collection process but ruled out certain rules to fire for this particular patient. Even though fever data was available for this patient, the system was flexible enough to provide some information for incomplete collection environment.

It needs to be accepted that some events are impossible to deduce. Either due to privacy considerations, such as: lack of room locators in certain areas, restrictions on the use of cameras and microphones; or due to the lack of particular information/sensors. Therefore, even though some rules have been successfully developed via interactions with healthcare specialists, refinements also have limits and not every activity is recognisable. Some events are simply impossible to infer with the technology used.

With all the imposed barriers and limitations the project has met most of its requirements already (further development and applications for funding are considered) apart from the last bullet-point of the Req. 5 "During the course of the day, a given patient has spent X hours lying in their bed...". This requirement was planned to be met but proved to be very time consuming and hence the focus was shifted to other areas. However, having all the data available and having identified two possible strategies, this functionality is achievable with enough effort put into it.

Having satisfied physiotherapy experts that the system was producing useful information, it was set up to run permanently 24/7 for just over a year. 172 tags were deployed on the unit (attached to beds, chairs, walking aids and worn by HCPs) and longitudinal data has been collected for 52 patients. In-depth micro level analysis satisfied the team that the system was functioning

adequately. Subsequently, based on deep-detail reliability and validity studies rehabilitation specialists decided to trust the system and agreed that it produced awful lot of useable information. The system-collected dataset (1 year of data) feeds Arshi Iqbal's longitudinal healthcare study.

5.3 Home Monitoring Pilot Study

The Home Monitoring (HM) second pilot study was a one year project sponsored by the National Institute for Social Care and Health Research (NISCHR[7]), for which the research group applied for funding in the autumn 2012. It was commissioned partly on the strength of early success of the hospital study. Funders have considered it positively because they thought this approach seemed promising. The Home Monitoring project run from October 2012 till October 2013. The main idea was to take the RSU equipment to patients' homes and to measure their activity. For this study patients, who consented to take part in the RSU study, were approached first, so that a complete record for them could be created and analysed. From computer science point of view, the challenge was to take a system embedded into one environment (RSU) and to adapt it to a completely different environment (home). Obviously, due to the huge differences in the two environments, not every activity measured at the RSU was measurable at home e.g. there is no physiotherapy or occupational therapy offices or HCPs at home, so the `Physiotherapy` rule did not apply to home environment. The rehabilitation ontology was extended to include subclasses for `Home_Activity`, which in turn is a subclass of `Activity`..

5.3.1 Introduction & Motivation

The full title of the HM project was: "Measuring activity recovery at home in the post-acute phase of stroke rehabilitation by means of a novel computerised system" . The rehabilitation specialists need tools to unobtrusively measure patients in various environments. Many studies have shown that the rate of stroke recovery at home is higher than at stroke units. Why is it that patients surrounded by healthcare professionals, getting all the care they need (in theory),

[7]http://www.wales.nhs.uk/sites3/home.cfm?orgid=952

do not recover as quickly as when they get discharged to home? The HM project allowed collecting data for patients who were part of the RSU study and hence enabled to compare the two environments. In the reminder of this section I will not be reporting on the clinical findings – as these belong to the rehabilitation research field – but rather on how the system was adopted to the new setting and all the implications and solutions that followed.

The project aimed to answer the following research questions:

1. Can functional activities of people recovering from stroke be measured reliably and validly at home on a continuous basis using unobtrusive sensor technology?
2. What are the main characteristics of recovery in the home environment and what are the main differences compared to recovery in the stroke unit?

These questions led to the hypothesis that “Activity and functional recovery can be successfully measured (in terms of reliability and validity) on a continuous basis in the home environment". In order to test this hypothesis, the following aims of the project were agreed on:

1. Validity Study in Home Environment of an Automated Sensor-Based System.
2. Developing Analytic Tools & Testing for Activity Recovery in the Home.
3. Exploratory Study to Answer the Second Research Question.

Based on these aims project the plan was agreed and the study run for 1 year. The lay summary submitted for the NISCHR application gives more insight into the objectives of the project: “The aim of this study is to further develop a computerised system for detecting functional activities related to mobility of stroke patients in their own homes. We have already developed such a system for use in a hospital environment (Regional Stroke Unit). To the best of our knowledge, such a system for continuous activity measurement does not yet exist. Because rehabilitation at home is becoming more important we need to develop tools that permit us to explore recovery of movement in both the hospital and home environments. Setting up our system at home brings many new challenges. Therefore before we can explore how early discharge home with follow up rehabilitation can best be undertaken, a preliminary study providing the

basic information on how to carry out such measurements at home is essential. On completion of this study we will have a system for such measurements at home and will have established its reliability and validity. On this basis further grant applications will be developed to study this area of home-based rehabilitation. This will provide us with opportunities to study what type of exercise approach should be used and how adherence to exercise can be promoted."

5.3.2 Requirements & Approach

For the HM study 11 patients were selected and for each of them the system collected their mobility data for 2 weeks. Patients were asked to wear two RFID tags – one on the wrist and one on the ankle. In addition, patients were asked to wear a Step Activity Monitor (SAM[8]) – small device, worn on the ankle, measuring number of steps taken. This device is a 'gold standard' in the field of rehabilitation and it was used in the study to record patients' steps. The hardware setup was the same as in the RSU study but only one antenna for the RFID reader was used to provide the coverage for the entire home. All the software was loaded onto the Mac mini (Intel Dual-Core i5 2.3GHZ, 4GB RAM, 750 GB HDD) headless PC running Ubuntu 12.04LTS. The main reason for selecting the Mac mini for the study was that the system should be able to run unsupervised without attracting subject's attention. Moreover, it takes very little space and thanks to running Ubuntu, all servers (ZM, GSN) and other software were compatible and easily tuneable. The remote log in feature was used to connect to the PC in order to export the collected DB data into a CSV file (one click on the user interface – Fig.5.13) and then to a pen-drive. As for RTLS hardware, the room locators were only placed in living room and kitchen. RFID tags were attached to all walking aids patients normally use and to other equipment such as stair lifts.

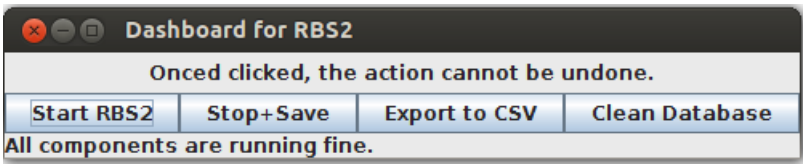


Figure 5.13: Controlling the System: Dashboard

[8]http://orthocareinnovations.com/pages/stepwatch_tradesystem

The HM requirements were compatible with the RSU ones. However, due to different environment and different objectives of the project, the aim was to measure very basic information. The high-level requirements for the project were as follows:

- Detection of patient leaving home.
- Detection of the amount of movement patient did every day.
- Interactions with walking aids (if any were used) and with equipment such as stair lifts etc.
- Detection of the proportion of active and inactive time spent in living room and kitchen.
- Comparison of SAM data against system recorded movement data.

5.3.3 Home Monitoring Ontology

The home ontology has been developed in cooperation with domain experts. Fig.5.14 represents activities which patients perform at home and are of interest to HCPs. The URL to the home rehabilitation ontology can be found in Appendix B.

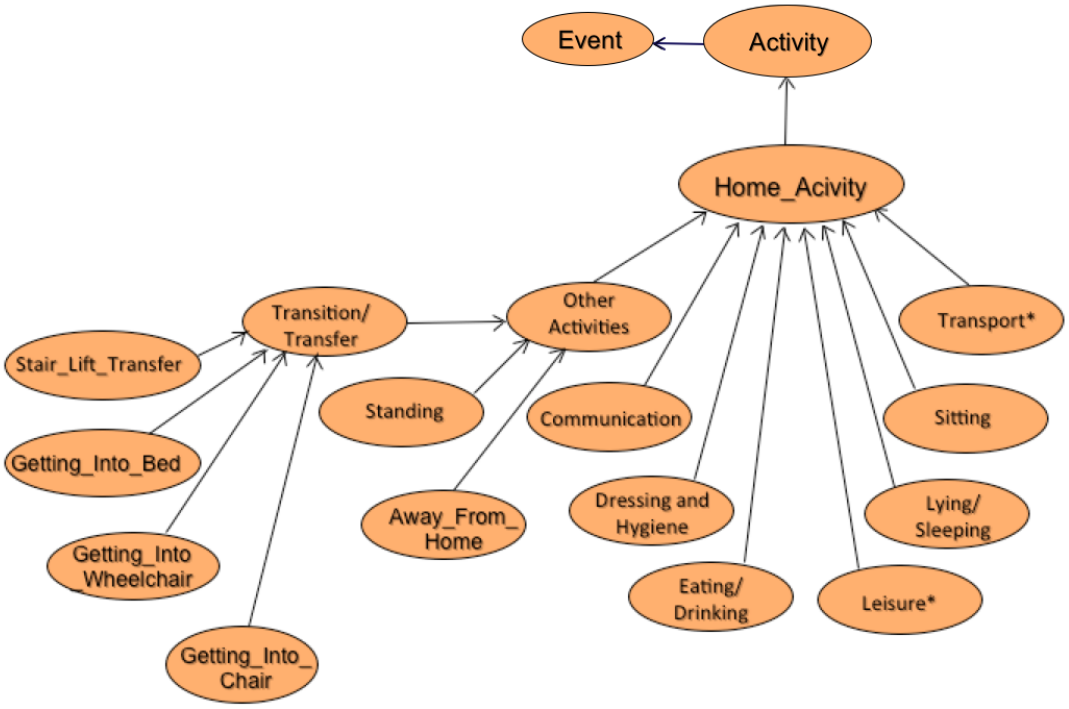


Figure 5.14: Home Activity Classes

5.3.4 Results

The results of the Home Monitoring project will be published in the Journal of Neuro-engineering and Rehabilitation, in collaboration with domain experts. Presenting the clinical findings here is beyond the scope of this thesis and hence the focus is on the Computer Science contribution and data rather than clinical contribution or importance of these findings. Moreover, some content of this article is owned by our collaborator in the School of Healthcare Sciences, Cardiff University and presenting it here would be academically unfair. Our collaborators have been provided with software tools and data, and therefore they own their clinical findings.

Table 5.8: System vs SAM Comparison

Patient	System		SAM
	wrist	ankle	ankle
1	43.4%	22.46%	26.67%
2	6.92%	6.88%	5.5%
3	17.62%	20.14%	22.37%

The first stage of this project was to compare of data collected by the system against SAM's data ('gold standard in physiotherapy') – establishing its validity. Due to the fact that the RFID-fed system was not capable of measuring steps taken, the active/inactive times were compared – directly available through SAM's analysis software. The analysis of the results (Table 5.8) for three patients revealed that the system is usually slightly underestimating the percentage of active time for the ankle tag (2 out of 3 cases) but the two figures are very close. For the wrist tag confronted against SAM's data, in 2 out of 3 cases the system overestimated patient's movement – one of which is almost double the SAM's figure. These findings were expected. Two devices worn on the same limb, should in theory give similar results and this was the case here. When comparing SAM (worn on ankle) and RFID wrist tags the differences, as expected, were bigger. Wrist movement does not necessarily have to be associated with walking and hence the discrepancies. Overall, the percentage of active time reported by SAM correlated well with RFID tag worn on ankle, which concludes that the system is valid for measuring active time when RFID tag is worn on ankle.

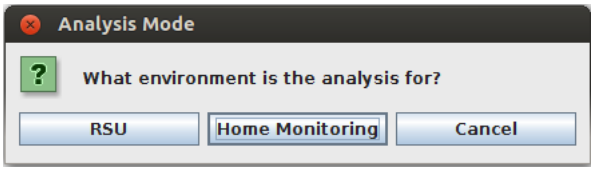


Figure 5.15: Analytical Application: Mode Selection

The next stage was to address all the requirements stated at the beginning of the project. All these objectives were met through the use of the analytical tool developed for the RSU pilot study, which was extended to meet the new requirements. The 'Mode Selection' frame allows for selection of analysis mode – Fig.5.15. After selecting the mode, user is presented with exactly the same interface as in Fig.5.12. The only difference is the unification of location codes, so that in the HM mode the activity data is broken down onto the following locations: Living Room, Kitchen, Outside of Range and Lack of Location – as represented by Fig.5.16. Each location has a 'Details' button which displays a frame with all occurrences of the patient at this location. Therefore, the first requirement of detecting patient leaving home was satisfied by counting the number of mobility traces in the 'Outside of Range' location. The amount of movement in each location is displayed just above the 'Details' button and allows to determine

the proportion of time spent at various locations, broken down further onto time active and inactive. The interactions with walking aids can be discovered by selecting walking aid tag on the initial user interface. The lesson learned from this project was that the system could be brought from one environment to another with minimum programming effort. Neither the software configuration, nor the hardware changed. The same tools were used for data lookup – with very little modifications.

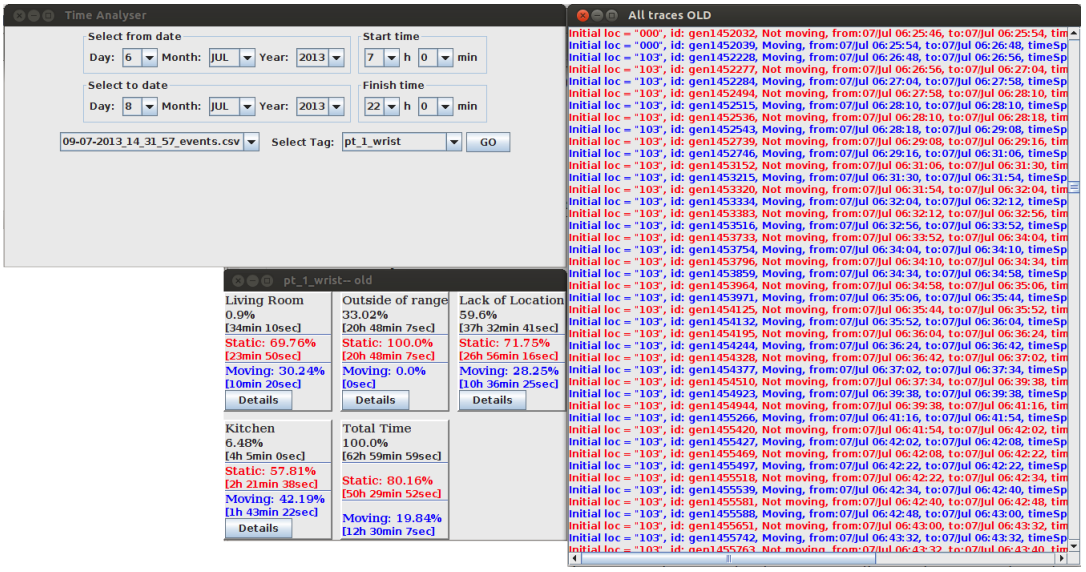


Figure 5.16: Analytical Application: Location Breakdown

5.3.4.1 Ontological Comparison with the RSU

The usage and importance of ontologies in the system was partially discussed and it serves many purposes. Thanks to it, the system produces semantically rich data and it enables validation and verification of the system – Chapter 6. Another advantage of formally representing knowledge in the system is event equivalence. If one wants to express that e.g. the *Away_From_Home* event is similar to *Away_From_Hospital* to enable comparisons between the two environments, they simply need to capture this knowledge in the ontology. In OWL one can use the built-in property *owl:equivalentClass*; in RDFS class' equivalence (e.g. A = B) can be expressed via the *subClassOf* property e.g. *subClassOf(A,B)* and *subClassOf(B,A)* imply that A and B are equivalent. In OBO, on the other hand, one can use the *synonym* property of classes to express their equivalence. Capturing the equivalence of classes in various ontology

languages is achievable via the use of appropriate class properties. Therefore, the platform can benefit from this feature by producing inter-linked semantically rich events. Thanks to this, a task of comparing patient's performance in two very different environments becomes easier.

This chapter introduced hardware used in the two pilot studies and their motivation and requirements. Rehabilitation models were introduced together with problem-specific rules. It presented the reader with the chosen approach and results of the reliability and validity studies. The Home Monitoring second case study and its preliminary results were presented. The next chapter looks at validation and verification of the RBS2P with the aid of: data instances, rules and ontologies.

Chapter 6

Validation & Verification

This chapter explains to the reader how the rule- and model-based approach aids validation and verification of the implemented platform. The validation and verification procedures are introduced in detail and results of these strategies applied to the implemented/live system are shown. The chapter concludes with the summary of benefits that implemented validation & verification strategies provide.

6.1 Verification

Maintenance of system's integrity is an important aspect of any system [51]. Once a system has been built it needs to be maintainable. Verification is an important aspect of software engineering as it helps to establishing whether the system works as the client/user envisioned. Especially in fast evolving sensor-based systems (evolving either at the bottom – sensors, or at the top – requirements) maintenance of system's integrity is highly important. Having established that maintenance is important, here is how verification cross-checking can help in the proposed platform. The verification of the system is performed by cross-checking of rules, models and event data – inconsistencies between these components are found. The focus in this thesis is on the type of incoherence between models, rules and (sensor) data. In this process there is no logical analysis of rules, however there is other research work which looks at deep analysis of validation and testing of rule-based systems e.g. [36]. Logical analysis of rules is not in the scope of this thesis but it could be a promising area for future work to provide even more in-depth verification at the logical level. The selected approach increases the transparency of the system and allows to see if it is properly 'joined up'.

6.1.1 Checking for Consistency, Coherence and Redundancy

A part of the system's cross-checking for coherence is to test it for redundancy and deficiency cases. As already mentioned, this is achieved via the KB analysis of the RBS. The redundancy discovery is simply a process of finding **rules in the KB which never fire**. It is then up to the developer to interpret these findings correctly. There is more than one possible reason for a rule not to fire. Firstly, it is redundant either due to the fact that other rules are superior to it, or it simply exists in the system by mistake. Secondly, the rule might be important to the problem but never fires due to lack of inputs. In this case, the developer would need to investigate why the flow of information is blocked. Finally, there might be sufficient inputs however, they do not satisfy rule's LHS conditions and hence the rule never fires. The deficiencies are also discovered via the RBS analysis by highlighting **facts for which there are no rules**. These cases should also be alerted to the developer, as they might be very serious. Either the system lacks rules which ought to exists but do not, or certain facts, which should not be present in the WM, somehow found their way into. Regardless of the reasons, these cases should be raised and dealt with in order to keep the system coherent.

Verification Procedure Pseudocode

Redundancy (rules which never fire):

1. Compare events produced by the system against rules' outputs.

Deficiencies (data but no rules):

1. Get all events produced by the system.
2. Delete those which are used as inputs in existing rules.
3. Delete goal events as they are the end product.

Consistency Checking

1. Create a set (A) out of all events produced by the system.
2. Create a set (B) of all event classes in the domain-specific ontology.

Events with no corresponding classes:

1. From set A delete every event present in set B.

Classes with no corresponding events

1. From set B delete every event (& its ascendants) present in set A.

Events for specialisation

1. Keep events which are present in both: set A and set B.
2. Delete events which have no descendants.

The consistency checking, on the other hand, is achieved via the analysis of domain ontology against the running system – in more details against the Events table. Inconsistencies are found by discovering **events for which there are no classes** in the ontology. These cases should be dealt with quickly, as it either means that the ontology is incomplete or that the system is inefficient as it produces irrelevant information. In some cases, the incorrect spelling may be the cause for this and only cosmetic corrections are required for the system to be consistent again. Another angle to checking consistency is discovery of **classes**, in the ontology, **for which the system does not produce any events**. This might mean two things. Either the system is incomplete, as it does not infer all information captured in ontology (rules don't exist or are incorrect), or these classes are simply not considered (beyond project's scope) and should be ignored. Regardless of reasons, it is beneficial for the developer to be aware of system's incompleteness. Finally, the last step in testing for system consistency is to suggest a **specialisation of classes which have descendant classes**. For example, if the system produces `Transport` events in the RSU setting, the user may want these events to be specified either into `Walking` or `Wheelchair_Ambulation`. Therefore it is desirable for the system to report these cases and it is up to developer to handle these. The extended system diagram, which includes the verification module, is shown in Fig.6.1.

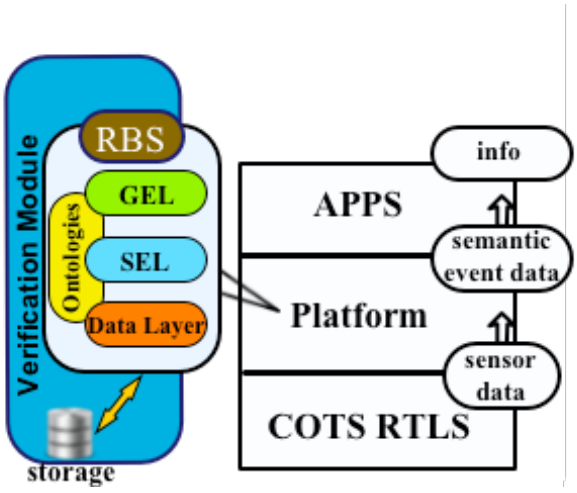


Figure 6.1: Extended System Diagram

6.1.2 Implementation of Verification Procedures

The coherence check consists of two elements, so called: 'rules which never fire' and 'un-used inputs'. The list of rules which never fire are simply those which are never activated. There are more than one way of solving this problem. One solution would be to crawl the DB of events, pull each record and create a set out of it. Then the next step would be to, for all rules, analyse their outputs against the set of events pulled from the DB. However, there are two problems with this approach. Firstly, crawling the DB consisting of thousands or even millions of events would be very slow and inefficient. Secondly, multiple rules may produce the same events and matching events against rule's outputs would simply be impossible. Another approach is to look at `EvidenceFor` facts that are in the WM. However, the problem with this approach is its inefficiency, as all the instantiations of `EvidenceFor` facts would need to be looked up in order to create a set of rules associated with these facts. Ideal solution would be not to crawl all the instantiations of some classes (`EvidenceFor, Events`) but to somehow maintain a record of rules' firing. In Jess, this can be achieved via use of the `(store <string or symbol> <expression>)` function, which does not slow the rule's execution down noticeably as it is an internal function. This function can be called on every rule's RHS with two arguments – the first one is the name under which some content is stored (in this case it is the name of the rule) and the second is the value to be stored (in this case the value is irrelevant but is set to the name of the rule too). Even if Rule_A fires 100k times, only one instance of Rule_A record is stored in the working memory (it might be updated 100k-1 times but only one instance of it exists for each rule). Therefore, the list of all rules which persist in the system can be confronted against rules' firing records pulled from the WM via the use of the `(fetch <string or symbol>)` function. In this way, a list of rules with no firing records is populated and can contribute to the developer's report. Below is the sample of the output of this routine:

```
<rule>
   <rule-name>Rule_5a</rule-name>
   <file-path>bin/scm6prw/dl_and_sel_rules/comovement.clp</rule-file-path>
</rule>
```

All outputs of the coherence and consistency checks are produced in the XML format, so that all the elements can be resolved against the XML schemas or DTDs.

The un-used inputs are harder to compute than the earlier case. The logic for this problem is as

follows: populate a list of all events produced by the system and confront it against inputs of all rules. The first part of this solution can be accomplished in many ways. The DB of events can be crawled to create a set of class instantiations – again an inefficient solution and incomplete as not all events are stored in the DB, as some only live in the WM to support the inference of others i.e. `ProbabilityOfFailingDetection`. If not the database, the only other part of the system which holds event data is WM of the RBS. However, the analysis of fact instances in the working memory would again not produce a sound solution. Firstly, because facts from the WM are periodically dumped to the DB. Secondly, this solution again would involve looking up all facts in the WM (efficiency issues). The best approach seems to be to look at the fact templates defined in the RBS' working memory. Rule-based systems, such as Jess, often provide useful functions to aid rule-base analysis – such as `listDeftemplates()`. This method returns a collection of all the deftemplates ever used in the rule engine, both explicit and implied. From this list of all fact deftemplates, facts which correspond to classes in the goal ontology (in the case of the pilot studies the Rehabilitation ontology) need to be removed. This is done to make sure that the events of interest/end products are not reported as un-used inputs. Finally, these findings can be confronted against the set of inputs of all rules – these can be computed via analysis of every rule in working memory (`listDefrules()`). This solution is very quick and efficient, and below is the sample of the produced output:

```
<fact>
   <fact-name>EvidenceFor</fact-name>
</fact>
```

Worth mentioning is the fact that the coherence check is performed by analysing the running system (its knowledge base) against its ontologies. It does not involve the analysis of the database as this would have been inefficient and also would not provide a complete solution. All the above have been tested on the running system and produced valid results.

The consistency check has three parts to it: 'events with no classes in the goal ontology', 'classes with no events' and 'classes to be considered for specialisation'. These cases need to be resolved by analysing events DB. Firstly, because database holds all 'outgoing' events of interest (goal events) – the only part of the system visible to the outside world. Secondly, if all deftemplates were to be pulled from WM, this would produce an unsound result, as some facts are used internally by the system to assist in various routines, and hence do not need to exist in

the ontology. Moreover, presence of a definition of a fact template in WM does not mean that the system actually produces these. Therefore, a reasonable solution to this problem, although inefficient, is to analyse what the system produces i.e. the content of the DB. Once the set of all classes (set A) which are instantiated and get stored to the DB is populated, the ontology needs to be analysed. This is achieved by mapping the goal ontology into a tree structure. From this tree structure a set of all classes can be easily generated (set B). Hence, events with no classes in the ontology is the relative complement of B in A (A \B). Classes with no events is the relative complement of A in B (B \A). It is worth noting that all parents (and their parents, etc.) of every event which exists in the DB need to be removed from the results i.e. an existence of a `Walking` event in the DB means that its parent class `Transport` has been instantiated in the form of its descendant. Finally, the intersection of A and B (A ∩ B) is the set of classes which are to be considered for specialisation if they have at least one descendant in the ontological tree. Overall, reporting on consistency issues involves the analysis of events produced by the system (stored in Events table) against the goal ontology. Even though inefficient in one aspect (crawling DB) it is a sound solution which solves the problem in a correct way. The working of the above solution has been tested on the instance of a running system and produced valid results, sample of which follows.

```
<events-with-no-classes>
   <event-name>Away_From_Home</event-name>
</events-with-no-classes>

<classes-with-no-instantiations>
   <class>
<class-name>Standing</class-name>
<id>0000077</id>
<namespace>Rehabilitation</namespace>
   </class>
</classes-with-no-instantiations>

<classes-for-specialisation>
   <class>
      <class-name>Physiotherapy</class-name>
      <id>0000057</id>
      <namespace>Rehabilitation</namespace>
      <descendant>
         <class>
            <class-name>Group_Physiotherapy</class-name>
            <id>0000082</id>
            <namespace>Rehabilitation</namespace>
         </class>
         <class>
            <class-name>Individual_Physiotherapy</class-name>
            <id>0000081</id>
            <namespace>Rehabilitation</namespace>
         </class>
      </descendant>
```

```
   </class>
</classes-for-specialisation>
```

The above XML data is not the expected end result (in terms of visual presentation) but rather a structured content which feeds the developer's report. The example of such report can be found in Appendix D.

6.2 Validation

Every system is created for particular end-user(s) and hence should be developed in consultation with them. "*Verification* is the process of checking whether the software system meets the specified requirements of the user" and "*validation* is the process of checking whether the software system meets the actual requirements of the user" [51]. In the validation process the system was made transparent to rehabilitation experts by visualising the classes of events it produced. There were two rounds of consultation where the proposed approaches were critically evaluated by the three members of our group with medical/rehabilitation background. These prototypes were designed to show that the user can engage with the system and understand what is inside it.

6.2.1 Designs & Feedback

In order to help assess the value of the ontology and of the rule-based approach, a set of user interface (UI) sketches was presented to the clinical members of the research group in one of the meetings. The three members of the group with rehabilitation background were asked to comment on the appearance of the presented interfaces, system functionality and possible shortcomings. After an extensive discussion UI sketches were modified, in accordance to the received feedback, and took the shape presented by Fig.6.2 and Fig.6.3. In this process, presence of the ontology in the system proved to be useful providing means for automatic exposure of events the system produces and of those which exist as a concept but are not instantiated. Secondly, the structure of the *Event model* (EO) provides a means for querying of data and its filtering.

The first discussion regarded the login screen which would precede the UI in Fig.6.2. There is more than one type of anticipated end-user and hence a login screen is important. The login credentials would determine the access levels of the individual based on their profession. Researchers and managers would have full access to the entire data and also ability to see, not only individual patient reports, but also combined statistics for all patients on the unit, filtered by various categories e.g. gender, age, length of stay, etc. These users, apart from raw individual patient data, need the bigger picture of the unit in order to carry out their tasks and to make decisions. Another class of users are healthcare professionals, who work with patients on daily basis. They are mostly interested in how individuals they look after perform. Therefore, after logging in, their interface would have certain categories blocked or invisible in order to provide them with a quick and convenient way of acquiring the relevant data. Moreover, there might be some data which the hospital's management would not like their staff to have access to. Therefore, the role-based access was identified as an important feature as it allows for controlled system transparency.

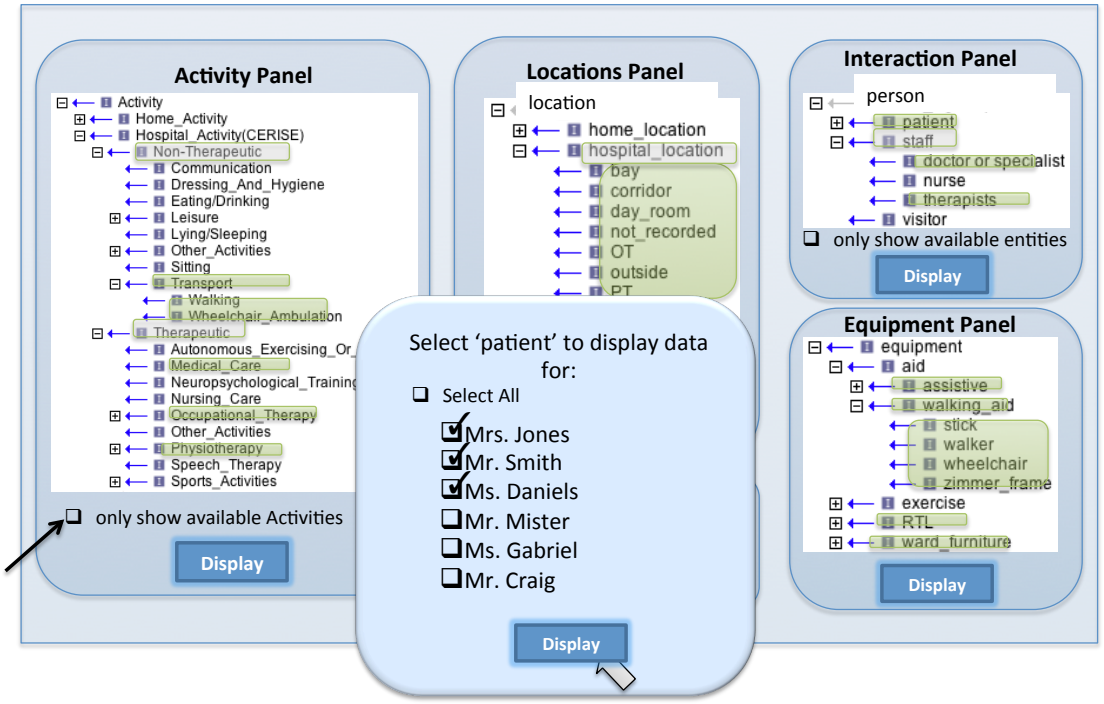


Figure 6.2: UI Design: Patient Select

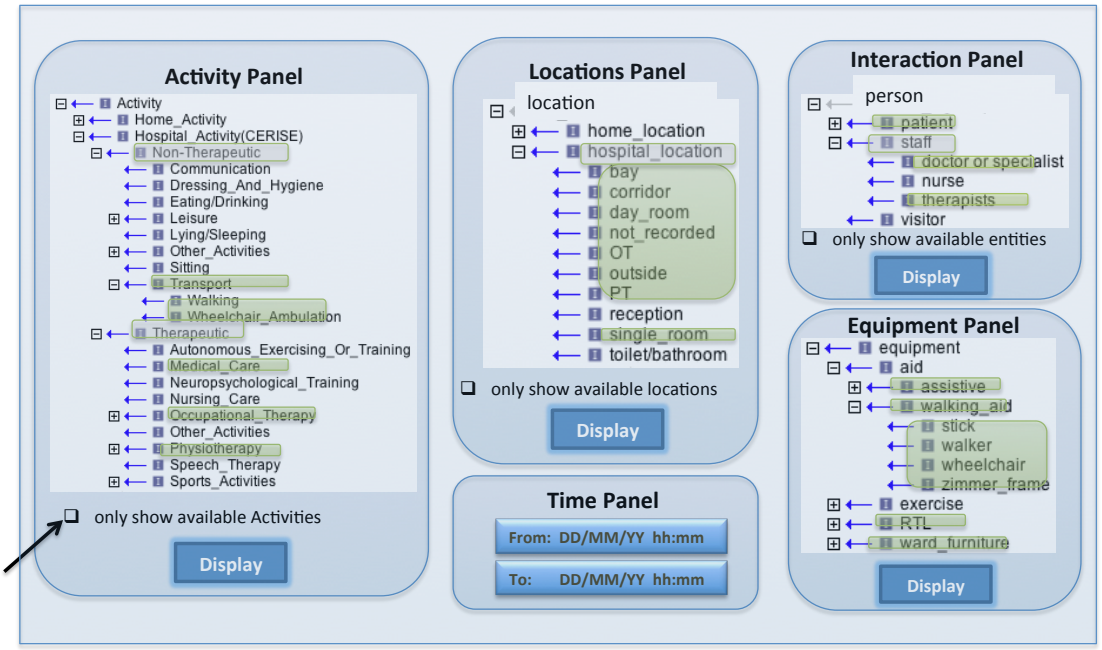


Figure 6.3: UI Design: Data Select

After logging in, agent (in this example patient – Fig.6.2) selection panel is presented to the user. The list of agents is compiled, based on the content of appropriate table in the DB. Once observables (patients) of interest are selected , the user is taken to the UI which is directly based on the goal ontology feeding the system – Fig. 6.3. All the panels on this interface are based on the attributes of the *Event model*. Hence the `Activity` panel corresponds to the name of the event; the `Locations` panel corresponds to the *atPlace* attribute; `Interaction` panel to the *involvedAgent*; `Equipment` panel to the *involved* and `Time` panel to the *atTime* slot of the EM. Available events i.e. those which are produced by the system, are highlighted in green and the unavailable ones are white. The white frame is drawn around classes for which not all of their descendant classes are available. Moreover, a tick box in each panel allows to hide unavailable categories.

Let's consider a situation where the user wants to select `Transport` events from the `Activity` panel, at all `hospital_locations`, assisted by any type of the walking aid (Equipment Panel), involving any staff. This selection would compose a query to be executed on the `EVENT` table. In return to the issued query, the user (based on their role/level of access) would be presented with relevant event data. In the domain of rehabilitation monitoring, researchers and management

would be presented with the data fit for their purpose i.e. individual patient reports and also a visual display of certain population's data in the form of graphs. These tools (Fig.6.4) would enable them to see the average (for entire unit) for the selected activity, and where the given patient is in relation to the mean values.

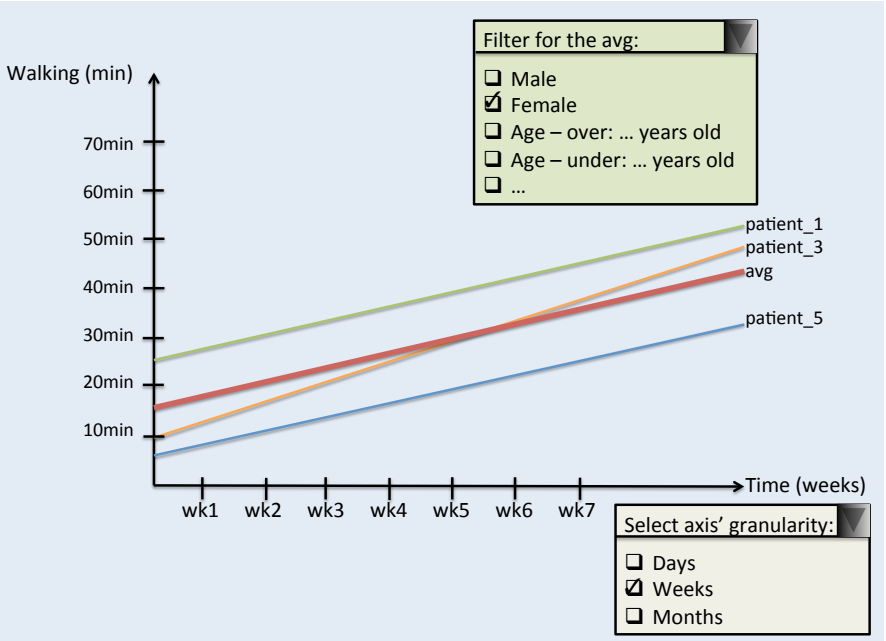


Figure 6.4: UI Design: Activity Graph

6.2.2 Results of the Validation Process

The team agreed that the *Event model* made events, which the system is capable of inferring, transparent to end-users. They were able to quickly and easily identify which (rehabilitation) events – out of all defined classes of rehabilitation – the system was capable of producing. The structure of the EM made them think it terms of querying the data i.e. things (such as involvement of walking aids) they could query the system for. Moreover, experts identified two additional features. Firstly, the need for a role-based access. Secondly, that a list of *involvedAgents* (which appear in events in the Event table) should also be presented for filtering – the system exposes human agents for which data is held. Overall, such high level of system's transparency allowed rehabilitation experts to understand and validate the system. Hence, the presented approach to validation easies the cooperation between IT specialists and domain ex-

perts by exposing system capabilities directly to end-users.

This chapter introduced validation and verification strategies developed in this project. Theoretical verification strategies were backed up by real outputs of the running system. Subsequently, the reader was presented with the validation process which was carried out in consultation with domain experts. The next chapter concludes the work presented in this thesis and gives pointers for future work.

Chapter 7

Conclusions & Future Work

7.1 Conclusions

This thesis introduced a taxonomy for sensor network middleware (SNM). Solutions which are closer to sensors i.e. handle sensor data well but do not make application development easy are classed as low-level SNM. On the other hand, those approaches which make application development easier but, either assume that the sensor data is simply available, or are bound only to one specific type of low-level SNM are classed as high-level SNM. The ideal solution lies somewhere in between i.e. is open at the bottom – to new **sensors** and SNM; and remains open at the top – supports development, maintenance, deployment and execution of sensing based **applications** (definition of SNM [58]). Moreover, as pointed out in [13] there is no perfect solution to the problem of activity recognition, as it all depends on the type of recognised activities and application's requirements. Many solutions in this space are complementary and it is impossible to claim that one is superior to the other. Instead, suitable approaches should be considered to work jointly.

The thesis introduced a set of *layered event models* 4.2.1, which are based on the *Event model* (EM). The EM sets the basic structure for all sensor-captured events. The Real Time Location (RTL) model builds on the EM, and so do domain-specific models. This layered structure brings the following benefits. Firstly, it separates the hardware related concepts from the domain concepts. Bringing new types of sensors to the existing setup only requires the lowest-level model to be extended. On the other hand, adaptation to a new domain does not require any changes at the lowest level (sensors remain the same) but instead requires a formulation of domain's knowledge at the upper layer i.e. domain specific model. Machine learning techniques that are

often used to solve the problem of AM, are not adaptable to changing requirements or changing sensors and require laborious re-training [13]. Moreover, these solutions are not transparent. In opposition to these techniques, the *layered event model* approach is an effective way of solving the problem of activity monitoring. It provides openness at the bottom – to new sensors, and at the top – to changing requirements, new applications and new domains. Moreover, the *evidencedBy* property of the EM supports explanation of reasoning i.e. provides some degree of transparency – not a black box solution.

The proposed system architecture is based on the EM and rules. Low-level events (LLEs) represent atomic sensor data and high-level events (HLEs) represent complex events. Both classes of events follow the structure of the *Event model*. The role of rules is to assist in the process of events aggregation i.e. they fuse LLEs into HLEs. Moreover, HLEs can be re-used to form HLEs of even greater complexity – as shown in Fig.3.2. In this process, very simple sensor data evolves to meaningful event data of high complexity. In addition, HLEs inferred for one purpose can be re-used to form an answer to a different query i.e. already derived events do not need to be formed again. All these arguments supports the conclusion that rule-based approach is an effective way of using the *layered event model* and an effective way to generate complex events (HLEs) from LLEs.

The proposed rule-based approach based on the *Event model* fully addresses six out of ten identified challenges for WSNM: *abstraction support, data fusion, resource constraints, application knowledge, programming paradigm* and *adaptability*. Another three challenges: *dynamic topology, scalability* and *QoS*; are handled to some extent as the responsibility in satisfying these also lies on the low-level SNM's side. *Security* challenge has not been addressed by this research at all.

The Rule-Based Semantic Sensing Platform (RBS2P) was designed to provide a bridge between commercial off-the-shelf RTLS and applications in the domain of activity monitoring. In this platform, the *layered event model* and rules are separated between three processing layers: Data Layer (DL), Sensor Event Layer (SEL) and Goal Event Layer (GEL). The DL interfaces with low-level SNM and injects translated event data into the system. The SEL merges LLEs with other LLEs and include logic for correcting erroneous sensors' data and for data redundancy. The GEL applies domain knowledge to infer HLEs of user's interest. The data aggregation lo-

gic is explicit to the SNM and is spread across the three system layers, which sit above and are decoupled from the low-level SNM(s) feeding the system. This layered approach provides even more benefits. The low coupling of the platform easies the process of adding new functionality to the system, improves orientation in the code and its testability, and easies its implementation, maintenance and deployment. Moreover, to increase the performance (or overcome single computer's physical limitations), layers of the system could be hosted on different machines. A good example of layer separation is the regional stroke unit case study presented in Chapter 5. The clinical environment is very strict and does not allow external access to its network. Therefore, only the DL layer was present on the on-site data collection PC. The other layers (SEL & GEL) were loaded onto the off-site machine to which collected data was transferred. Moreover, the RBS2P exposes semantically enriched goal events. This is achieved through the presence of problem-specific models, which are used to annotate end results. Such an approach addresses the need of semantically describing sensor data highlighted in [60].

One of the drawbacks of the RBS2P approach is that it does not consider any sophisticated security techniques to safeguard the system. However, the layered architecture might make it easier to deal with security threats. In a non-layered architecture, the programmer has to apply security measures to the entire system, which can be difficult. Assumingly in layered architectures, every security threat can be dealt with at the layer to which a particular threat applies (a bigger problem is broken down onto more manageable chunks). Due to the fact that sensor-based systems carry sensitive information, security metrics would need to be investigated further to make the system resilient to external threats. This is a scope for future work and it would need further investigation.

The work presented in this thesis lies somewhere in between the two subcategories of Chen and Khalil's [13] classification of sensor-based AR approaches: *wearable sensor-based* and *object-based*. Even though examples presented in this thesis did not include any camera sensors, the proposed approach does not exclude the use of visual sensors and remains open to any sources of information, including static data.

In order to assess effectiveness of SNM funded on rules and events and bounded together by the *layered event model*, two pilot studies were conducted. Active RFID RTLS technology was used in this case study research, as it provided an unobtrusive data collection environment.

Unlike other *wearable sensor-based* solutions [49] [38] [50] [8], these sensors were found acceptable from the subject point of view (small and unobtrusive). Measurement, validity and reliability research methodology was followed in order to assess the proposed system.

The first pilot study took place in the Regional Stroke Unit (RSU). The rehabilitation model was designed together with rehabilitation rules – both feeding the GEL layer of the RBS2P. Firstly, a reliability study was conducted, and the overall intraclass correlation coefficient (ICC) scores of 0.98 (wrist) and 0.90 (ankle) were achieved for the time taken by patients to walk between two locations. Moreover, the Pearson's Correlations Coefficients (PCC) were calculated for patient tags against staff tags, and for patient tags against the walking-aid tags – both were equal to 0.98, which demonstrated that the system was **capable of reliably detecting** the assistance of walking aids or healthcare professionals (HCP)s for the walking events. These results were considered completely acceptable by clinical specialists and were presented in [32].

Subsequently, two validity studies were conducted to judge whether the system actually measured what it was supposed to measure. For the purpose of the first stage of the validity study, a 31-step script for data collection was designed for patients to follow. The approach was to compare the system-recorded data against the manually-recorded data. First phase of validation was done by the means of the 10-meter walk test and its results were presented in the Welsh Stroke Conference [30]. In the phase two of the validity study, manual observations of patients were carried at the stroke unit. This data was then compared against system recorded data and lead to the conclusion that the system was comparable to the observational behaviour mapping techniques used in stroke units in the UK. Again, these results were published, this time at the UK Stroke Forum [31]. Moreover, the **Physiotherapy** and **Walking** rules were successfully validated against the manually recorded observations. This has led to the conclusion that the system is valid and capable of inferring these events.

Having satisfied the rehabilitation experts that the system was producing reliable and useful information, it has been permanently deployed at the stroke unit, running 24/7 for the course of 12 months. All rehabilitation equipment, patient beds and chairs, healthcare professionals and patients (who consented to take part in the study) were tagged (total of 172 tags); in order to provide longitudinal data (automatically recorded by the system) for our collaborators to analyse. The system was found capable of inferring multiple events for multiple patients and

hence overcomes limitations of work presented in [17] – where their solution was unable to recognise more than one activity at the time or the activities of more than one person.

The Home Monitoring (HM) study was commissioned based on the strength of the success of the RSU study. The home rehabilitation model was developed for this purpose and the same system (as in the stroke unit) was deployed at 11 patients' homes. This time, the system-recorded data (ankle and wrist) was compared against the step activity monitor (SAM), which reliably measures steps. High correlation between SAM and the system were observed. The applicability of the RBS2P to a problem of rehabilitation monitoring at home, has proven its adaptability to a different domain/set of requirements. However, due to privacy concerns not many locations in patients' homes were equipped with room locators. This has resulted in a reduced usability of the system and hence it would be beneficial to bring other sensors into the setup. Therefore, the Home Monitoring study also has potential to be continued. Due to new COTS sensors entering the market everyday and offering various sensing capabilities, it might be possible to incorporate other types of sensors into the setting. This would enable to acquire more information on patient's physical activity.

Presence of the *layered event model* and rules in the RBS2P carries further benefits in terms of validation and verification (V&V) of the system. The verification of the system is performed by cross-checking of rules, ontologies and event data. As a result, cases of redundancy (rules which never fire), deficiencies (data not consumed by any rules) and inconsistencies (events with no corresponding classes in the ontology, classes with no corresponding events and events for specialisation) can be discovered and highlighted to developers. This increases the maintainability (and transparency) of the system, especially in respect to new or changing user requirements. Sensor-based systems are fast evolving (either at the bottom – sensors, or at the top – requirements) and maintenance of system's integrity is highly important. There is other research work which looks at deep analysis of validation and testing of rule-based systems e.g. [36], however this is not in scope of this thesis. Exploring dynamic/in-detail analysis of the rule-base could a fruitful area for future work.

The validation of the system was carried out through the collaboration with our domain experts. Prototyped user interfaces were used in discussions in order to present the classes of event captured in the problem-specific (rehabilitation) ontology. Moreover, rehabilitation events that

the system was capable of producing were clearly highlighted. Such high level of system's transparency allowed rehabilitation experts to understand and validate the system. Hence, the presented approach to validation easies the cooperation between IT specialists and domain experts. All the presented V&V techniques help to fulfil the main purpose of SNM i.e. to support development, maintenance, deployment and execution of sensing based applications [58]. To the best of our knowledge there are no sensor-based systems, which offer such effective V&V techniques. The next step would be to implement the proposed GUI sketches and to further explore their usability.

7.2 Future Work

Apart from the future work areas already highlighted in this chapter, there is room for even more work to be carried out. The following are the areas which can be expanded further:

Low-level SNM wrappers: At the time of writing this thesis the only implemented wrappers were for Global Sensor Networks SNM. Implementing wrappers to interact with other low-level SNM, such as ITA Sensor Fabric or SWE:SensorML compatibles, would be highly beneficial.

Conflict resolution and uncertainty: These areas were beyond the scope of this research. RTL data was tested to be consistent and never reported a tag in two different places simultaneously. The strength of this type of technology is accurate localisation of objects and location information were one of the key conditions in rehabilitation rules. However, it is anticipated that conflict resolution can be achieved in the rule-based approach to activity monitoring. Events for some entity, before publishing, could be checked for temporal overlap with other events for the same entity. If conflicting cases occurred resolution strategies based on rules priorities or other qualities could be implemented. Techniques for handling uncertainty also require further exploration. Handling of uncertainty could perhaps be implemented by extending the *Event model* with a property to store the confidence values. In the process of events aggregation, these confidence scores could then be used to calculate a score for the infered event. Confidence scores could

potentially be used in the process of conflict resolution – events of lower confidence would be disregarded.

Backward chaining: The implemented system uses a forward chaining reasoning and hence it considers every single combination of premises of a rule. As a result, facts which might never be used get inferred anyway. Goal-driven backward chaining approach could be more efficient, as it only applies rules which are necessary to achieve the goal.

Cloud storage: Instead of using local storage, pushing data to the cloud would be a good alternative. In this way physical memory limitations of the machine hosting the platform could be overcome and the data could be safely stored on the Web. This solution would also tackle the problem of hardware failures, however it would require a constant Internet connection what in some environments is not permitted (hospital) or achievable (rural areas with no Internet connectivity).

Front-end rule editor: A front-end rule editor would enable domain experts to formulate rules without system administrator's assistance. This editor, apart from providing an environment for specifying rules, could also provide users with feedback based on meta-analysis of rules and good practice recommendations for writing rules. Such front-end interface would also enable users to publish their rules to the running system and enable them to review existing rules.

Real-time processing: Due to the nature of the two pilot studies and the inability to connect the system to the Internet, it was only field-tested in an 'offline' mode. Further experiments are needed to explore its functioning in an 'online' mode. Sensor systems are most useful when they provide real-time information. However, in the hospital or home setting this was not required.

Compliance with existing models: Integration with the Semantic Sensor Network (SSN) ontology would be beneficial, as it becomes a standard for sensor data semantics. For the idea of Semantic Sensor Web to realise, new technologies should link to existing standard to enable greater consumability of sensor-based data.

Trying the approach outside AM domain: To find the limitations of the proposed approach, it would be beneficial to test it in domains other than activity monitoring. Obviously the

Event model would need extending, as it is has been designed to carry temporal and spatial information important in the context of AM. This model however, is not capable of capturing all relevant information in other domains e.g. a thermometer reading would need a value and the indication of a metric system used.

7.3 Research Questions Answered

1. **How effective is a rule-based approach to SNM in the context of activity monitoring:**

   a) How well are the ten identified challenges for SNM met by this approach?

   The rule-based approach to SNM in the context of activity monitoring fully satisfies six out of ten identified challenges for SNM: abstraction support, data fusion, resource constraints, application knowledge, programming paradigm, and adaptability. Another three challenges: scalability, dynamic topology and QoS support; are also met by the proposed approach, however these also need to be handled by a low-level SNM plugged into the proposed platform. Finally, the security challenge was outside of the scope of this thesis, however some recommendation have been made as to the possible security strategies.

   b) What are the advantages & limitations of the proposed solution?

   The proposed solution apart from satisfying most of the ten identified challenges for SNM carries further benefits in terms of bridging COTS RTLS' with applications – eases development, maintenance, deployment, and execution of sensing-based application (definition of SNM). The rule-based approach not only helps to capture domain specific knowledge in software (in the form of rules) but also provides the means for explicitly representing data aggregation logic. Rules also provide means for reusing already computed/inferred events by other rules, which saves unnecessary computations and computer's primary and secondary memory. Rules assist in the process of events chaining, which transforms low-level sources of information into higher level conclusions. On the other side this approach has some limitations. One limitation is the expressiveness of rule-based systems.

Moreover, the example of the 'sandwich rule' presented in this thesis shows that complex rules have to be translated into simpler rules in order to get satisfactory performance. This is down to the working of the *Rete* algorithm, which performance is dependant on the number of rules, facts and patterns in each rule. Conflict resolution and handling uncertainty were beyond the scope of this thesis, however studies such as [20] shows that rule-based approach can tackle these.

2. **How does a semantic event model aid the process of sensor-data fusion/information inference?**

   The semantic *Event model* aids the process of sensor-data fusion by providing a uniform model which abstracts away sensors' heterogeneity. Uniform structure of sensor-inferred events is easier to process/fuse with other events. Moreover, the *Event model* separates hardware from domain concepts, which makes this solution open (at the bottom) to new sensors. Only the lowest-level model needs extending to accommodate for new sensors. Adapting to a new domain also becomes easier as no changes need to be made at the lower-level but only requires a formulation of domain's knowledge (openness at the top). Finally, the proposed *Event model* supports explanation of reasoning via its *evidencedBy* property, which adds credibility to the produced high-level events.

3. **How adaptable is a rule-based SNM platform in terms of changing user requirements, and bringing new sensors into the system?**

   Separation of layers in the proposed architecture brings many benefits. Firstly, low coupling of the system makes it open at both ends. Incorporating new sensors into the platform, apart from extending the existing models, requires changes to be made only at the platforms lowest layer by providing a wrapper for the sensor. Changing user requirements, on the other hand, only requires modifications to be made at the top GEL layer. This layer needs to be provided with experts knowledge formulated in the form of rules in order to meet new objectives. Throughout the two pilot studies presented in this thesis, requirements have been implemented incrementally and often rules had to be modified, re-written or added into the system. However, new rules very rarely had any impact on the existing rules, unless this was intentional. Therefore, its evolution, even with constantly changing requirements, was fairly smooth. Overall, this rule-based

SNM platform is highly adaptable to changing requirements/sensors.

4. **Can rule-based systems effectively help with validation and verification of a sensor network middleware?**

   Yes. As presented in Chapter 6 the platform funded on a rule-based system and a layered semantic event model helps greatly to verify and validate the system. The clear structure of the layered event model increases the transparency of the system and allows to judge whether its requirements are met – validate the system. Domain experts can contribute in the validation process as the layered event model is simple and easy to understand. Moreover, the presence of a rule-based system, event data and ontologies allows for system's verification. By cross checking these three components, one can discover inconsistency, deficiency and redundancy cases

Appendix A

Data Collection

A.1 CERISE Scoresheet

CERISE Tool

Patient code number: | **Date and Day of the week:**

	Activity	Location	Interaction
7.00			
7.10			
7.20			
7.30			
7.40			
7.50			
8.00			
8.10			
8.20			
8.30			
8.40			
8.50			
9.00			
9.10			
9.20			
9.30			
9.40			
9.50			
10.00			
10.10			
10.20			
10.30			
10.40			
10.50			
11.00			
11.10			
11.20			
11.30			
11.40			
11.50			

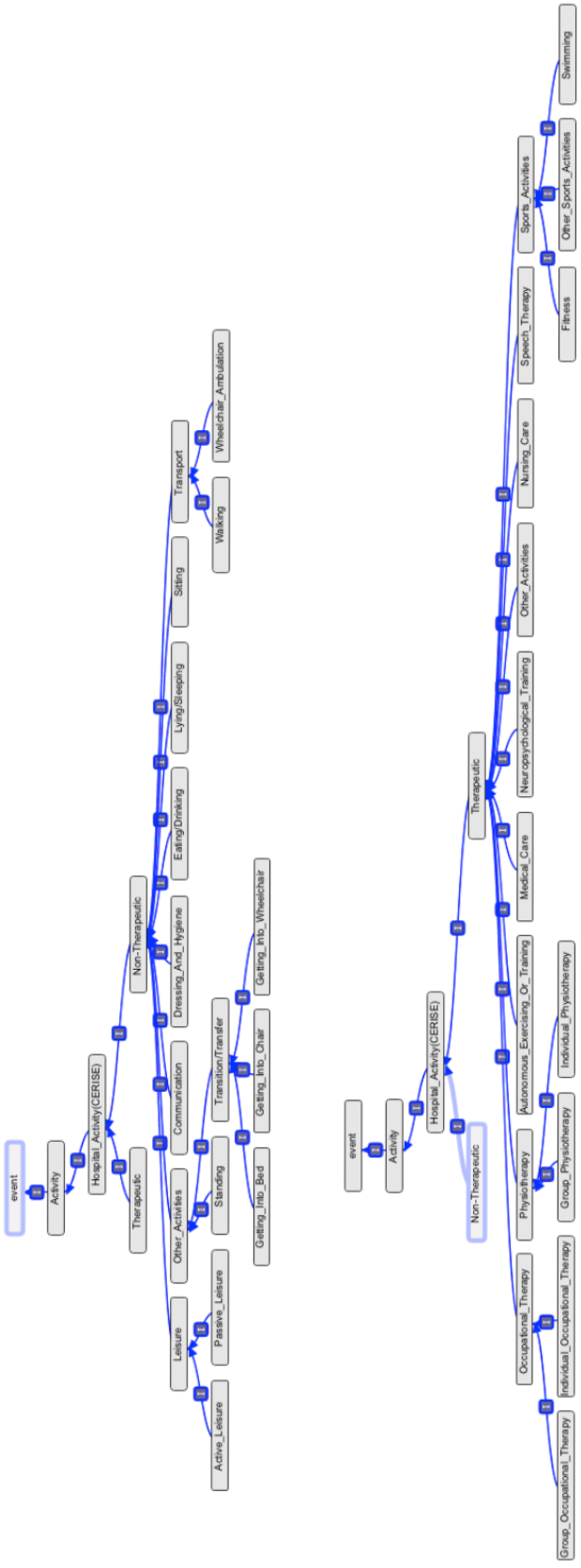
Code	**ACTIVITY** **Therapeutic activities**
	physiotherapy
111	*individual with physiotherapist*
112	*group therapy*
	occupational therapy
121	*individual with occupational therapist*
122	*group therapy*
13	speech therapy
14	neuropsychological training
15	nursing care
16	medical care
	sports activities
171	*swimming*
172	*fitness*
173	*other (e.g. horse riding,etc.)*
18	autonomous exercising and/or training
19	other activities
	Non- therapeutic activities
21	sitting
22	eating
23	transport (covering distances)
24	lying or sleeping
25	communication
26	dressing and hygiene
27	active leisure (reading a book, walking in garden etc.)
28	passive leisure (watching TV, listening to music, etc.)
29	other activities
	LOCATION
31	patient's room
32	therapy room
33	corridor
34	dining room
35	day room
36	toilet or bathroom
37	cafeteria
38	outside
39	any other location
	SOCIAL INTERACTION
41	doctors and/or specialists
42	therapists
43	nurses
44	other patients
45	visitors
46	other persons (e.g. cleaning personnel, etc.)
47	no-one
999	**not detectable**

A.2 Data Collection Scenario

1. Pt supine lying
2. HCS enters patient's room (Trigger event)
3. HCS positions chair
4. Pt starts to get up (Lying to sitting initiated)
5. Pt sits at the edge of the bed (Sitting completed)
6. Pt starts to get off the bed (Sit to stand initiated)
7. Pt sits in chair (Stand to sit completed)
8. HCS leaves patient's room
9. Pt in chair-inactive
10. Wheelchair brought into patient's room (Trigger event)
11. Pt starts to get up from chair (Sit to stand initiated)
12. Pt stands (Sit to stand completed)
13. Pt sits in wheelchair (Stand to sit completed)
14. Pt leaves own room
15. Pt enters Day room
16. Pt starts to get up from wheelchair (Sit to stand initiated)
17. Pt stands (Sit to stand completed)
18. Pt sits in chair (Stand to sit completed)
19. Wheelchair moved out of Day room
20. Pt in chair-inactive
21. HCS enters the Day room with walking aid (Trigger event)
22. HCS positions walking aid
23. Pt starts to get up from chair (Sit to stand initiated)
24. Pt stands (Sit to stand completed)
25. Walking with aid initiated (Toe off)
26. Pt leaves Day room
27. 10MWT starts
28. 10MWT ends
29. Pt enters Physiotherapy room
30. Pt sits in chair/bed (Stand to sit completed)
31. Walking aid moved to the side

Appendix B

Activity Classes



Event, RTL and Rehabilitation ontologies available at http://users.cs.cf.ac.uk/P.R.Woznowski/ont/rehab/

Appendix C

CLIPS Rules

C.1 Data Layer

```
(defrule Rule_1a
    ?trace <- (DeviceInMotionEvent (atPlace ?location)(involved ?address)(evidencedBy ?eb)(atTime
    ?time)(id ?id))
    (Person (name ?name)(defaultLocation ?defLoc)(deviceAddress ?address))
    =>
    (bind ?id2 (gensym*))
    (assert (MovingObject (atPlace ?location)(involvedAgent ?name)(involved nil)(evidencedBy ?id)
    (atTime ?time)(id ?id2)))
    (retract ?trace)
)

(defrule Rule_1b
    ?trace <- (DeviceInMotionEvent (atPlace ?location)(involved ?address)(evidencedBy ?eb)(atTime
    ?time)(id ?id))
    (Equipment (name ?name)(defaultLocation ?defLoc)(deviceAddress ?address))
    =>
    (bind ?id2 (gensym*))
    (assert (MovingObject (atPlace ?location)(involved ?name)(involvedAgent nil)(evidencedBy ?id)
    (atTime ?time)(id ?id2)))
    (retract ?trace)
)

(defrule Rule_1c
    ?trace <- (DeviceNotInMotionEvent (atPlace ?location)(involved ?address)(evidencedBy ?eb)
    (atTime ?time)(id ?id))
    (Person (name ?name)(defaultLocation ?defLoc)(deviceAddress ?address))
    =>
    (bind ?id2 (gensym*))
    (assert (StaticObject (atPlace ?location)(involvedAgent ?name)(involved nil)(evidencedBy ?id)
    (atTime ?time)(id ?id2)))
    (retract ?trace)
)

(defrule Rule_1d
    ?trace <- (DeviceNotInMotionEvent (atPlace ?location)(involved ?address)(evidencedBy ?eb)
    (atTime ?time)(id ?id))
    (Equipment (name ?name)(defaultLocation ?defLoc)(deviceAddress ?address))
    =>
    (bind ?id2 (gensym*))
    (assert (StaticObject (atPlace ?location)(involved ?name)(involvedAgent nil)(evidencedBy ?id)
    (atTime ?time)(id ?id2)))
    (retract ?trace)
)
```

```
(defrule Rule_2a
    "Aggregates time in events -> forms new events with two elements in the atTime slot which
    means first el is the 'from' and latter is the 'to'"
    ?trace1 <- (MovingObject (atPlace ?location)(involved ?name)(involvedAgent ?agent)(atTime
    ?time1)(id ?id1)(evidencedBy $?ev1))
    ?trace2 <- (MovingObject (atPlace ?location)(involved ?name)(involvedAgent ?agent)(atTime
    $?time2)(id ?id2)(evidencedBy $?ev2))
    ;?time1 has to be greater than ?time2
    (test (> ?time1 (nth$ (length$ $?time2) $?time2)))
    (test (< (- ?time1 (nth$ (length$ $?time2) $?time2)) 3000))
    =>
    (retract ?trace1)
    (if (listp $?time2) then
        (modify ?trace2 (atTime (replace$ $?time2 2 2 ?time1)) (evidencedBy $?ev2))
     else
        (retract ?trace2)
        (bind ?id3 (gensym*))
        (assert (MovingObject (atPlace ?location)(involved ?name)(involvedAgent ?agent)
        (evidencedBy ?id2 ?id1)(atTime (replace$ $?time2 2 2 ?time1))(id ?id3)))
    )
)

(defrule Rule_2b
    "Aggregates time in events -> forms new events with two elements in the atTime slot which
    means first el is the 'from' and latter is the 'to'"
    ?trace1 <- (StaticObject (atPlace ?location)(involved ?name)(involvedAgent ?agent)(atTime
    ?time1)(id ?id1)(evidencedBy $?ev1))
    ?trace2 <- (StaticObject (atPlace ?location)(involved ?name)(involvedAgent ?agent)(atTime
    $?time2)(id ?id2)(evidencedBy $?ev2))
    ; ?time1 has to be greater than ?time2
    (test (> ?time1 (nth$ (length$ $?time2) $?time2)))
    (test (< (- ?time1 (nth$ (length$ $?time2) $?time2)) 3000))
    =>
    (retract ?trace1)
    (if (listp $?time2) then
        (modify ?trace2 (atTime (replace$ $?time2 2 2 ?time1)) (evidencedBy $?ev2))
     else
        (retract ?trace2)
        (bind ?id3 (gensym*))
        (assert (StaticObject (atPlace ?location)(involved ?name)(involvedAgent ?agent)
        (evidencedBy ?id2 ?id1)(atTime (replace$ $?time2 2 2 ?time1))(id ?id3)))
    )
)
```

C.2 Sensor Event Layer

```
(defrule Rule_3a
    ;discovers the initial pair of loc + "000" and asserts LocCorrData fact
    (declare (auto-focus TRUE)(salience 100))
    ?fact2 <- (MovingObject (atPlace "000")(involved "nil")(involvedAgent ?agent)(atTime $?time2)
    (id ?id2)(evidencedBy $?ev2))
    (MovingObject (atPlace ?location1&~"000")(involved "nil")(involvedAgent ?agent)(atTime
    $?time1)(id ?id1&~?id2)(evidencedBy $?ev1))
    ;tests if f2 was after f1 and within 3sec
    (test (>= (nth$ 1 $?time2) (nth$ (length$ $?time1) $?time1)))
    (test (< (- (nth$ 1 $?time2) (nth$ (length$ $?time1) $?time1)) 3000))
    =>
    (assert (LocationCorrectionData (name ?agent)(location ?location1)(listOfZeroLoc ?fact2)
    (initFactId ?id1)(finishTime (nth$ (length$ $?time2) $?time2)) ))
    (store Rule_3a "Rule_3a")
)
(defrule Rule_3b
    ;discovers the initial pair of loc + "000" and asserts LocCorrData fact
    (declare (auto-focus TRUE)(salience 100))
    ?fact2 <- (StaticObject (atPlace "000")(involved "nil")(involvedAgent ?agent)(atTime $?time2)
    (id ?id2)(evidencedBy $?ev2))
    (StaticObject (atPlace ?location1&~"000"|"999")(involved "nil")(involvedAgent ?agent)(atTime
    $?time1)(id ?id1&~?id2)(evidencedBy $?ev1))
    ;tests if f2 was after f1 and within 3sec
    (test (>= (nth$ 1 $?time2) (nth$ (length$ $?time1) $?time1)))
    (test (< (- (nth$ 1 $?time2) (nth$ (length$ $?time1) $?time1)) 3000))
    =>
    (assert (LocationCorrectionData (name ?agent)(location ?location1)(listOfZeroLoc ?fact2)
    (initFactId ?id1)(finishTime (nth$ (length$ $?time2) $?time2)) ))
    (store Rule_3b "Rule_3b")
)
(defrule Rule_3c
    ;discovers the initial pair of loc + "000" and asserts LocCorrData fact
    (declare (auto-focus TRUE)(salience 100))
    ?fact2 <- (StaticObject (atPlace "000")(involved "nil")(involvedAgent ?agent)(atTime $?time2)
    (id ?id2)(evidencedBy $?ev2))
    (MovingObject (atPlace ?location1&~"000")(involved "nil")(involvedAgent ?agent)(atTime
    $?time1)(id ?id1&~?id2)(evidencedBy $?ev1))
    ;tests if f2 was after f1 and within 3sec
    (test (>= (nth$ 1 $?time2) (nth$ (length$ $?time1) $?time1)))
    (test (< (- (nth$ 1 $?time2) (nth$ (length$ $?time1) $?time1)) 3000))
    =>
    (assert (LocationCorrectionData (name ?agent)(location ?location1)(listOfZeroLoc ?fact2)
    (initFactId ?id1)(finishTime (nth$ (length$ $?time2) $?time2)) ))
    (store Rule_3c "Rule_3c")
)
(defrule Rule_3d
    ;discovers the initial pair of loc + "000" and asserts LocCorrData fact
    (declare (auto-focus TRUE)(salience 100))
    ?fact2 <- (MovingObject (atPlace "000")(involved "nil")(involvedAgent ?agent)(atTime $?time2)
    (id ?id2)(evidencedBy $?ev2))
    (StaticObject (atPlace ?location1&~"000"|"999")(involved "nil")(involvedAgent ?agent)(atTime
    $?time1)(id ?id1&~?id2)(evidencedBy $?ev1))
    ;tests if f2 was after f1 and within 3sec
    (test (>= (nth$ 1 $?time2) (nth$ (length$ $?time1) $?time1)))
    (test (< (- (nth$ 1 $?time2) (nth$ (length$ $?time1) $?time1)) 3000))
    =>
    (assert (LocationCorrectionData (name ?agent)(location ?location1)(listOfZeroLoc ?fact2)
    (initFactId ?id1)(finishTime (nth$ (length$ $?time2) $?time2)) ))
    (store Rule_3d "Rule_3d")
)
```

```
(defrule Rule_3e
    ;updates LocCorrectionData if more 000 follows
    (declare (auto-focus TRUE)(salience 10))
    ?fact1 <- (LocationCorrectionData (name ?agent)(location ?location1)(listOfZeroLoc
    $?zeroLocs)(initFactId ?id1)(finishTime ?fTime))
    ?fact2 <- (MovingObject (atPlace "000")(involved "nil")(involvedAgent ?agent)(atTime $?time2)
    (id ?id2)(evidencedBy $?ev2))
    ;tests if f2 was after f1 and within 3sec
    (test (>= (nth$ 1 $?time2) ?fTime))
    (test (< (- (nth$ 1 $?time2) ?fTime) 3000))
    =>
    (modify ?fact1 (listOfZeroLoc ?zeroLocs ?fact2) (finishTime (nth$ (length$ $?time2) $?time2)))
    (store Rule_3e "Rule_3e")
    )

(defrule Rule_3f
    ;updates LocCorrectionData if more 000 follows
    (declare (auto-focus TRUE)(salience 10))
    ?fact1 <- (LocationCorrectionData (name ?agent)(location ?location1)(listOfZeroLoc $?zeroLocs)
    (initFactId ?id1)(finishTime ?fTime))
    ?fact2 <- (StaticObject (atPlace "000")(involved "nil")(involvedAgent ?agent)(atTime $?time2)
    (id ?id2)(evidencedBy $?ev2))
    ;tests if f2 was after f1 and within 3sec
    (test (>= (nth$ 1 $?time2) ?fTime))
    (test (< (- (nth$ 1 $?time2) ?fTime) 3000))
    =>
    (modify ?fact1 (listOfZeroLoc ?zeroLocs ?fact2) (finishTime (nth$ (length$ $?time2) $?time2)))
    (store Rule_3f "Rule_3f")
)

(defrule Rule_3g
    (declare (auto-focus TRUE)(salience 1))
    ?fact1 <- (LocationCorrectionData (name ?agent)(location ?location1)(listOfZeroLoc $?zeroLocs)
    (initFactId ?id1)(finishTime ?fTime))
    (MovingObject (atPlace ?location1)(involved "nil")(involvedAgent ?agent)(atTime $?time2)(id ?id2)
    (evidencedBy $?ev2))
    ;tests if f2 was after f1 and within 3sec
    (test (>= (nth$ 1 $?time2) ?fTime))
    (test (< (- (nth$ 1 $?time2) ?fTime) 3000))
    =>
    (foreach ?fID ?zeroLocs (modify ?fID (atPlace ?location1)) )
    (retract ?fact1)
    (store Rule_3g "Rule_3g")
)
```

C.3 Goal Event Layer

```
(defrule Rule_4a
    (declare (auto-focus TRUE)(salience 100))
    ?fact <- (StaticObject (atPlace ?location)(involved ?name)(involvedAgent ?agent)(atTime $?time)
    (id ?id1)(evidencedBy $?ev))
    =>
    (bind ?id2 (str-cat gcr (gensym*)))
    (assert (Event (atPlace ?location)(involved ?name)(involvedAgent ?agent)(atTime ?time)(id ?id2)
    (evidencedBy ?id1)))
    (assert (EvidenceFor (rule "Rule_4a")(factID ?id2)))
    (store Rule_4a "Rule_4a")
)

(defrule Rule_4b
    (declare (auto-focus TRUE)(salience 100))
    ?fact <- (MovingObject (atPlace ?location)(involved ?name)(involvedAgent ?agent)(atTime $?time)
    (id ?id1)(evidencedBy $?ev))
    =>
    (bind ?id2 (str-cat gcr (gensym*)))
    (assert (Event (atPlace ?location)(involved ?name)(involvedAgent ?agent)(atTime ?time)(id ?id2)
    (evidencedBy ?id1)))
    (assert (EvidenceFor (rule "Rule_4b")(factID ?id2)))
    (store Rule_4b "Rule_4b")
)

(defrule Rule_4c
    (declare (auto-focus TRUE)(salience 10))
    ?fact2 <- (Event (atPlace ?location)(involved ?name)(involvedAgent ?agent)(atTime $?time2)
    (id ?id2)(evidencedBy $?ev2))
    ?fact1 <- (Event (atPlace ?location)(involved ?name)(involvedAgent ?agent)(atTime $?time1)
    (id ?id1&~?id2)(evidencedBy $?ev1))
    (test (>= (nth$ 1 $?time2) (nth$ (length$ $?time1) $?time1)))
    (test (< (- (nth$ 1 $?time2) (nth$ (length$ $?time1) $?time1)) 3000))
    =>
    (bind ?id (str-cat mixE (gensym*)))
    (assert (Event (atPlace ?location)(involved ?name)(involvedAgent ?agent)(atTime (nth$ 1 ?time1)
    (nth$ (length$ ?time2) ?time2))(id ?id)(evidencedBy $?ev2 $?ev1)))
    (assert (EvidenceFor (rule "Rule_4c")(factID ?id)))
    (retract ?fact1)
    (retract ?fact2)
    (store Rule_4c "Rule_4c")
)

(defrule Colocated_Rule
    (declare (auto-focus TRUE))
    (Patients::Patient (name ?agent1)(defaultLocation ?ptDLoc)(deviceAddress ?ptRFAddress))
    (Staff::Staff (name ?agent2)(role "physio")(defaultLocation ?staffLoc)
    (deviceAddress ?staffDevice))
    (Patients::Event (atPlace ?location)(atTime $?time1)(involved "nil")(involvedAgent ?agent1)
    (id ?id1)(evidencedBy $?ev1))
    (Staff::Event (atPlace ?location&)(atTime $?time2)(involved "nil")(involvedAgent ?agent2)
    (id ?id2&~?id1)(evidencedBy $?ev2))
    =>
    (bind ?time (overlaps ?time1 ?time2))
    (if (neq 0 ?time) then
        (bind ?id (gensym*))
        (assert (Colocated (atPlace ?location)(involved "nil")(involvedAgent (create$ ?agent1
        ?agent2))(atTime ?time)(id ?id)(evidencedBy (create$ ?id1 ?id2))))
        (assert (EvidenceFor (rule "Colocated_Rule")(factID ?id)))
        (store Colocated_Rule "Colocated_Rule")
    )
)
```

```
(deffunction overlapping-time (?a $?b)
    "Function to test if ?a is between elements of ?b"
    (if (and (>= ?a (nth$ 1 $?b))(<= ?a (nth$ 2 $?b))) then
         (return ?a)
        else
         (return 0)
    )
)

(deffunction overlapping-period (?a ?b)
    ;if there are no overlaps, return 0
    (if (or (< (nth$ 2 ?a) (nth$ 1 ?b) ) (< (nth$ 2 ?b) (nth$ 1 ?a))) then
         (return 0)
    )
    ;if starting points are equal
    (if (= (nth$ 1 ?a) (nth$ 1 ?b)) then
        ;if ?a finishing point is later than ?b finishing point
        (if (> (nth$ 2 ?a) (nth$ 2 ?b) ) then
             (return ?b)
            else
             (return ?a)
        )
)
    ;if ?a starting point is earlier than ?b starting point
    (if (< (nth$ 1 ?a) (nth$ 1 ?b) ) then
        ;if ?a finishing point overlaps with ?b starting point
        (if (= (nth$ 2 ?a)(nth$ 1 ?b) ) then
            (return (nth$ 1 ?b) )
        )
        ;if ?a finishing point is earlier or equal to ?b finishing point
        (if (<= (nth$ 2 ?a)(nth$ 2 ?b) ) then
             (return (create$ (nth$ 1 ?b)(nth$ 2 ?a)) )
            else
             (return ?b )
        )
    ;if ?a starting point is later than ?b starting point
    else
        ;if ?b finishing point overlaps with ?a starting point
        (if (= (nth$ 2 ?b)(nth$ 1 ?a) ) then
            (return (nth$ 1 ?a) )
        )
        ;if ?a finishing point is later or equal to ?b finishing point
        (if (>= (nth$ 2 ?a)(nth$ 2 ?b) ) then
             (return (create$ (nth$ 1 ?a)(nth$ 2 ?b)) )
            else
             (return ?a )
        )
    )
)
```

```
(deffunction overlaps (?a ?b)
    (if (> (length$ ?a) 1) then
        (if (> (length$ ?b) 1) then
             ;compute overlap
             (return (overlapping-period ?a ?b))
            else
             ;single point overlap if ?b is between elements of ?a
             (return (overlapping-time ?b ?a))
        )
     else
         (if (> (length$ ?b) 1) then
             ;single point overlap if ?a is between elements of ?b
             (return (overlapping-time ?a ?b))
            else
             ;single point overlap if ?a==?b
             (if (= ?a ?b) then
                  (return ?a)
                else
                 (return 0)
             )
         )
     )
)
```

Appendix D

Verification Report

All verification techniques mentioned in Chapter 6 are used to contribute to the developer's report, which provides a great degree of maintainability to the system, as well as helps in the development process. The UI design of the developer interface is shown in Fig.D.1. This interface has not been developed at the time of writing this thesis, however the code for generating data for each of the 5 panels has been implemented and tested to work in accordance to the specification. Currently all this data is printed to the console as raw text and needs to be presented in a more readable form.

Fig.D.1 contains 5 panels, each of which reports on a different aspect of the system. Panels number 1 and 2 report on system's coherence. They reports on rules which never fire and un-used inputs (data not consumed by any rules) respectively. Each of these panels points developer to the component (fact, rule, ontology) which causes the lack of consistency, by providing detailed description of faults, and resolution strategies. The causes for each of the 5 cases were discussed in Chapter 6. Panels 3 and 4 report on: events in the WM with no corresponding classes in the domain ontology, and on classes within the ontology for which there are no instances in the system WM, respectively. Again, the events or classes of interest are listed together with recommended actions. Finally, panel 5 lists all classes which can be further specialised into their sub-classes. The class descendants are listed together with the name of the rule which produces the superclass. Therefore, the developer is kept up-to-date with the current state of the system and has all diagnostic information needed to locate the faults quickly and to resolve them.

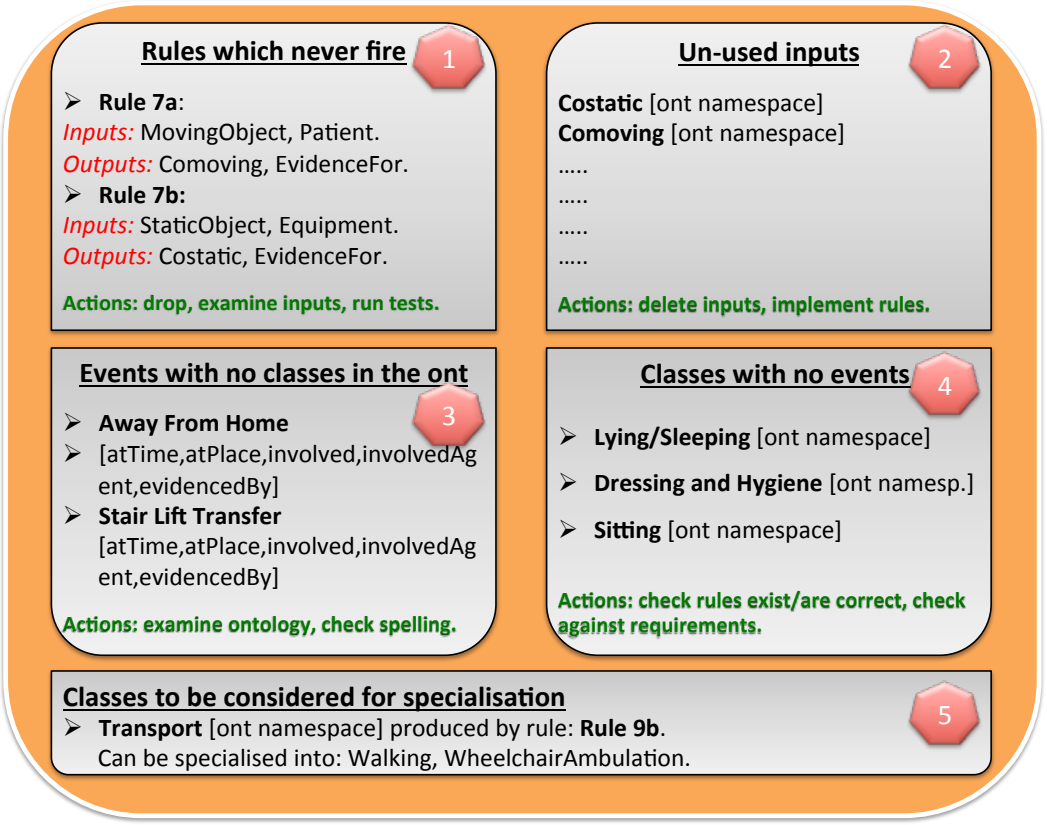


Figure D.1: UI Design: Developer's Report

Bibliography

[1] Karl Aberer, Manfred Hauswirth, and Ali Salehi. A middleware for fast and flexible sensor network deployment. In *Proceedings of the 32nd international conference on Very large data bases*, pages 1199–1202. VLDB Endowment, 2006.

[2] Jake K. Aggarwal and Michael S Ryoo. Human activity analysis: A review. *ACM Computing Surveys (CSUR)*, 43(3):16, 2011.

[3] James F Allen and George Ferguson. Actions and events in interval temporal logic. *Journal of Logic and Computation*, 4(5):531–579, 1994.

[4] Vincenzo Ambriola and Genoveffa Tortora. *Advances in Software Engineering and Knowledge Engineering*, volume 2 of *Series on software engineering and knowledge engineering*. World Scientific, 1993.

[5] Anastasia Analyti, Grigoris Antoniou, Carlos Viegas Damásio, and Gerd Wagner. Stable model theory for extended RDF ontologies. In *The Semantic Web–ISWC 2005*, pages 21–36. Springer, 2005.

[6] Alexander Artikis, Anastasios Skarlatidis, Francois Portet, and Georgios Paliouras. Logic-based event recognition. *The Knowledge Engineering Review*, 27:469–506, 12 2012.

[7] Majid Bahrepour, Nirvana Meratnia, and Paul JM Havinga. Sensor fusion-based event detection in wireless sensor networks. In *Mobile and Ubiquitous Systems: Networking & Services, MobiQuitous, 2009. MobiQuitous' 09. 6th Annual International*, pages 1–8. IEEE, 2009.

[8] Ling Bao and StephenS. Intille. Activity Recognition from User-Annotated Acceleration Data. In Alois Ferscha and Friedemann Mattern, editors, *Pervasive Computing*, volume 3001 of *Lecture Notes in Computer Science*, pages 1–17. Springer Berlin Heidelberg, 2004.

[9] Len Bass, Paul Clements, and Rick Kazman. *Software architecture in practice*. Addison-Wesley Professional, 2003.

[10] Konrad Borowiecki and Alun Preece. An ontological approach to integrating task representations in sensor networks. In *Proc 17th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2010). Springer*, 2010.

[11] Mike Botts, George Percivall, Carl Reed, and John Davidson. OGC® sensor web enablement: Overview and high level architecture. In *GeoSensor networks*, pages 175–190. Springer, 2008.

[12] Loannis Chatzigiannakis, Georgios Mylonas, and Sotiris Nikoletseas. 50 ways to build your application: A survey of middleware and systems for Wireless Sensor Networks. *2007 IEEE Conference on Emerging Technologies & Factory Automation (EFTA 2007)*, pages 466–473, September 2007.

[13] Liming Chen and Ismail Khalil. Activity Recognition: Approaches, Practices and Trends. In Liming Chen, Chris D. Nugent, Jit Biswas, and Jesse Hoey, editors, *Activity Recognition in Pervasive Intelligent Environments*, volume 4 of *Atlantis Ambient and Pervasive Intelligence*, pages 1–31. Atlantis Press, 2011.

[14] Michael Compton, Payam Barnaghi, Luis Bermudez, Raul García-Castro, Oscar Corcho, Simon Cox, John Graybeal, Manfred Hauswirth, Cory Henson, Arthur Herzog, et al. The SSN ontology of the W3C semantic sensor network incubator group. *Web Semantics: Science, Services and Agents on the World Wide Web*, 17:25–32, 2012.

[15] Liesbet De Wit, Koen Putman, Birgit Schuback, Arnošt Komárek, Felix Angst, Ilse Baert, Peter Berman, Kris Bogaerts, Nadine Brinkmann, Louise Connell, et al. Motor and Functional Recovery After Stroke A Comparison of 4 European Rehabilitation Centers. *Stroke*, 38(7):2101–2107, 2007.

[16] Norman Dziengel, Georg Wittenburg, and Jochen Schiller. Towards distributed event detection in wireless sensor networks. In *Adjunct Proc. of 4th IEEE/ACM Intl. Conf. on Distributed Computing in Sensor Systems (DCOSSÓ8), Santorini Island, Greece*, 2008.

[17] Vasileios Efthymioue, Maria Koutraki, and Grigoris Antoniou. Real-Time Activity Recognition and Assistance in Smart Classrooms. *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal*, 1(1):9–22, 2013.

[18] Xiang Fei and Evan Magill. Rule execution and event distribution middleware for PROSEN-WSN. In *Sensor Technologies and Applications, 2008. SENSORCOMM'08. Second International Conference on*, pages 580–585. IEEE, Aug 2008.

[19] Xiang Fei and Zixi Yu. Development of a Rule-Based Wireless Sensor Network Middleware. In *Proceedings of the 16th International Conference on Automation & Computing*, pages 13–18, 2010.

[20] Chrysi Filippaki, Grigoris Antoniou, and Ioannis Tsamardinos. Using constraint optimization for conflict resolution and detail control in activity recognition. In *Ambient Intelligence*, pages 51–60. Springer, 2011.

[21] Jill A Fisher and Torin Monahan. Evaluation of real-time location systems in their hospital contexts. *International journal of medical informatics*, 81(10):705–712, 2012.

[22] Ernest Friedman-Hill. *JESS in Action*. Manning Greenwich, CT, 2003.

[23] Qing Fu and Guenther Retscher. Active RFID Trilateration and Location Fingerprinting Based on RSSI for Pedestrian Navigation. *Journal of Navigation*, 62:323–340, 4 2009.

[24] Patrik Fuhrer and Dominique Guinard. Building a Smart Hospital using RFID Technologies. *ECEH*, 91:131–142, 2006.

[25] Wendi B Heinzelman, Amy L Murphy, Hervaldo S Carvalho, and Mark A Perillo. Middleware to support sensor network applications. *Network, IEEE*, 18(1):6–14, 2004.

[26] Karen Henricksen and Ricky Robinson. A survey of middleware for sensor networks. *Proceedings of the International Workshop on Middleware for Sensor Networks - MidSens 06*, pages 60–65, 2006.

[27] Ian Horrocks, Peter F Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, Mike Dean, et al. SWRL: A semantic web rule language combining OWL and RuleML. *W3C Member submission*, 21:79, 2004.

[28] Duy Tâm Gilles Huynh. *Human activity recognition with wearable sensors*. PhD thesis, TU Darmstadt, 2008.

[29] Tomohito Inomata, Futoshi Naya, Noriaki Kuwahara, Fumio Hattori, and Kiyoshi Kogure. Activity recognition from interactions with objects using dynamic Bayesian network. In *Proceedings of the 3rd ACM International Workshop on Context-Awareness for Self-Managing Systems*, pages 39–42. ACM, 2009.

[30] Arshi S Iqbal, Przemyslaw R Woznowski, Allison Cooper, Alun Preece, and Robert van Deursen. Comparing a sensor based Real Time Location System with an Observation Based System and the 10-MWT to determine walking speed in the clinical setting. In *Welsh Stroke Conference*, Newport, Wales, June 2012.

[31] Arshi S Iqbal, Przemyslaw R Woznowski, Allison Cooper, Alun Preece, and Robert van Deursen. Measuring functional activities of patients in a stroke unit: Comparison of a sensor based Real Time Location System with the Observational Behaviour Mapping Technique. In *UK Stroke Forum*, Harrogate, December 2012.

[32] Arshi S Iqbal, Przemyslaw R Woznowski, Allison Cooper, Alun Preece, and Robert van Deursen. Reliability of a new computerised system to detect walking ability in the early stages of recovery post stroke. In *Physiotherapy Research Society 30th Scientific Meeting*, Sheffield, May 2012.

[33] Arshi S Iqbal, Przemyslaw R Woznowski, Allison Cooper, Alun Preece, and Robert van Deursen. A new computerised system can continuously measure functional activities op patients in a stroke rehabilitation unit. In *European Stroke Conference*, London, May 2013.

[34] Adrianne Jeffries. A Sensor In Every Chicken: Cisco Bets on the Internet of Things, July 2010. `http://readwrite.com/2010/07/29/cisco_futurist_predicts_internet_ of_things_1000#co_awesm=~ojafefewJLzaKH`.

[35] R. Kazman, M. Klein, M. Barbacci, T. Longstaff, H. Lipson, and J. Carriere. The architecture tradeoff analysis method. In *Engineering of Complex Computer Systems, 1998. ICECCS '98. Proceedings. Fourth IEEE International Conference on*, pages 68–78, Aug 1998.

[36] James D. Kiper. Structural testing of rule-based expert systems. *ACM Trans. Softw. Eng. Methodol.*, 1(2):168–187, April 1992.

[37] Zakir Laliwala, Vikram Sorathia, and Sanjay Chaudhary. Semantic and rule based event-driven services-oriented agricultural recommendation system. In *Distributed Computing Systems Workshops, 2006. ICDCS Workshops 2006. 26th IEEE International Conference on*, pages 24–24. IEEE, 2006.

[38] Seon-Woo Lee and Kenji Mase. Activity and location recognition using wearable sensors. *Pervasive Computing, IEEE*, 1(3):24–32, 2002.

[39] Lucas Leidinger and Kerry Taylor. Demonstration: Defining and Detecting Complex Events in Sensor Networks. In *SSN*, pages 111–113, 2011.

[40] Jie Liu and Feng Zhao. Composing semantic services in open sensor-rich environments. *Network, IEEE*, 22(4):44–49, 2008.

[41] Takuya Maekawa, Yutaka Yanagisawa, Yasue Kishino, Katsuhiko Ishiguro, Koji Kamei, Yasushi Sakurai, and Takeshi Okadome. Object-based activity recognition with heterogeneous sensors on wrist. In *Pervasive Computing*, pages 246–264. Springer, 2010.

[42] Mihai Marin-Perianu and Paul Havinga. D-FLER–a distributed fuzzy logic engine for rule-based wireless sensor networks. In *Ubiquitous Computing Systems*, pages 86–101. Springer, 2007.

[43] Pedro Jose Marron. Middleware approaches for sensor networks. *University of Stuttgart, Summer School on WSNs and Smart Objects Schloss Dagstuhl*, 2005.

[44] Cecilia Mascolo, Stephen Hailes, Leonidas Lymberopoulos, Gian Pietro Picco, Paolo Costa, Gordon Blair, Paul Okanda, Thirunavukkarasu Sivaharan, Wolfgang Fritsche, Mayer Karl, et al. Survey of middleware for networked embedded systems. *Project Report: http://www. ist-runes. org/docs/deliverables/D5_01. pdf*, 2005.

[45] Mohammad M Molla and Sheikh Iqbal Ahamed. A survey of middleware for sensor network and challenges. In *Parallel Processing Workshops, 2006. ICPP 2006 Workshops. 2006 International Conference on*, pages 6 pp.–228. IEEE, 2006.

[46] Georgios Mylonas. A Survey of Middleware and Systems for Wireless Sensor Networks. In *AEOLUS Fall Workshop*, 2007.

[47] Martin Oćonnor, Holger Knublauch, Samson Tu, Benjamin Grosof, Mike Dean, William Grosso, and Mark Musen. Supporting rule system interoperability on the semantic web with SWRL. In *The Semantic Web–ISWC 2005*, pages 974–986. Springer, 2005.

[48] Michael J OĎonnell. Knowledge Engineering 2008: Advantages and Disadvantages of Rule-based Reasoning, March 2008. `http://arantxa.ii.uam.es/ modonnel/IC/03_RulesIII.pdf`.

[49] Juha Parkka, Miikka Ermes, Panu Korpipaa, Jani Mantyjarvi, Johannes Peltola, and Ilkka Korhonen. Activity classification using realistic data from wearable sensors. *Information Technology in Biomedicine, IEEE Transactions on*, 10(1):119–128, 2006.

[50] Donald J Patterson, Dieter Fox, Henry Kautz, and Matthai Philipose. Fine-grained activity recognition by aggregating abstract object usage. In *Wearable Computers, 2005. Proceedings. Ninth IEEE International Symposium on*, pages 44–51. IEEE, 2005.

[51] Alun Preece. Building the right system right. In *Proceedings of KAẂ98 Eleventh Workshop on Knowledge Acquisition, Modeling and Management*, 1998.

[52] Alun Preece, Diego Pizzocaro, Konrad Borowiecki, Geeth de Mel, M Gomez, W Vasconcelos, Amotz Bar-Noy, Matthew P Johnson, Thomas La Porta, Hosan Rowaihy, et al. Reasoning and resource allocation for sensor-mission assignment in a coalition context. *MILCOM 2008 - 2008 IEEE Military Communications Conference*, pages 1–7, November 2008.

[53] Yves Raimond and Samer Abdallah. The event ontology. Technical report, Technical report, 2007. http://motools. sourceforge. net/event, 2007.

[54] G. Retscher and Qing Fu. Continuous indoor navigation with RFID and INS. In *Position Location and Navigation Symposium (PLANS), 2010 IEEE/ION*, pages 102–112, May 2010.

[55] Daniele Riboni and Claudio Bettini. Context-aware activity recognition through a combination of ontological and statistical reasoning. In *Ubiquitous Intelligence and Computing*, pages 39–53. Springer, 2009.

[56] Daniele Riboni and Claudio Bettini. COSAR: hybrid reasoning for context-aware activity recognition. *Personal Ubiquitous Comput.*, 15(3):271–289, March 2011.

[57] Kay Römer, Christian Frank, Pedro José Marrón, and Christian Becker. Generic role assignment for wireless sensor networks. In *Proceedings of the 11th workshop on ACM SIGOPS European workshop*, page 2. ACM, 2004.

[58] Kay Römer, Oliver Kasten, and Friedemann Mattern. Middleware challenges for wireless sensor networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 6(4):59–61, October 2002.

[59] Ryan Shaw, Raphaël Troncy, and Lynda Hardman. Lode: Linking open descriptions of events. In *The Semantic Web*, pages 153–167. Springer, 2009.

[60] Amit Sheth, Cory Henson, and Satya S Sahoo. Semantic sensor web. *Internet Computing, IEEE*, 12(4):78–83, 2008.

[61] Amit Sheth and Matthew Perry. Traveling the semantic web through space, time, and theme. *Internet Computing, IEEE*, 12(2):81–86, 2008.

[62] E. Simon. *Distributed information systems: from client/server to distributed multimedia*. McGraw-Hill, 1996.

[63] Martin Strohbach, Hans-Werner Gellersen, Gerd Kortuem, and Christian Kray. Cooperative artefacts: Assessing real world situations with embedded technology. In *UbiComp 2004: Ubiquitous Computing*, pages 250–267. Springer, 2004.

[64] Kerry Taylor and Lucas Leidinger. Ontology-driven complex event processing in heterogeneous sensor networks. In *The Semanic Web: Research and Applications*, pages 285–299. Springer, 2011.

[65] Kerry Taylor and Patrick Penkala. Using explicit semantic representations for user programming of sensor devices. In *Proceedings of the Fifth Australasian Ontology Workshop-Volume 112*, pages 47–56. Australian Computer Society, Inc., 2009.

[66] Kirsten Terfloth, Georg Wittenburg, and Jochen Schiller. FACTS–a rule-based middleware architecture for wireless sensor networks. In *Communication System Software and Middleware, 2006. Comsware 2006. First International Conference on*, pages 1–8. IEEE, 2006.

[67] Weixin Wang, Jongwoo Sung, and Daeyoung Kim. Complex event processing in epc sensor network middleware for both rfid and wsn. In *Object Oriented Real-Time Distributed Computing (ISORC), 2008 11th IEEE International Symposium on*, pages 165–169. IEEE, 2008.

[68] Kamin Whitehouse, Feng Zhao, and Jie Liu. Semantic Streams: a framework for declarative queries and automatic data interpretation. *Microsoft Research*, 1, 2005.

[69] Kamin Whitehouse, Feng Zhao, and Jie Liu. Semantic streams: A framework for composable semantic interpretation of sensor data. In *Wireless Sensor Networks*, pages 5–20. Springer, 2006.

[70] Joel Wright, Christopher Gibson, Flavio Bergamaschi, Kelvin Marcus, Tien Pham, Ryan Pressley, and Gunjan Verma. ITA sensor fabric. In *SPIE Defense, Security, and Sensing*, pages 733319–733319. International Society for Optics and Photonics, 2009.

[71] Joel Wright, Christopher Gibson, Flavio Bergamaschi, Kelvin Marcus, Ryan Pressley, Gunjan Verma, and Gene Whipps. A dynamic infrastructure for interconnecting disparate isr/istar assets (the ita sensor fabric). In *Information Fusion, 2009. FUSION'09. 12th International Conference on*, pages 1393–1400. IEEE, 2009.

[72] Jiaxin Xie, Zixi Yu, Xiang Fei, and Partheepan Kandaswamy. Applications development on a rule-based WSN middleware. In *SENSORCOMM 2012, The Sixth International Conference on Sensor Technologies and Applications*, pages 106–111, 2012.