CONTEXT INFORMATION REFINEMENT FOR PERVASIVE MEDICAL SYSTEMS

BY

BIRHANU MEKURIA ESHETE

A THESIS SUBMITTED TO
THE SCHOOL OF GRADUATE STUDIES OF ADDIS ABABA UNIVERSITY
IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR
THE DEGREE OF MASTER OF SCIENCE
IN COMPUTER SCIENCE

JULY 2007
ADDIS ABABA

Acknowledgements

First and for most, I would like to express my heartfelt gratitude to my advisor, Dr. Dawit Bekele, for being by my side with his constant guidance, encouragement, comments and suggestions in the preparation of this thesis. It is with him that I first learned how to conduct and write a research work.

My next special appreciation goes to all fellow classmates and staff members of the Department of Computer Science for their encouragement and cooperation in the course of my graduate study. I am also grateful to my friends Daniel Assefa, Fitsum Meshesha and Surafel Lemma for their friendly encouragements and providing me with useful reading materials for this work.

Finally, I would like to thank my parents for their inspirational encouragement and moral support throughout my study. Thank you all for making me a better person.

Table of Contents

ACKNOWLEDGEMENTS II
TABLE OF CONTENTS III
LIST OF FIGURES V
LIST OF TABLES VI
LIST OF APPENDICES VI
ABSTRACT VII

1. INTRODUCTION 1
1.1. BACKGROUND 1
1.2. MOTIVATION 4
1.3. STATEMENT OF THE PROBLEM 5
1.4. OBJECTIVES 5
1.5. SCOPE 6
1.6. LIST OF ACTIVITIES 6
1.7. THESIS ORGANIZATION 8

2. CONTEXT-AWARENESS AND PERVASIVE MEDICAL SYSTEMS 9
2.1. OVERVIEW 9
2.2. DEFINITION, CHARACTERISTICS AND CLASSIFICATION OF CONTEXT 9
2.3. CONTEXT-AWARE SYSTEMS: DEFINITION, FUNCTIONS AND EXAMPLES 12
2.4. CONTEXT MODELING AND REASONING IN PERVASIVE SYSTEMS 14
2.4.1. Context Modeling 14
2.4.2. Context Reasoning 18
2.5. CONTEXT INFORMATION MANAGEMENT AND PERVASIVE MEDICAL SYSTEMS 20
2.6. CONTEXT INFORMATION REFINEMENT 22

3. RELATED WORKS 25
3.1. APPLICATION OF CONTEXT-AWARE COMPUTING IN HOSPITAL WORK 25
3.2. QUALITY OF CONTEXT: WHAT IT IS AND WHY WE NEED IT? 27
3.3. TOWARDS CONTEXT-AWARE COMPUTING IN CLINICAL CARE 28
3.4. IMPLICIT, CONTEXT-AWARE COMPUTING FOR HEALTHCARE 29
3.5. CONTEXT BROKER ARCHITECTURE FOR CONTEXT-AWARE PERVASIVE SYSTEMS 31

4. PROPOSED CONTEXT INFORMATION REFINEMENT ARCHITECTURE.....32

4.1. OVERVIEW OF THE PROPOSED ARCHITECTURE 33
4.2. DETAILS OF THE PROPOSED ARCHITECTURE 35
4.2.1. The Service Parameters Specification Interface (SPSI) 36
4.2.2. The Event Listener (EL) 38
4.2.3. The Device Tracker (DT) 38
4.2.4. The Service Parameters Manager (SPM) 39
4.2.5. The Reasoning and Decision Engine (RDE) 40
4.2.5.1. The Interpretation Component (IC) 40
4.2.5.2. The Decision Component (DC) 42
4.2.6. The Context Acquisition Component (CAC) 44
4.3. SUMMARY OF THE PROPOSED ARCHITECTURE 45

5. PROTOTYPE IMPLEMENTATION AND DEMONSTRATION 46

5.1. OVERVIEW 46
5.2. TOOLS AND TECHNOLOGIES UTILIZED FOR IMPLEMENTATION 47
5.2. PROTOTYPICAL SETUP 48
5.3. IMPLEMENTATION DETAILS 49
5.3.1. Modules on the Client Mobile Device 49
5.3.2. Modules on the Context Refinement Server 50
5.4. DEMONSTRATION 53

6. CONCLUSIONS AND FUTURE WORK 56

REFERENCES 58

APPENDICES 63

LIST OF FIGURES

Figure 2.1: CC/PP for Nokia 2160 Phone [13] 15

Figure 2.2: An Example Object-Relational Context Model [7] 17

Figure 2.3: RDF/OWL Ontology Example for Location Context Model [13] 18

Figure 2.4: Example on Low-level and High-Level Context 19

Figure 2.5: Layered Context Management in Context-Aware Pervasive Systems, Partially adapted from [8] 21

Figure 2.6: Context Information Refinement Scenario 23

Figure 3.1: Context-Aware Scenes in a Hospital [18] 26

Figure 3.2: Context Management Sub-system [19] 28

Figure 3.3: Architecture for Context Management System [5] 30

Figure 4.1: The Proposed Context Information Refinement Architecture 33

Figure 4.2: Internal Detail of the Reasoning and Decision Engine 43

Figure 5.1: Class hierarchy for the developed pervasive healthcare ontology 52

Figure 5.2: Mobile device interface for service parameters specification 53

Figure 5.3: Automatic notification on the mobile device pertinent to user's location 54

Figure 5.4: Automatic notification of abnormal vital signs reported from sensors 55

LIST OF TABLES

Table 2.1: Logic-Based Context Modeling Example 16

Table 4.1: Major Components in the proposed Architecture 34

Table 4.2: Parameter Types in the Service Parameters Specification Interface 37

Table 4.3: Examples of Reasoning Rules 41

LIST OF APPENDICES

A: J2ME MIDlet for Event Listener and Device Tracker on Mobile Device 63

B: Java Servlet Class for the Reasoning and Decision Engine 66

C: Protégé OWL Ontology Code for Pervasive Healthcare Domain 69

Abstract

In pervasive medical systems with context-aware computing facility, the quality of decisions made by medical professionals is influenced by the quality of context information supplied by the context management sub-system. Central to context management is context information refinement aimed at deriving context information that can assist applications in making valuable decisions about what to deliver to users. In this thesis, we identified the shortcomings of existing works in relation to context information refinement in pervasive medical systems. The shortcomings are lack of adequate consideration for: quality parameters of context information, relevance of context information and particular requirements of the pervasive healthcare domain.

In order to overcome these shortcomings, we proposed a context information refinement architecture that facilitates and coordinates the refinement procedure starting from acquisition of context information up until the refined context information is delivered to the target application (user) in a pervasive medical system by addressing the above-mentioned shortcomings. The architecture is composed of the client mobile device end and the context refinement server. The client mobile device end consists of components responsible for tracking the device context (E.g. location), listening to incoming events from the server, providing interface for specifying service constraints and local caching services. On the context refinement server side, the major components are the reasoning and decision engine that performs ontology-supported context reasoning, the service parameters manager that maintains up-to-date list of service constraints, the context ontology that models the concepts and relationships between concepts in the pervasive healthcare domain and the context acquisition component that collects and aggregates context data from potential context data sources like sensors.

To demonstrate the validity of the proposed architecture, we developed a prototype that implements the core components of the proposed architecture. The implementation has been evaluated with a real-life pervasive healthcare scenario and encouraging initial results have been obtained as an indication to the usability of the proposed architecture in a real-life setting.

1. Introduction

1.1. Background

After 1950s, Computing had gone through the mainframe era (from the early days of computing to 1970s) and the personal computing era (from 1980s to the end of the 20th century) [15]. In 1991, Mark Wieser predicted a third era of computing for the 21st century, which he called Pervasive computing. He described it as a new computing trend in which computing systems are seamlessly integrated into the life of everyday user until they are indistinguishable from it [14]. Pervasive computing, also called ubiquitous or ever-present computing, uses the increasing number of mobile computing devices available to provide the information and services needed by the users anywhere, on various devices and at any time.

Pervasive computing is characterized by four fundamental principles: Decentralization, Diversification, Connectivity and Simplicity [15]. Decentralization refers to distributing computing power on the various small devices with synchronization of information and remote management of applications. Consequently, computing is no more restricted to desktop computers in offices. Rather, it is possible to undertake computing being anywhere and anytime. Diversification comes with heterogeneous hardware (E.g. personal digital assistants-PDAs, mobile telephones, smart phones, laptops and desktops) and software variations (E.g. operating systems) of the various devices. The devices may vary in terms of dimensions such as processing speed, storage capacity, display capability, input capability, communication capability, user interface capability and power consumption. The third principle of pervasive computing is the strong demand for anytime connectivity from anywhere. In order to realize such a connectivity, it is crucial to ensure the interoperability of various communication technologies and protocols through an agreed upon standards. Simplicity is an equally important issue in pervasive computing as the diverse number of devices and the corresponding applications could be used by people ranging from experts in computing to those who are perfectly non-experts. Hence, providing intuitive and easy-to-use devices and/or applications will incorporate every extreme of the user community.

Recently, pervasive computing has got a significant attention by the research community. The new paradigm created by pervasive computing is attracting a large number of researchers from many application domains such as e-Commerce, e-Government, and e-Health, to mention few [15]. In particular, pervasive computing has tremendous applications in the medical field for two basic reasons. First of all, the workflow of medical professionals is usually characterized by high degree of mobility. The second reason is that medical professionals have high demand of getting the right information at the right time and place so as to facilitate their day-to-day activity.

One specific feature of pervasive computing that is particularly interesting for medical field is context-aware computing whose primary concern is to provide information relevant to the current context (location, time, activity, and environment) of a user and/or application. For instance, a certain user has three telephone lines: office, home and cell phone. The cell phone can also send and receive voicemail messages. All the lines are interfaced with a call forwarding system. In one occasion, the office phone of the user is ringing while the user has already left his office. Then, the system detects that he is on his way to home and hence forwards the call to his cell phone. Here, location is utilized as context. On another circumstance, the system detects that the user is in a meeting when his office telephone rings. The system forwards the call to his voice mail. This is an example where current activity is utilized as context.

In the last few years, several researches have focused on the management of information in pervasive systems in general and pervasive healthcare systems in particular as an enabling input to the delivery of relevant, up-to-date and above all reliable content. However, the management of context information remains a challenge due to its inherent dynamic nature as a function of time, location, current activity and the surrounding environment, to mention the major parameters. This dynamicity is more outstanding when it comes to pervasive healthcare systems, as the workflow of medical scenarios exhibits high degree of mobility.

In effect, a suitable context management technique has not been found. Thus, it sounds reasonable to closely investigate better techniques of managing context information for specific areas such as pervasive medical systems that deal with context information leading to life-critical decisions. Due to the high degree of mobility and frequent demand for information at the point of decision, healthcare professionals rely on the information they receive from the underlying computing infrastructure to pass each decision about their patients. Therefore, the quality of decision made by medical practitioners is influenced by the quality of context information received from the underlying context-aware pervasive computing facility [1], [5], [18].

Central to context information management is the refinement of information collected from several context sources such as sensors, context history servers and databases. Refinement[1] of context information is aimed at filtering the most relevant and high quality context information based on which users can make valuable decisions. For example, suppose that you are in a supermarket and wondering of what to buy for tonight. Probably, you are overwhelmed by several items in the store and unable to refine your preference. Suddenly, a shopping assistant system on your PDA detects that you are at a supermarket and alerts you with a relevant list of items that you should buy based on your personal preference.

In the existing context information management models, the major focus is on how to represent, classify and interpret context information. However, apart from these ones, there are critical issues to be addressed in context information refinement that will play vital roles in providing context information with the right quality and relevance while considering the nature of the domain. This thesis work is concerned with the refinement of context information in pervasive medical systems with a particular emphasis on quality and relevance of context information and the nature of the domain by taking the existing context information management models as a foundation.

[1] A more detailed definition of context information refinement is presented section 2.6 of chapter 2

1.2. Motivation

Since pervasive computing is touching several domains of applications, there are many issues that need thorough investigation in order to come up with solutions that facilitate the day-to-day activities of users. In the medical domain, pervasive computing comes with tremendous opportunities one of which is context-aware computing service provision to patients and physicians [31]. One important issue in context-aware computing is context information refinement in pervasive medical systems. As a result of the inherent mobility of medical work and demand for the right information at the right time and place, the quality of decision passed by medical professionals relies on context information received from the underlying computing facility [1], [5]. The quality of context information in turn is determined by the refinement procedure applied to information gathered from the underlying context sources. The context refinement procedure should, as much as possible, incorporate the necessary issues that may have consequences on final context information to be delivered to the user and/or application.

Among the important issues with regard to context refinement, representation of context, classification of context, reasoning of context, quality of context, relevance of context and the degree to which the characteristic of the domain is addressed by the refinement are the major ones. Representation of context is concerned with modeling context information so as to make it convenient for machine processing [8]. Classification of context deals with grouping context information types of similar nature based on a certain grouping criteria [11]. Context information can either be static or dynamic depending on how often it changes overtime. Alternatively, it can also be classified as sensed, user-supplied, inferred or profiled (historical) based on its source. Reasoning of context is concerned with derivation of new knowledge from an existing knowledge [16]. Quality of context is the extent to which the context information is of worth for a particular purpose of use [6]. Relevance of context is mainly concerned with the extent to which context information is applicable for a particular situation in addition to its quality [6]. The domain sensitivity is all about whether the context refinement technique is suitable to consider the distinctive

characteristics of the domain. The first three issues are already considered by most context information management approaches while the last three are not incorporated in most of the existing models. Therefore, it is interesting to incorporate these issues in the context refinement model in such a way that it is useful for pervasive medical systems. Hence, here comes the subject of this thesis work.

1.3. Statement of the Problem

In a pervasive healthcare environment with context-aware computing facility, the decision made by medical personnel relies on the quality of information provided by the context management system. The quality of context information is in turn determined by the refinement technique applied to it before the actual delivery. As a result, context information plays an important role in pervasive healthcare systems. Hence, its refinement turns out to be an important issue.

The core theme of this research work is realization of context information refinement model that is aware of issues that have not been addressed by the existing models. These issues are quality of context, relevance of context and the dynamic and life-critical nature of the pervasive medical systems domain.

1.4. Objectives

General Objective

The general objective of this research work is to come up with a model for context information refinement that takes into account the dynamic nature of pervasive healthcare environment in addition to the quality and relevance of context information.

Specific Objectives

The specific objectives of this research work are stated as follows:

- Studying existing context information management techniques in pervasive computing systems in relation to pervasive medical systems.

- Identification of a refinement technique that better suits to the dynamic nature of context information in pervasive medical systems.
- Developing a prototype that demonstrates context information refinement in pervasive medical systems.

1.5. Scope

In general, context information incorporates location (where), identity (who), time (when), current activity (what) and the surrounding environment such as which device or resource is nearby a given device (and hence a user). However, only location and identity are considered in the proposed context information refinement architecture of this work. Nevertheless, incorporating the left out constituents of context information can be realized if a thorough investigation is conducted on current activity detection and identification of surrounding environment like nearby devices and users. Moreover, even if security and privacy of context information are important topics in context information refinement, this work does not go into the detail of these issues, as the topic is very wide and demands to undertake a separate exploration.

1.6. List of Activities

In order to achieve the general and specific objectives mentioned before, we employed different methods in three consecutive phases of this work. In the first phase, review of the relevant literature was conducted to gain deeper understanding of the research domain. To have a better and solid understanding of the problem domain, reading of books, journals and research articles relevant to the research topic were undertaken. The major activities carried out in this phase were the following:

- Major researches in the area of pervasive computing have been studied.
- Context-aware computing and related issues in pervasive environments in general and pervasive medical systems in particular have been investigated.
- Existing context management models were studied with an emphasis on pervasive medical systems since one of the major issues in context-aware pervasive computing is context information management

- The major issues in context information refinement (E.g. quality parameters of context information, impact of refinement on the quality of context information) were studied.

The second phase of this work was to develop a model for context information refinement based on the inputs from the first phase. The major activities undertaken in this phase were the following:

- Specifying the requirements of the model to be developed based on the study made in the first phase.
- Development of the model as per the outlined requirements. This involved:
    - Identifying the various components of the model.
    - Defining the internal details of each component.
    - Defining the means and scope of communication among the components.

The third and last phase of this work was to design and develop a prototype so as to demonstrate the validity of the proposed context information refinement architecture. Both the preceding phases provided valuable inputs to this phase. The following major activities were conducted in this phase:

- Specifying the requirements of the prototype.
- Studying tools, technologies and protocols needed for implementing the prototype.
- Developing design of the prototype as per the specified requirements.
- Implementation and testing of the prototype design.

1.7. Thesis Organization

The rest of this thesis report is organized into five chapters. Chapter two presents introductory background on context-awareness and pervasive medical systems that will be a foundation for the subsequent chapters. In this chapter, definition of context information, classification and characteristics of context information, modeling (representation) and reasoning issues of context information, context-aware systems and the characteristics of pervasive medical systems in relation to context-awareness and the issues related to context information refinement are discussed. Chapter three covers the summary of related works, with respect to this work, which goes through works of other researchers with regards to their purpose, achievements, limitations and lessons learned. The fourth chapter presents the proposed context information refinement architecture for pervasive medical systems with a detailed explanation of the purpose, the internal detail and constraints of each component, and their association with the rest of the architecture and constraints on each component. We have developed a prototype implementation as a proof of the concepts introduced in the architecture discussed in chapter four. The detail of the prototype is presented in the fifth chapter by incorporating the description of tools used in developing the prototype, the prototypical setup, demonstration of its functionalities and the results obtained when testing the prototype. Finally, the sixth chapter wraps-up the whole thesis and points out possible rooms of improvement in the future.

2. Context-Awareness and Pervasive Medical Systems

2.1. Overview

In this chapter, topics that will set landscape for the rest of the thesis will be presented in order to acquaint the reader with the important terms and concepts that may be used in different sections of the upcoming chapters. To this end, the subject of this chapter includes definition, characteristics and categorization of context and context-aware systems followed by summary of well-known techniques for modeling (representation) and reasoning of context. In the remaining sections, pervasive medical systems, management aspects of context information in relation to pervasive medical systems, context information refinement and related issues are presented in sequence.

2.2. Definition, Characteristics and Classification of Context

The term context has been re-defined several times by several researchers since its first introduction in 1992 by Schilit and Theimer [4]. In the early definition, the originators associated context with the location of use, collection of nearby people and objects, as well as modifications to those objects overtime. Many of the definitions have focused on elements, which can be inferred from the user's physical movement through sensor-based technology (E.g. location and identity), in addition to time and environment (E.g. network capacity, network connectivity, and communication cost). The most commonly accepted definition of context as given in [1], [4], [5], [11] and that we take as a working definition throughout this thesis is the following:

Context is any information that can be used to characterize the situation of entities (E.g. person, place or object) that are considered relevant to the interaction between a user and an application including the user and the application themselves.

As per the definition, if a piece of information can be used to characterize the situation of a participant in an interaction, then that information is considered as context. To illustrate this definition, let us consider an indoor mobile tour guide as an example. The obvious entities in this example are the user, the application and the tour sites. Consider two pieces of information, i.e., weather and the presence of other people. The weather does not affect the

application because it is being used indoors. Therefore, it is not context. The presence of other people, however, can be used to characterize the user's situation. If a user is traveling with other people, then the sites they visit may be of particular interest to him/her. Therefore, the presence of other people is context because it can be used to characterize the user's situation. [11]. From now on, we will be using context and context information interchangeably as it is commonly done in other literature [9], [11], [13].

The following are the common properties with which context can be characterized: [3]

- **It exhibits range of temporal characteristics**, i.e., context may change significantly over time or may stay relatively stable depending on the nature of the context.
- **It is often imperfect**. A certain context that is claimed to characterize a particular situation may not completely describe that specific situation. Such imperfectness may arise from limitation on the underlying technology for context collection and /or representation. For example, in a certain application that gives restaurant-finding service, the service tells you the name of the best restaurant as per your already specified preferences. However, you do not know where the restaurant is located. Such context information is imperfect, as it does not provide direction to the suggested restaurant.
- **It has many alternative representations.** For instance, location can be alternatively represented using latitudinal and longitudinal coordinates, street numbers, building names or numeric values.
- **There is high degree of inter-relationship between different context types.** For example, the current location of a user may be of significant indication to predict the user's current activity based on the user's profile.

In addition to the above-mentioned major characteristics, it is critical to define quality of context (QoC) because there are distinctive parameters used to formulate QoC and its implications on the final context information consumed by applications. Before discussing the quality parameters of context, let us define what QoC means. It refers to any information that characterizes the worth of information that is used as context information.

Hence, QoC actually refers to information and not to the process or the hardware component that possibly provides the information [6].

The following are the major parameters that contribute to the complete characterization of quality of context information: [6], [7]

- **Accuracy:** It describes the extent to which the provided context information mirrors the reality. For instance, a sensor reading gives a temperature of a patient to be 150 degree Celsius. In reality, such a temperature reading can never indicate existence of a patient and therefore is not in agreement with the reality.
- **Probability of Correctness:** It denotes the probability that a piece of context information is correct. This parameter can be linked with the well being of the context information source like a temperature sensor.
- **Trust-worthiness:** It describes the level of trust associated with the context information source. For example, let us assume that there are two sources (A and B) of context information and A has 70% of trustworthiness while B has 99%. At the receipt of context information of the same type from A and B, that of B is preferred as the context information receiver has better trust on source B.
- **Resolution:** It denotes the extent to which the information is fine-grained (granular). For example, a certain temperature sensor at one corner of a room may report 25 degree Celsius while there is a big toaster at the opposite corner of the room where no temperature sensor is available. In such a case, the actual temperature would have been more granular if there was a sensor around the toaster.
- **Freshness:** It describes the age of context information that is specified by attaching timestamp to context information at time of production. Thus, clock synchronization between the context producer and context consumer is a necessity as it is likely to encounter context information of distributed entities.

Context can be classified taking into account its temporal characteristics and source of production. There are two classes of context, i.e., static and dynamic depending on its temporal characteristics overtime [3]. A context that does not change significantly over time is called static (E.g. personal information) while context that changes frequently over a given time range is called dynamic (E.g. location, activity). On the other hand, user-

provided, interpreted, sensed and profiled (historical) context are the four types of context if source of production is taken into consideration. With this background on the definition, characteristics and classification of context, let us move on to the definition, functional classification and common examples of context-aware systems, which is the subject the next section.

2.3. Context-Aware Systems: Definition, Functions and Examples

Before we describe the distinctive features and some legendary examples of context-aware systems, it is reasonable to define the term context-awareness or what is meant by such a system. The following are two definitions that are repeatedly given in the existing literature:

A system is said to be context-aware if its operations and services can be adapted to the current context without explicit user intervention and thus aims at increasing the usability and effectiveness by taking environmental context into account [8].

Another alternative definition is presented as follows:

A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the extent to which it anticipates users' needs and acts in advance by "understanding" their context [11].

To illustrate the above definitions, consider a seminar is being conducted in a certain hall of a university building and you have registered to be a potential attendant. However, you do not need to attend all the presentations, i.e., you need the system to notify few minutes before the commencement of a presentation of your preference. At a particular time, the system identifies the current presenter, current audiences, current presentation and more importantly that the ongoing presentation is of particular interest to you and hence you are recommended by the system to attend it. This is not a generic example, but can give an imaginative picture of a context-aware system.

One of the very first context-aware systems is called "Active Badge Location System" developed in 1992 by Want et al [32]. It is an infrared technology based system that is able to automatically determine the current location of a user so as to forward phone calls to a

telephone close to the user. In the middle of 1990s, several context-aware tour guides were developed to provide information as per the user's current location [8]. While location information is by far the most frequently utilized attribute of context in most of the context-aware applications, attempts to use other context information (E.g. current activity, nearby environment) as well have grown over the past few years [8].

There are three fundamental functional classifications[2] of context-aware systems. [12]:

- **Presenting Information and Services:** these are applications that present context information to the user or use context to propose appropriate services. Examples of such functions include presenting choice of nearby printers, nearby sites of interest like a supermarket, sensing and presenting the in/out status of a person in a group of users.
- **Automatic Execution of a Service:** such applications trigger a command or reconfigure the system on behalf of the user as per changes that happen to context. Among the examples in this class of functionality are teleport systems where the desktop environment follows the user as the user moves from one device to another, car navigation systems that recompile driving directions when a user misses a turning point and mobile devices equipped with sensors that can automatically change their setting (E.g. to vibration mode when the user is in a meeting).
- **Attaching Context Information for Later Retrieval:** these are classes of applications that tag captured data with relevant context information. Attaching context to desktop or networked files so as to enable easier retrieval later on is one example under this category of functionality.

[2] Although context-aware systems are classified based on the set of functionalities they provide, it is often common to observe an overlap of functionalities in some of the applications. This is an indication that such kinds of applications are very much inter-related and creating absolute boundaries is not easy.

2.4. Context Modeling and Reasoning in Pervasive Systems

2.4.1. Context Modeling

In order to facilitate the development of context-aware applications, an infrastructure that collects, stores and manipulates context information is important. Before storing context information, it should be abstracted in such a way that it is suitable for machine processing. Such an abstraction results with a model, i.e., a context model. The most popular context modeling approaches are based on the data structures used for representing and exchanging context information. In light of this, there are six context-modeling techniques that will be described below [8], [9], [13], [26].

Set Theory-Based Model: This model represents the simplest data structure for the purpose of context modeling. A context T is described by a set of two-dimensional vectors where each vector consists of a symbolic value v describing the situations and a number p indicating the certainty that the user (the device) is currently in this situation. The set is represented as $\mathbf{T = \{< v_1, p_1>, < v_2, p_2>, < v_3, p_3>, ..., < v_n, p_n>\}}$ where n is the number of distinct situations. In this approach, even though context is represented using a mathematical model, the dependency or inter-relationship between context information, which is crucial for context reasoning, is missing.

Markup Scheme Model (Preferences and User Profile): All markup-based models use a hierarchical data structure called profile consisting of components with attributes and content (see Figure2.1) used to describe device capabilities and user preferences. Typical examples for such profiles are the W3C[3] recommended Composite Capabilities / Preference Profile (CC/PP) and the WAP (Wireless Application Protocol) forum recommended User Agent Profile (UAProf), which are encoded in the Resource Description Framework/Schema (RDF/S)[4]. One of the major drawbacks of such models is that

[3] W3C (World Wide Web Consortium) is a group that develops interoperable standards, guidelines, software and tools under the objective of utilizing the World Wide Web to its full potential. http://www.w3.org

[4] RDF is a general-purpose language used to represent metadata, i.e., a structured data about data, on the web. [17]

representation of complex relationships becomes difficult as the hierarchy grows with nested components.

```
<?xml version="1.0"?>
 <rdf:Description about="HardwarePlatform">
 <prf:Defaults  Vendor="Nokia“  Model="2160“  Type="PDA"
   ScreenSize="800x600x24“ CPU="PPC"  Keyboard="Yes"
   Memory="16mB"  Bluetooth="YES" Speaker="Yes" />
 <prf:Modifications  Memory="32mB" />
 </rdf:Description>
<rdf:Description about="SoftwarePlatform">
 <prf:Defaults    OS="EPOC1.0"  HTMLVersion="4.0"
  JScriptVersion="4.0"WAPVersion="1.0"WMLScript="1.0"/>
 <prf:Modifications  Sound="Off"  Images="Off" />
 </rdf:Description>
 <rdf:Description about="EpocEmail1.0">
 <prf:Defaults  HTMLVersion="4.0" />
 </rdf:Description>
<rdf:Description about="EpocCalendar1.0">
<prf:Defaults HTMLVersion="4.0" />
 </rdf:Description>
<rdf:Description about="UserPreferences">
<prf:Defaults  Language="English"/>
</rdf:Description>
</rdf:RDF>
```

Figure 2.1: CC/PP for Nokia 2160 Phone [13]

Object-Oriented Model: Modeling context by using object-oriented techniques allows using the majority of object-orientation principles such as encapsulation, reusability and inheritance. Furthermore, UML (Unified Modeling Language) is also another suitable tool used to represent context. Existing approaches use various objects to represent different context types (E.g. temperature, location) and encapsulate the details of context processing and representation in an object. Access to the context and the context processing logic is provided via well-defined interfaces. Although such models have attractive capability for modeling context as a collection of classes and defined interfaces, they lack expressiveness that hinders the interpretation of context.

Logic-Based Model: Typical to such context models is that facts, expressions and rules are used to define a context model. A logic-based system is then used to manage these terms and allows adding new, modifying or removing of existing facts. The inference process can be used to derive new facts based on existing rules in the system. The context information needs to be represented in a formal way as facts. One popular way to abstract context in such a model is in the form:

Context (<Context Type>, <Subject>, <Relater>, <Object>) where:

- Context Type is the class of context information like location and people,
- Subject is person, place or thing, with which the context is concerned,
- Object is a value associated with the subject and
- Relater is a comparison operator, verb or preposition used to apply logical operations.

Consider the following example (Table 2.1) that illustrates how logic-based representation helps to deduce valuable context given the rule: "At a certain point in time, if there are four or more people in the surgery room and the surgery toolkit application is running, there must be a surgical activity going on in that room".

Table 2.1: Logic-Based Context Modeling Example

Context (People,Room 205, >=, 4)	Number of people in room 205 is at least 4
Context(Application,SurgeryToolkit,Is, Running)	SurgeryToolkit application is running
Context (RoomActivity, 205, Is, Surgery)	The activity in room 205 is surgery

Object-Relational Model: Context information is structured around a set of entities (shaded rectangles in the next figure) each of which describes physical or conceptual objects such as a person, a device or an activity. The entities are associated with one another through a unidirectional association link. (See Figure 2.2) The limitation of this modeling technique is that relations between instances of context concepts are represented by means of value-based foreign keys, resulting in a large number of expensive join operations for context queries in large and information-intensive application domains.

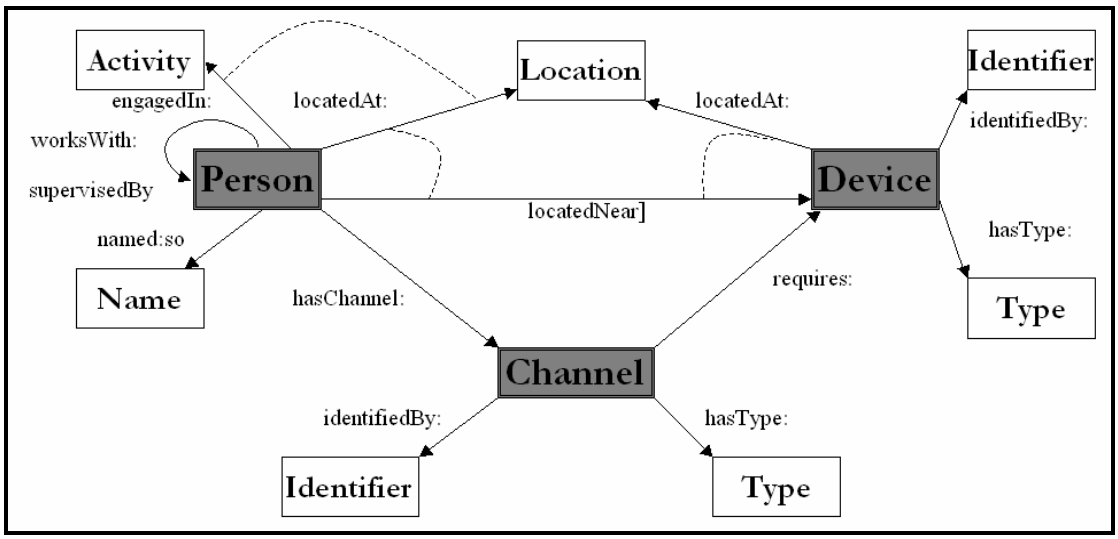

Figure 2.2: An Example Object-Relational Context Model [7]

Ontology-Based Model: Ontologies represent a description of the concepts, functionalities and relationships among the concepts and provide a common vocabulary that can be shared and reused among software agents and users [17]. As a result, ontologies are promising instruments for modeling context information in pervasive environments due to their high and formal expressiveness and possibilities for applying ontology reasoning techniques.

A typical ontology should be simple, flexible, extensible, generic and expressive. By simplicity, it means that the used expressions and relations should not be complicated so as to simplify the work of application developers. Flexibility and extensibility refer to the fact that the ontology should support introduction of new context elements and relations. Generality is concerned with the fact that the ontology should not be limited to special kind of context elements but rather support different types of context. Expressiveness is related to the extent to which the ontology allows to describe as much context states as possible in an arbitrary detail.

Several tools are available to define declarative representations, publish and share ontologies among which the W3C recommended RDF /OWL [5](Ontology Web Language) is the predominant one. Figure 2.3 shows an example RDF/OWL ontology model for location context.

```
<owl:Class rdf:ID="ContextEntity"/>
<owl:Class rdf:ID="Location">
  <rdfs:sudClassOf rdf:resource="#ContextEntity"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="longitude">
  <rdf:type rdf:resource="FunctionalProperty">
  <rdfs:domain rdf:resource="Location">
</owl:ObjectProperty>
<owl:Class rdf:ID="IndoorSpace">
  <rdfs:sudClassOf rdf:resource="#Location"/>
<owl:disjoinedWith rdf:resource="#OutdoorSpace"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="locatedIn">
  <rdfs:domain rdf:resource="#Entity">
  <rdfs:range rdf:resource="#Location">
</owl:ObjectProperty>
```

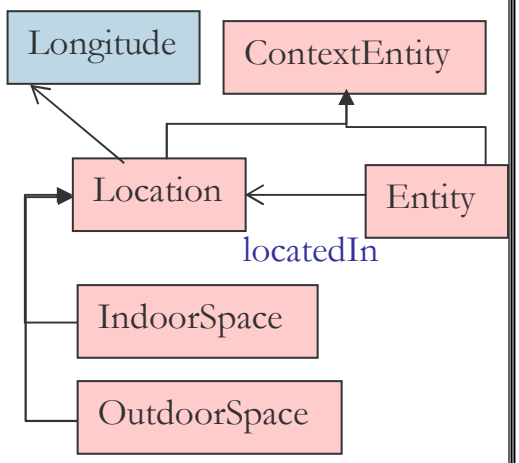

Figure 2.3: RDF/OWL Ontology Example for Location Context Model [13]

2.4.2. Context Reasoning

The comprehensive definition of context reasoning is that it is the task of deriving new knowledge from an existing knowledge. In context-aware pervasive systems, it refers to the task of deriving new and relevant context information to the use of application(s) and user(s) from a set of sources of context data [16]. The raw data that is supplied by the underlying context data sources is called low-level (implicit) context while the context information that is ready for consumption is called high-level (explicit) context. For example, GPS (Global Positioning System) coordinates expressed as (latitude, longitude) could be mapped into abstract locations such as At Home, At Work, At Hospital and so on. Context reasoning bridges the gap between the location coordinates and that of the abstract location names. Figure 2.4 illustrates this example further.

[5] OWL is a W3C recommended general-purpose language used for defining web ontologies. [17]

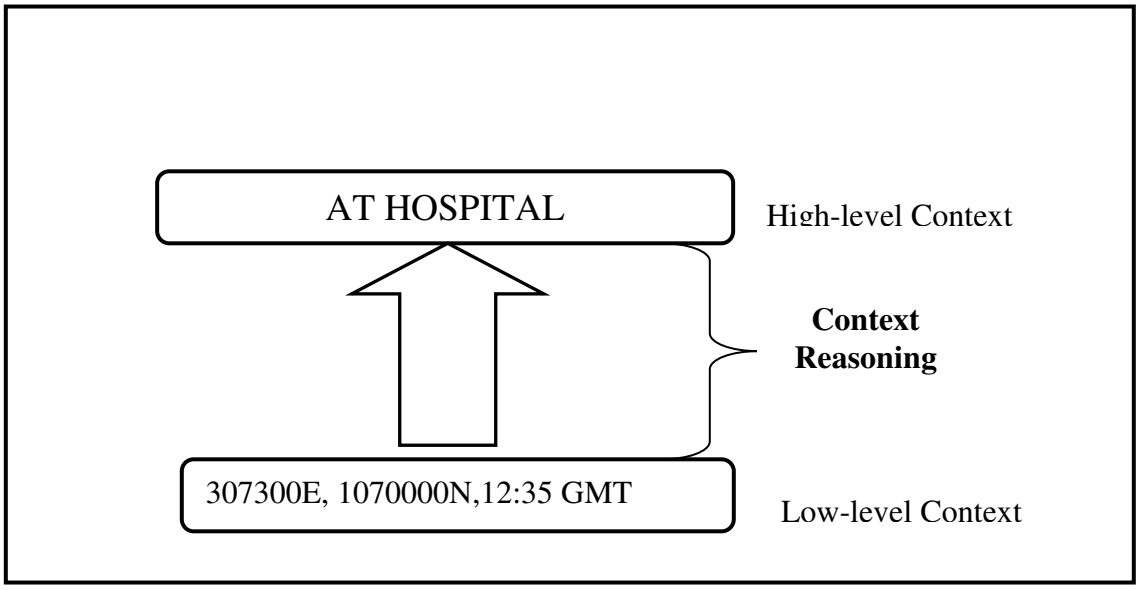

Figure 2.4: Example on Low-level and High-Level Context

There are two approaches for Context reasoning. The first and the most classical method is by hardwiring reasoning logic into context data source like sensors while the second and the most widely accepted for modern context-aware pervasive systems is by separating the reasoning logic from the context data representation and collection [16], [25]. The former one is not attractive for modern context-aware pervasive systems as it is difficult to extend the whole reasoning framework to a distributed and integrated architecture because the context data collection and reasoning are highly coupled. On the other hand, the later approach is in agreement with the nature of pervasive systems with a need to open architectures for scalability.

The specific reasoning technique ranges from simple IF-THEN kind of associating actions with conditions to more complicated rule-based and assumption-based reasoning techniques [16]. In rule-based reasoning, a set of rules will be defined in advance and these rules are taken as references before deriving any new context information. In assumption-based reasoning techniques, a set of assumptions is developed and a contradictory argument is made to derive new context information. The rule-based reasoning technique is the most widely employed one especially for context-aware pervasive systems where context information is reasoned in a layered manner and the reasoning requires complex procedures [25].

2.5. Context Information Management and Pervasive Medical systems

A typical medical work is accomplished by applying a wide range of documents, schemas and charts which all are tailored and structured according to the specific work setting in which they are used [3]. Furthermore, information is often retrieved and used according to a concrete usage situation. For instance, during the ward round at the patient's bedside, only the patient's medical record is used, whereas in a radiology analysis, x-ray images for all the patients in a ward are displayed on large light-screens. In addition, information is often located where it is mostly used or at point-of-care. For example, medical charts are primarily used to pour and document the medication given to patients, and are hence located in the medicine room. In general, the usage of medical information is highly dependent on the concrete usage situation – or in other words, is dependent on the context of use.

In pervasive systems in general and pervasive medical systems in particular, the various pervasive devices used have constraints in terms of their processing, storage, communication and input/output capabilities. Furthermore, the battery life is not that relaxed as compared to the ordinary desktops. On the other hand, the healthcare environment is an information intensive scenario where the large volume of medical information is beyond the limited capability of the pervasive devices in use [1]. As a result, a proper context information management scheme that aims at filtering information that is relevant at a particular situation of usage is essential. Therefore, the role of context-aware computing facility in medical systems with pervasive nature is immense.

For the sake of context information management, a layered approach is practiced in most of the modern context-aware pervasive systems [8]. Such an approach is described using the following generic framework (see Figure 2.5) that spans from context data sources (the bottom layer) to the consumption of context information by applications (the top layer).

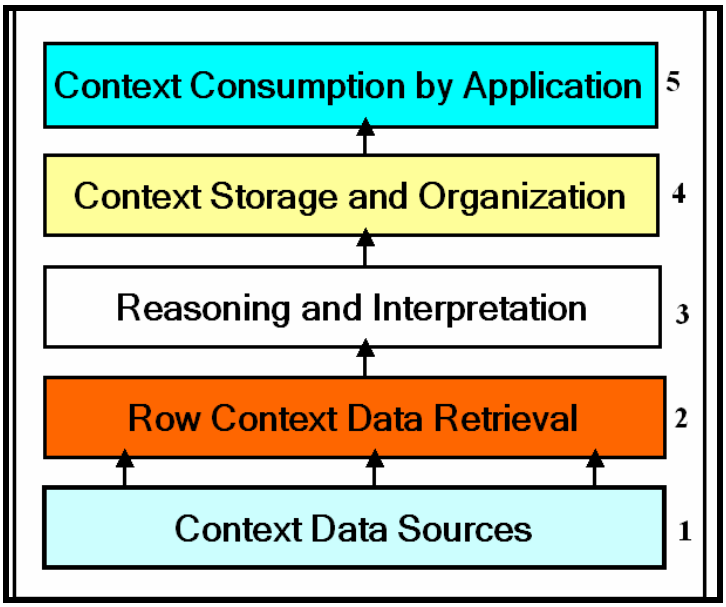

Figure 2.5: Layered Context Management in Context-Aware Pervasive Systems, Partially adapted from [8]

Layer 1 comprises the potential context data sources, mainly sensors and context history servers. The second layer is responsible for collection of raw context data by making use of appropriate drivers and APIs (Application Programming Interfaces) for the respective sensors and context history servers. Layer 3 is in charge of reasoning (discussed in section 2.4.2) and interpretation of context information. Because the raw context data are usually not in the form that applications can consume, it is the duty of this layer to transform, aggregate, and compose context information so that the next layer can store it. The fourth layer is dedicated to organize the gathered context information to client application in layer 5, via public interfaces. Depending on the nature of the clients and the requirements of the applications, clients may access the stored context information either by periodic polling based on some defined interval or by subscribing to set of events so that the client is notified or invoked on occurrence of one of the events.

2.6. Context Information Refinement

What is Context Information Refinement?

Context information refinement is defined as a process employed to generate concrete context information from attributes offered by available context sources [6]. By concrete, it means that context information that is ready for consumption by a certain application and its consumption leads to making valuable decisions by users. Moreover, the application and/or the user are not expected to participate in the process. Refinement comprises the transformation between different formats of representation (E.g. from GPS coordinates to street names and numbers), the extension of context information with attributes (E.g. to express its accuracy) and the combination of context information to derive another one (E.g. calculating the distance between a patient and an ambulance using their locations) [10].

In addition to the above-listed tasks, refinement also involves the task of categorizing, normalizing and pre-interpreting sensed context information that may result in accepting information that is in-context and rejecting otherwise. Categorization refers to grouping together of context information with similar nature (E.g. location related information, vital sign indicators, and patient's medical profile). Normalization is concerned with reducing the heterogeneity in terms of parameters such as representation formats, measurement units, precisions and timestamps that the various context data sources provide. In pre-interpretation, even if the sensed context information is found to be normal, a closer look at other related information may not encourage acceptance and hence delivery. A simple example is, if the patient and the doctor are co-located in the same room, it may not be important that the location of the patient is delivered to the doctor.

Hence, context information refinement is concerned with making the critical decision of whether a given context information is of significant importance for an application or not, right before delivering the information to the application, i.e., acting on behalf of the application so that the application relies on the refinement process than auditing the relevance of context information delivered to it. Figure 2.6 shows an example that illustrates the issues pertinent to context information refinement and how it is crucial in delivering relevant information with the right quality to applications.

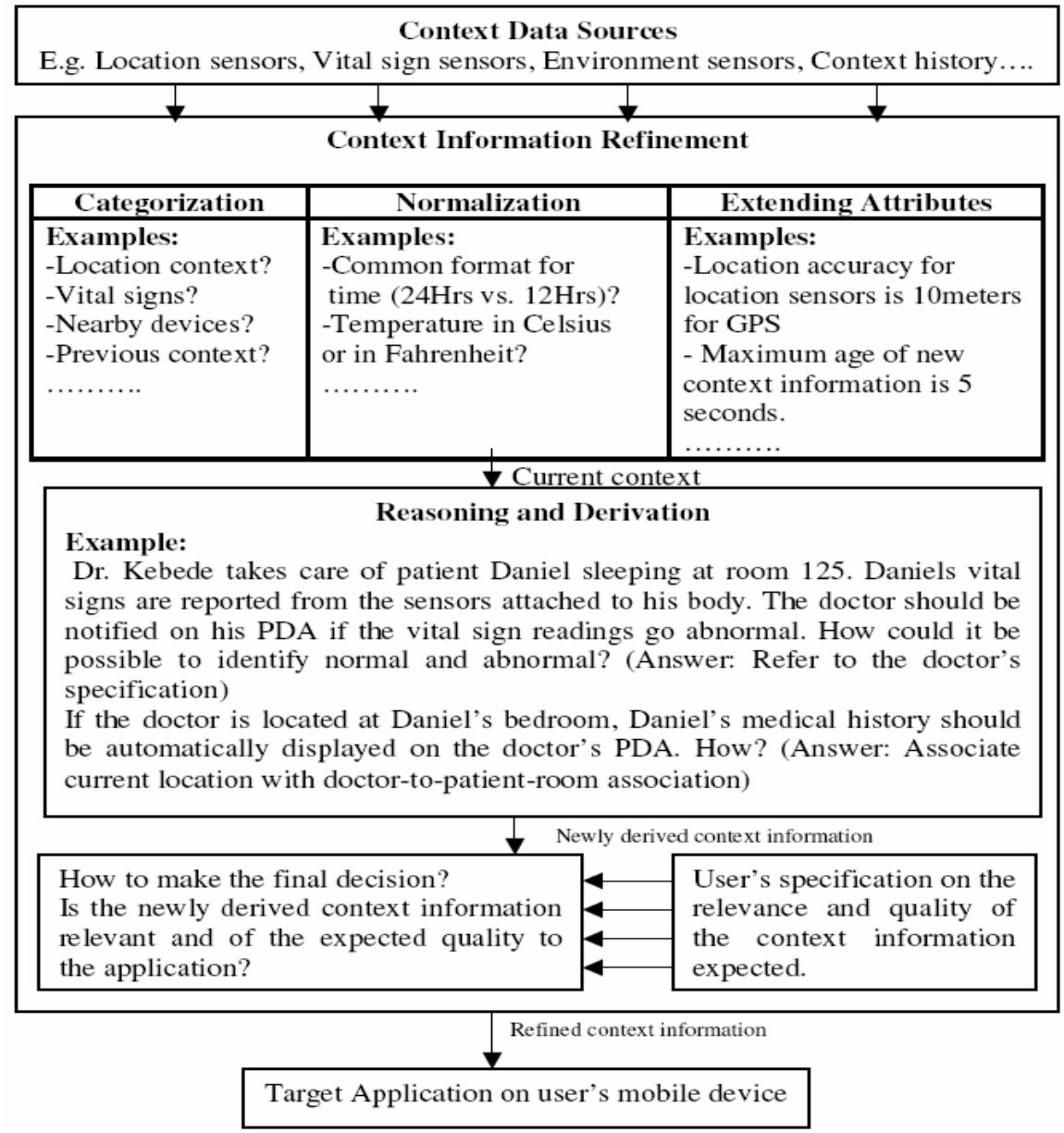

Figure 2.6: Context Information Refinement Scenario

How Complex is Context Information Refinement?

The complexity of context information refinement can be determined by several factors such as the underlying context sources (E.g. accuracy, representation format, degree of correctness, trust, and timeliness of sensed context), the demand of getting context information of a certain quality by the application domain and the underlying context management model [9].

In healthcare environments, information is often required at the point-of-care, for a particular patient, at a particular time. This indicates that the computing facility should have the capability to assist the medical practitioners obtain the right information being at their duty using mobile devices. This needs a careful consideration of additional parameters such as quality of context information, relevance of context information with respect to the current location, time, activity and the surrounding environment. Moreover, taking in to account the nature of the domain is an important parameter as every application domain has its unique requirements.

Summary

This chapter presented the definition, characteristics, classification, modeling, reasoning and refinement of context information. We also touched the meaning, functional categorization and common examples of context-aware systems. In addition, the typical context information management approach (layered approach) and the issues pertinent to context information refinement in the sense of pervasive medical systems are addressed.

3. Related Works

From the early 1990's, context information management in pervasive systems has been a subject of several research works like [5], [6], [18], [19], [20], [22], [23], [24] and [25]. These works vary in aspects like the software architecture they follow, the principal issues they address, the domain of application and the models they utilize for context modeling and reasoning. In this chapter, five of the related works are presented with an emphasis on their primary goals, the issues addressed, the gaps we have identified and the comparison of each to our work.

3.1. Application of Context-Aware Computing in Hospital Work[18]

The main goal of this paper is developing and deploying context -aware applications in hospital environments. It demonstrates four sample scenarios of using context-aware applications in a hospital. These scenarios are the entrance of a nurse to an active zone, i.e., where she comes to treat her patient, a context-aware bed, a context-aware pill container and a context-aware electronic patient record. The paper also outlines key design principles for a context- awareness framework, assisting in the development and deployment of context-aware clinical applications in hospitals.

Figure 3.1 below shows the diagrammatic illustration of the four context-aware scenarios demonstrated in this paper.

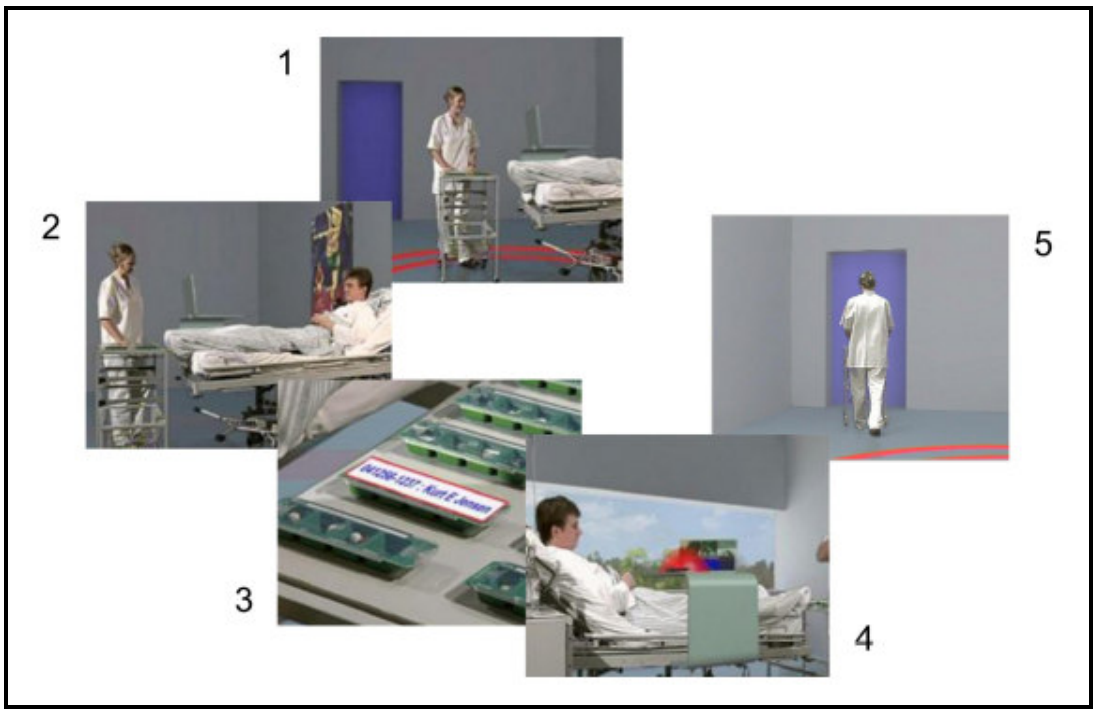

Figure 3.1: Context-Aware Scenes in a Hospital [18]

When a nurse enters an active zone (scene 1 in the above figure), the context-aware infrastructure automatically detects her and her patient. The context-aware hospital bed (scene 2 in the above figure) is equipped with a small display that is used by both physicians for accessing medical records when working at the bed and patients for entertainment purposes like television. The context-aware pill container (scene 3 in the above figure) has a computing power, communication capability and a display. It is also equipped with sensors that can detect whether or not a medicine is given to the patient as per a specified schedule. The context-aware electronic patient record (scene 4 in the above figure) communicates with the context-aware bed when arrival of the nurse is detected. Then after, it automatically logs in the nurse, fetches the patient's medical record and even highlights the prescribed medicines for the patient.

The other issue addressed in this paper is about how the context information assists in decision-making in the daily activities of the clinicians. Furthermore, critical design principles, one of which is the quality of context information, are listed to be very crucial in the course of providing context-aware pervasive computing facility in hospitals. This work is similar to our approach in that the context-aware infrastructure is specific to healthcare

environments and the distinctive characteristics of pervasive medical systems, such as mobility, are well considered in the design of the infrastructure. The context representation model is an object-oriented model that suffers from expressiveness power that hinders context reasoning. Our approach, however, uses ontology-based context representation that is known of its expressive power to facilitate context reasoning.

3.2. Quality of Context: What it is and why we need it? [6]

In this paper, the goal is to formalize issues that have implications on quality of context information. In particular, the important quality parameters that we mentioned in the previous chapter such as precision, probability of correctness, trust-worthiness, up-to-dateness and granularity of context information are described. The relationship and interdependence among QoC, QoD (Quality of Device) and QoS (Quality of Service) are also mentioned indicating that these three issues are different but inter-dependent. Furthermore, it is stated that context information refinement process is the determinant of QoC.

In order to provide context-aware services (CASs) in an inter-organizational manner, the paper formulates a role model for context-aware services with emphasis on the importance of refinement before any context-aware service attempts to deliver context information. The constituents of the role model are the CAS provider, CAS customer, CAS user, Context owner, and Context Provider. Central component to the role model is the CAS provider, which creates, deploys, offers and sells CASs to a CAS customer. The CAS customer negotiates with the CAS provider (about terms of uses for CASs) on behalf of one or more CAS users. The source of context for the CAS provider is the Context provider, which is mostly the operator of context sources. For instance, the context provider can be a cellular telephone network operator that tracks and provides current location of users to the CAS provider. The context owner is an entity (preferably an individual), to which context information belongs. The context owner has the right to put access restrictions on context information belonging to it.

The proposed model is different from our work in two aspects. Firstly, the model is domain neutral and has an inter-organizational vision, i.e., the context providers and owners are likely to be from different organizations. On the other hand, our work is domain specific and does not consider inter-organizational matters. Secondly, the role of the quality parameters formulated is not indicated in the model while we have considered the QoC parameters in the context refinement process.

3.3. Towards Context-Aware Computing in Clinical Care [19]

In this work, it is pointed out that various efforts have already been made in the course of developing context-aware system architectures and frameworks. Modeling, storage and retrieval issues of context information are explained in relation to context identification and inference. In the system's architecture (see Figure 3.2 below), the context management sub-system is one separate component along with context sensors and context actuators. It is composed of the Context Inference Engine that matches context patterns against context data in context base, Context Base as a repository of context information, Context Pattern as rules that govern the delivery of context information and Context Pattern Mining sub-component that analyzes the context data collected in the course of the system's operation to elicit usage patterns.

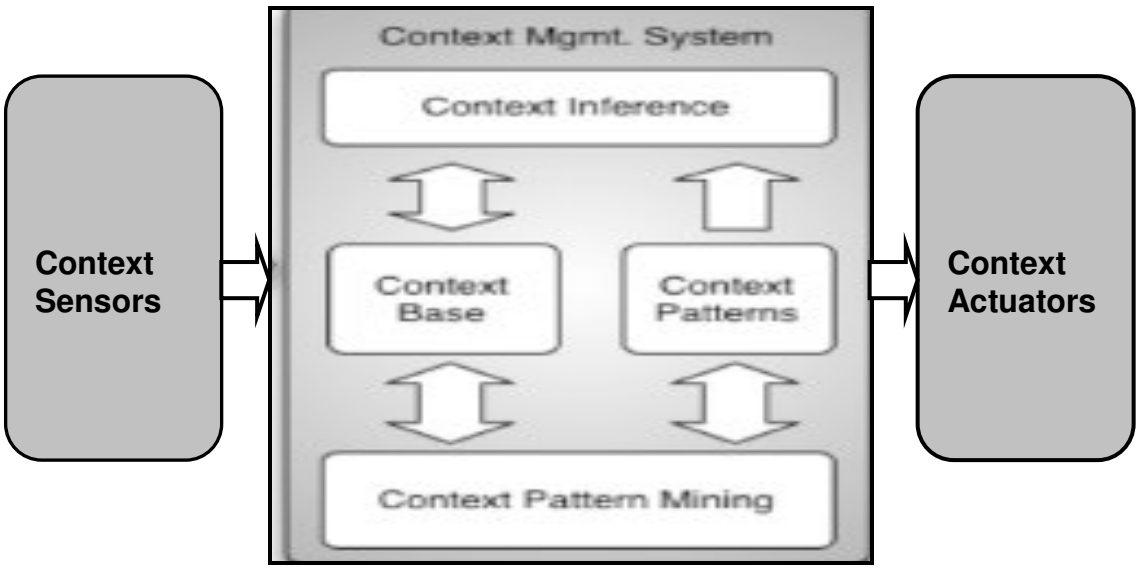

Figure 3.2: Context Management Sub-system [19]

The context information management portion is treated as a single sub-system in context-aware clinical systems, which opens slots for further enhancement of the context information management with a significant attention to the refinement of context information. Even if ontology-based context modeling is not utilized for the context management sub-system, it is recommended for enhancing the context inference.

3.4. Implicit, Context-Aware Computing for Healthcare [5]

In this work, the fundamental purpose is introduction of ontology-based context management in mobile health systems under the objective of reducing lack of quick and smooth interaction of medical professionals with the mobile devices. Making the service context-aware reduces this difficulty by minimizing the interaction of the user with the device.

The context management service (CMS) targets at correlating the predefined context information with the current situation of actors (like patients, physicians and nurses) so as to determine the information required to that actor. Central to the service is the context ontology that is used to represent the concepts, properties of concepts and the relationship between properties and concepts in the healthcare domain. Figure 3.3 shows the CMS with the major components.

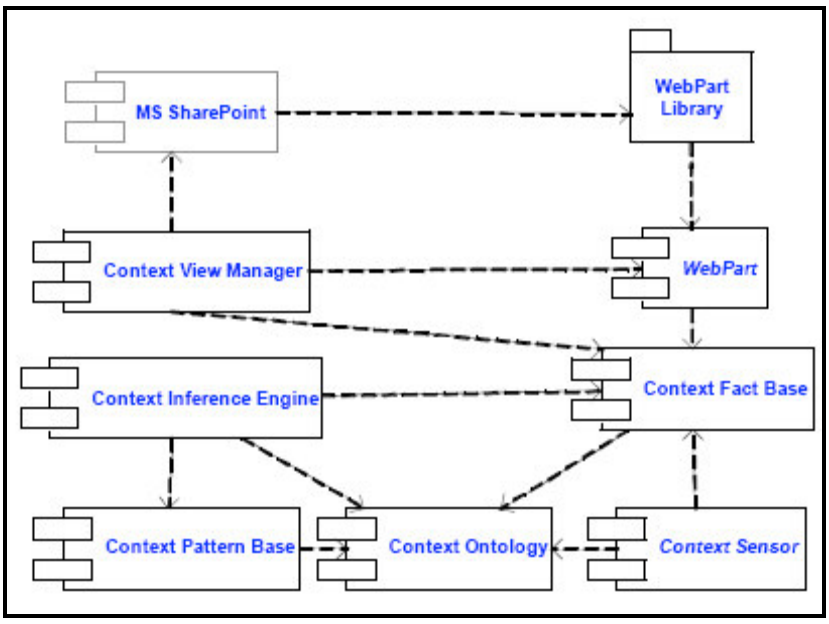

Figure 3.3: Architecture for Context Management System [5]

The whole service operates based on a commercial framework called SharePoint[6] that provides small sub-components of visual web pages called web parts that can be viewed with normal thin-client web browsers. The actual composition of context-aware web pages from Web parts is performed by an extension to SharePoint, called Context View Manager. At the core of the CMS are a Context Fact Base, a Context Pattern Base, and a Context Inference Engine. The Context Fact Base is graph-based database storing facts about situational parameters of user interactions with the information services provided by the system. Contextual facts can be delivered by a variety of different types of context sensors, including (but not limited to) location sensors. The Context Pattern Base stores rules that associate Web parts with formal definitions of pre-defined usage contexts. The Context Inference Engine infers actual usage contexts based on pre-defined context patterns and current data in the Context Fact Base.

This work resembles our work in three aspects. The first is that it utilizes ontology for context representation in pervasive healthcare even though the ontology is not fully utilized for context reasoning. The second similarity is the role of the Context Inference Engine in the architecture. It has similar roles to the Reasoning and Decision Engine of our context

[6] http://www.microsoft.com/sharepoint/default.mspx

information refinement architecture. The last resemblance to our work is the use of Context Fact Base. In our architecture, the equivalent component to the Context Fact base of this CMS architecture is the Context Knowledge Base. The principal difference between our work and this CMS architecture is that we considered the quality and relevance parameters of context information in addition to the domain specific characteristics addressed both in the CMS architecture of this work and our context refinement architecture through the context ontology.

3.5. Context Broker Architecture for Context-Aware Pervasive Systems

The other related work we discuss here is CoBrA (Context Broker Architecture) [25]. It is an agent-based architecture for supporting context-aware computing in intelligent spaces. According to the paper, intelligent spaces are defined as physical spaces like living rooms, vehicles, hospitals and meeting rooms that are populated with intelligent systems that provide pervasive computing services to users. Central to CoBrA is an intelligent context broker that maintains and manages a shared context model on behalf of a community of software agents[7]. These agents can be applications hosted by mobile devices that a user possesses, services that are provided by devices in a room (e.g., projector service in a meeting room) and Web services that provide a web presence for people, places and things in the physical world (e.g., services keeping track of location of people).

The context broker consists of four major components: the Context Knowledge Base, the Context Inference Engine, the Context Acquisition Module and the Privacy Management Module. The role of the Context Knowledge Base and the Context Acquisition Module are similar to the Context Knowledge Base and Context Acquisition Component in our proposed context information refinement architecture. The other similarity of CoBrA to our work is the use of ontologies for context representation and reasoning. However, CoBrA differs from our work in that it is a generic architecture for context-aware pervasive systems and it is agent-based while our context information refinement architecture is specific to healthcare domain and does not make use of software agents.

[7] A software agent is a piece of autonomous or semi-autonomous proactive and reactive, computer software.[25]

4. Proposed Context Information Refinement Architecture

After studying the layered approach for context information management in pervasive systems, we envisioned a domain-specific model developed under the objective of incorporating the three important issues that the existing works discussed in the previous chapter have not addressed, i.e., quality of context, relevance of context and the particular requirements of pervasive medical systems. It is important to have a dedicated architecture for pervasive medical systems with an underlying context-aware computing facility due to the inadequacy of existing similar architectures in incorporating the stated additional parameters that play decisive roles towards the provision of context information that is of the anticipated quality, with the anticipated relevance to the current situation of a user and is in agreement with the specific domain requirements.

As a result, we propose context information refinement architecture to refine context information in pervasive medical systems in such a way that the issues stated above are addressed. The architecture is composed of different components and interfaces located on the client mobile device and the context refinement server. These components cooperate and communicate in order to effectively assist the user in getting the right (relevant) information with the right quality, at the right time and place on his/her mobile device, without involving extensive interaction[8] with the device. Therefore, the key role played by this architecture is to facilitate and coordinate the refinement and delivery of context information pertinent to the current situation (e.g. location, identity, activity and surrounding environment), quality and relevance preferences set by the user and the requirements of the application domain, i.e., pervasive healthcare.

[8] The vision of context-aware pervasive computing is to obtain context information automatically, i.e., there would be no need for manual acquisition. In reality, absolute avoidance of user interaction is impossible in sensitive domains like pervasive medical systems. In effect, applications must rely on the user to manually provide it.

4.1. Overview of the Proposed Architecture

In the proposed architecture (see Figure 4.1), the two major constituents are the client mobile device and the context refinement server each of which is composed of several components that are needed to carryout different activities in the course of refining context information.

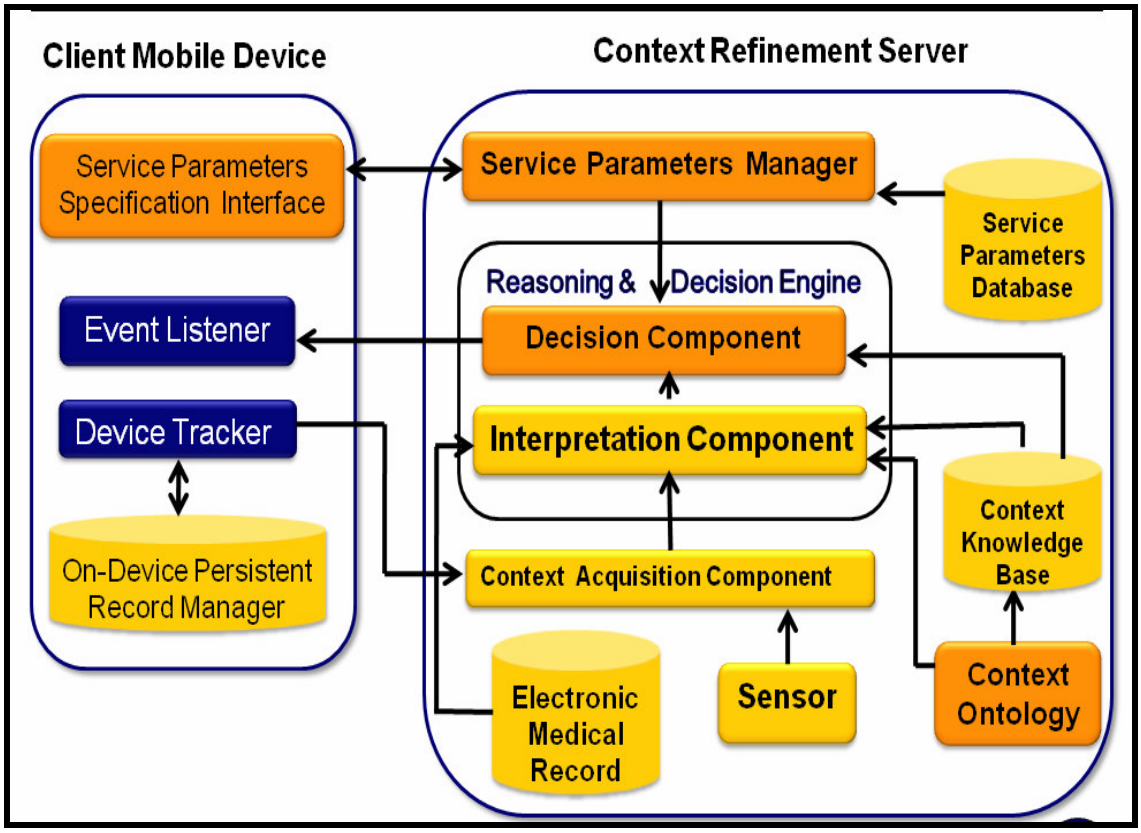

Figure 4.1: The Proposed Context Information Refinement Architecture

The core components in the above architecture and their corresponding purposes are described in Table 4.1.

Table 4.1: Major Components in the proposed Architecture

Component Name	Purpose
Service Parameters Specification Interface (SPSI)	Used to specify, modify or remove the list of service parameters by the user.
Event Listener (EL)	Continuously listens to incoming notifications from the decision component and alerts the user on arrival of notifications.
Device Tracker (DT)	Continuously monitors the device context (E.g. location, identity of user) and periodically submits the device context to the context acquisition component.
Service Parameters Manager (SPM)	Maintains list of up-to-date service parameters provided by the user through the service parameters specification interface. It also provides interface for manipulating service parameters.
Interpretation Component (IC)	Performs rule-based reasoning over context information it gets from the context acquisition component and the context knowledge base.
Decision Component (DC)	Passes the final decision of delivering or not of the context information to the user.
Context Acquisition Component (CAC)	Collects and aggregates context information from potential context sources like sensors and the Device Tracker.

The various components of the context information refinement architecture are distributed between the client mobile device and the context refinement server. The decision on where to place which component relies on the careful consideration of the already described constraints, with regards to storage, processing, input/output, communication and battery life, of the mobile devices. One way to achieve this is by considering the internal complexity of each component and hence placing those components that are lightweight (which consume less resource[9]) on the client mobile device while those components which require relatively more resources and/or accessed very frequently are placed on the server side, where constraints on resources are more relaxed.

[9] Resource can be one or more of storage, processing, communication, input/output and power consumption costs incurred by a particular component.

Accordingly, the Service Parameters Specification Interface, the Event Listener and the Device Tracker are placed on the client mobile device. On the other hand, the context refinement server consists of those components that are resource-intensive or those that can benefit from the performance of the whole architecture due to their proximity to the core component of the architecture. These components are the Service Parameters Manager, the Reasoning and Decision Engine and the Context Acquisition Component. Furthermore, the potential sources of information for these components including the Electronic Medical Record (EMR), the context knowledge base, the context ontology, the service parameters database and sensors are within the context refinement server.

4.2. Details of the Proposed Architecture

In this section, the logical architecture of the proposed context information refinement model is presented. In the architecture, the central (core) component that plays the most decisive role is the Reasoning and Decision Engine. This component is responsible for interpretation of context information that it gets from the Context Acquisition Component and the context knowledge base on the context refinement server. The other responsibility of this component is to determine (decide) whether delivering context information, that has been interpreted, to the target application (user) on the client mobile device is feasible or not.

In the proposed architecture, the service parameters are provided using the Service Parameters Specification Interface. The Service Parameters Manager maintains the list of up-to-date service parameters. The Reasoning and Decision Engine has two sub-components, i.e., the Interpretation Component that performs the actual reasoning and the Decision Component that interacts with the Service Parameters Manager, on the context refinement server, to request up-to-date service parameters and with the Event Listener, on the client mobile device, to deliver context information that is found useful with respect to the specifications provided. The Reasoning and Decision Engine also interacts with the EMR in order to retrieve medical history and the Context knowledge base in order to get context history that may be helpful in facilitating the interpretation, decision and

consequently delivery of context information to the Event Listener on the client mobile device.

4.2.1. The Service Parameters Specification Interface (SPSI)

As described in chapter one, the important issues that this thesis addresses are quality of context information, relevance of context information and the dynamic nature of the domain. These issues are addressed by providing a generic interface on the client mobile device that helps the user to exhaustively specify the potential determinants of these issues. In addition to specifying the service parameters, it also allows the user to modify or possibly remove existing service parameters if the need arises. This enables the Service Parameters Manager, on the context refinement server, to maintain up-to-date list of service parameters specified by the user and based on which the Decision Component commits its final decision.

We classify the major service parameters included in this interface into four major categories based on their similarity of specification. These are identification parameters, precision-based parameters, range-based parameters and special parameters. Table 4.2 shows the respective description and examples of these parameter types.

Table 4.2: Parameter Types in the Service Parameters Specification Interface

Parameter Type	Description	Example(s)
Identification Parameters	These are types of parameters that are used to specify unique identification of entities like patients and physicians.	-Doctor Identification -Nurse Identification -Patient Identification
Precision-based Parameters	These are types of parameters that are used to specify values against a certain acceptable constant value.	-Location Precision -Notification Delay
Range-based Parameters	These are types of parameters which are used to specify values defined between a certain minimum and maximum threshold.	-Temperature Range -Blood Pressure Range -Heartbeat Range -Insulin Level Range
Special Parameters	These are types of parameters that are used to specify special user parameters.	-The specific time when the user does not accept notifications like when s/he is off duty.

The parameter types listed in the above table have implications on the quality of context, relevance of context and the particularity of the domain. For example, the location precision and notification delay are good indicators of quality of information because they indicate the extent to which the location of a user is closer to the correct location and the notification delay is closer to an already specified acceptable delay. A more specific example can be, time at which the context information is produced is 4:30PM and the notification arrives five minutes later (at 4:35PM) where the acceptable delay is set to a

maximum of 3 minutes. This notification is useless, as it does not conform to one of the anticipated quality parameters, i.e., freshness.

The range-based parameters are linked with the relevance of context information. For instance, if a nurse has specified a range between 69.5 pulses/minute and 74 pulses/minute as a normal range for a child patient and the heartbeat reading shows 70 pulses/minute, notifying this value to the nurse is irrelevant as the value is within the specified normal range. However, if the reading was 75 pulses/minute rather than 70 pulses/minute, it would have been relevant, as it would trigger the nurse to take some measures. In general, the parameters mentioned here, if well specified and considered in the course of context information refinement, can contribute significantly in enhancing the quality and relevance of context information.

4.2.2. The Event Listener (EL)

The sole purpose of this component in the architecture is to continuously listen to incoming notifications from the decision component and forward the notification to the user. It possesses a continuous communication capability such that it binds itself to a particular communication channel (E.g. a port number) so as to immediately accept all incoming messages through the channel. As opposed to interval-based polling, it subscribes to list of events and is notified on occurrence of one of the events that it subscribed for. Such a subscription-based approach is selected for this component primarily due to the unpredictable nature of notification arrivals and the frequent change of the underlying context like sensor readings of a patient and current context (E.g. location) of a user.

4.2.3. The Device Tracker (DT)

The primary responsibility of this component is to periodically track and submit the current context information (location and identity) of the device to the Context Acquisition Component. In addition, it maintains all local information that is required to be stored on the device, by interacting with the on-device persistent record manager. The presence of on-device persistent record manager is of significant contribution to reduce the impact of intermittent wireless connections, one of the constraints in pervasive environments, by providing local caching service on the client mobile device.

Like the Event Listener, it also needs to bind itself to a particular channel to which it periodically delivers the current context of the device that in turn indicates the current context of the device owner (user) provided that the device-to-user (ownership) association is defined in advance. As opposed to the Event Listener that listens continuously, it operates by polling for context changes within a specified time interval. For instance, it may be configured to submit change to current context (E.g. location) of the device every five seconds after analyzing the interval of context change in a real-life setting.

4.2.4. The Service Parameters Manager (SPM)

This component is in charge of maintaining up-to-date list of service parameters, which can be immediately consumed by the Decision Component. It needs to communicate with:

- The Service Parameters Specification Interface on the client mobile device, to accept the list of parameters specified by the user and
- The Decision Component on the context refinement server, to provide the list of up-to-date service parameters that enables the Decision Component to commit the delivery of context information to the Event Listener on the client mobile device.

Moreover, the insertion of new, modification and removal of existing service parameter records that may be initiated on the Service Parameters Specification Interface are handled by this component for which it needs to interact with the Service Parameters Database (see Figure 4.1).

The list of service parameters, stated in section 4.2.1, is structured into manageable parameter attributes that can be represented in a relational database table that has the following common fields:

User Identification, Patient Identification, Location Precision, Notification Delay, Minimum Temperature, Maximum Temperature, Minimum Heartbeat, Maximum Heartbeat, Minimum Blood Pressure, Maximum Blood Pressure, Minimum Insulin Level, Maximum Insulin Level, Time for No Notification, Time of Recording, …etc. Here, the list of parameters may be very long as this is something to be determined during implementation.

4.2.5. The Reasoning and Decision Engine (RDE)

This is the central component of the whole architecture that communicates and cooperates with all the remaining components of the system. It is dedicated to perform ontology supported rule-based reasoning and pass final decisions on whether or not to deliver the currently ready context information to the target application on the client mobile device. In particular, the Interpretation Component does the rule-based reasoning while the final decision of whether or not to deliver the context information is carried out by the Decision Component. It communicates with:

- The Device Tracker on the client mobile device for periodically receiving the current context (location and identity) of the device and all sensor data, through the Context Acquisition Component.
- The EMR to retrieve the necessary medical history that may be helpful in the interpretation, through the Interpretation Component.
- The Context knowledge base for accessing context history and reasoning rules, through the Interpretation Component.
- The Service Parameters Manager to get access to the recent list of service parameters, through the Decision Component.
- The Event Listener that accepts the final notifications decided to be delivered to a target application on the client device, through the Decision Component.

4.2.5.1. The Interpretation Component (IC)

The functionality of this sub-component is described through the following major breakdowns:

- Periodically accepts current context (location and identity) and sensor information through the Context Acquisition Component on the context refinement server.
- Determines whether new context knowledge can be derived from the accepted context information. For example, if it receives context information that shows Dr. Kebede is located in room 205, based on this information it is possible to determine

whether new context knowledge can be inferred or not by analyzing the doctor-to-patient association from the context ontology or the EMR. Furthermore, if one of the sensors reports a context information in a form [Patient Identification, Blood Pressure, Heart Beat, Body Temperature, Sensing Time], the Interpretation Component can determine whether a new context knowledge can be derived or not by analyzing the service parameters-to-patients association in the service parameters database. Incase it is not possible to infer new context knowledge from the current context; it stores the information to the context knowledge base because this information may be helpful for historical reasoning in the future.

- Applies reasoning rules to derive new context knowledge that it hands over to the Decision Component. The reasoning rules (see examples in Table 4.3) can be ontology reasoning rules, which define constraints on the relationship of properties of concepts in the healthcare environment or user-defined reasoning rules, which associate specific actions to a combination of various conditions. While deriving new context knowledge, it may need to access the EMR to enrich the semantics of the derived context information that in turn enhances the usability of the context information for decision-making.

Table 4.3: Examples of Reasoning Rules

Example for Ontology Reasoning Rules	Example for User-defined Reasoning Rules
LocatedAt	**Rule1:**
(?P rdf:type owl:TransitiveProperty) ^ (?A ?P ?B)^ (?B ?P ?C) =>(?A ?P ?C)	If (getHeartBeat(Meron, now)>75) OR (getHeartBeat(Meron, now)<72)then ActivateHeartBeatNotification(Dr.Kebede,Meron);
E.g. (LocatedAt rdf:type owl:TransitiveProperty)^ (?Dr.Daniel ?LocatedAt? Room201) ^ (?Room201 ?LocatedAt ?Bldg5) => (?Dr.Daniel ?LocatedAt ?Bldg5)	**Rule2:**
SubClassOf	If (getTemperature(Meron, now)>38.5) OR (getTemperature(Meron, now)<36) then ActivateTempNotification(Dr.Kebede, Meron);
(?a rdfs:subClassOf ?b) ^ (?b rdfs:subClassOf ?c) ^ =>(?a rdfs:subClassOf ?c)	
E.g. (?InPatient ?SubclassOf ?Patient) ^ (?Patient ?SubClassOf ?Person> =>(? InPatient ?SubClassOf ?Person)	

- Finally, hands over the newly derived context information to the Decision Component.

4.2.5.2. The Decision Component (DC)

This component is responsible for accepting the context information that has been prepared by the Interpretation Component and goes through the following steps to reach to the final decision of whether or not to deliver the prepared context information to the target application on the client mobile device.

- Correlates the information it has received with the recent service parameters by consulting the Service Parameters Manager.
- Commits the final decision of whether or not to deliver the information to the client device.
- If delivery is decided, it immediately sends notification message to the specific application through the Event listener on the client mobile device.

Furthermore, it appends the notification information to the context knowledge base so that the context knowledge base is equipped with richer set of context overtime. Otherwise, it waits for the next handover of information from the Context Acquisition Component. Figure 4.2 shows how the Reasoning and Decision Component goes through different states in the course of refining context information it gets from the Context Acquisition Component.

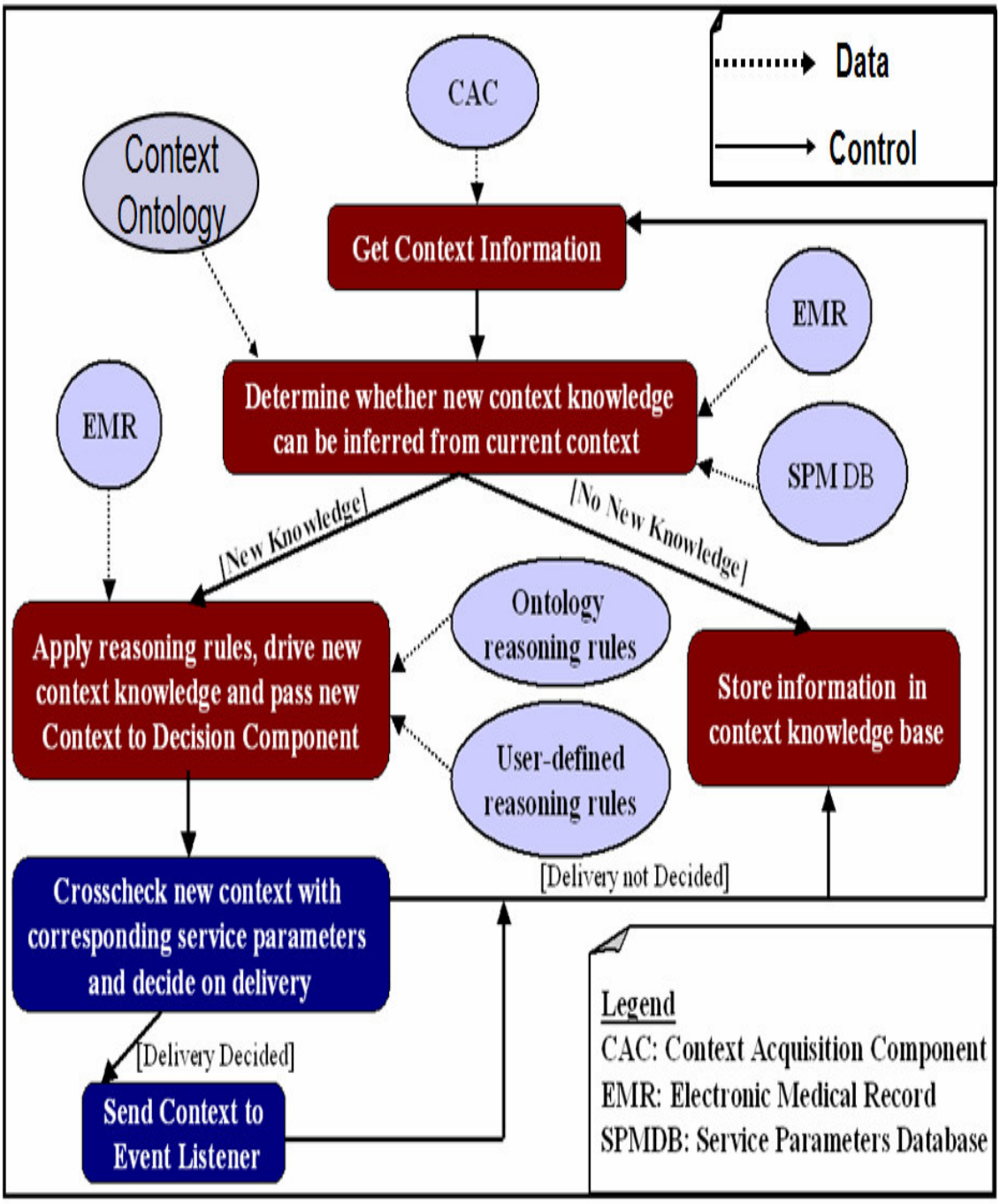

Figure 4.2: Internal Detail of the Reasoning and Decision Engine

4.2.6. The Context Acquisition Component (CAC)

This component, as its name indicates, is responsible for performing raw data acquisition and possible aggregation[10] of data it collects from the various potential sources of context and eventually delivers to the Interpretation Component. It is important to have a dedicated data acquisition component in the proposed architecture because of the following reasons:

- There should be a clear separation of role between the data collection and data interpretation component that reduces the dependency between these two components. Such separation of role simplifies the task of changing or enhancing one of the components without significantly affecting the other. As a result, the whole architecture will be open for modification and/or scalability issues.
- It creates insulation between the complexity and diversity of low-level context sensing and high-level context interpretation. This is an advantage for the Interpretation Component since it uses the ready-made context information and is relieved of the internal details of sensing.

The two potential sources of data for this component are the Device Tracker on the client mobile device and the sensor. From these data sources, it not only collects context data but also prepares the collected data in such a way that it will be easy for consumption by the Interpretation Component.

[10] Aggregation includes reducing heterogeneity in terms of representation formats and units of measurement for context data received from sources.

4.3. Summary of the Proposed Architecture

This chapter presented the detailed formulation of the proposed context information refinement architecture for pervasive medical systems. The architecture is composed of two major ends-the client mobile device and the context refinement server. On the client mobile device side, there are three components. These are: the Service Parameters Specification Interface used for specifying and possibly modifying and removing service constraints imposed by users, the Event Listener that continuously listens to incoming notification messages from the context refinement server, and the Device Tracker that periodically submits the current context of the device to the context refinement server.

The context refinement server side is composed of the Service Parameters Manager that is a counterpart for the SPSI, the Reasoning and Decision Engine responsible for interpretation of context information that it receives from the Context Acquisition Component and passing the final decision of delivering or not of the context information to the Event Listener on the opposite end. The sensor that periodically reports vital sign readings, the EMR that maintains persistent medical records and the Context Ontology that models the underlying knowledge in the domain are also part of the context refinement server.

5. Prototype Implementation and Demonstration

The previous chapter presented the detail of the proposed context information refinement architecture. This chapter discusses the prototype implementation of the architecture by stating the objectives of the prototype, tools and technologies used, the prototypical setup, the implementation details and evaluation results.

5.1. Overview

The prototype implementation developed in this thesis is principally aimed at providing proof of the concepts introduced in the detailed explanation of the proposed context information refinement architecture, discussed in the previous chapter. To achieve this principal goal, the majority of the components in the architecture have been implemented as per the theoretical specifications formulated for each component.

The prototype implementation lets users to receive notification messages from the context refinement server side on their mobile devices. The notification messages can either be location-based or urgent notifications like abnormal readings of vital signs for patients. In order to receive these kinds of notifications, the user is provided with a service parameters specification interface (on the mobile device) on which s/he specifies the service constraints for each of his/her patients. Furthermore, users can also modify or remove service constraints via the same interface. After specifying service constraints, the user performs his/her routine tasks.

The context refinement server continuously analyses the current context of the user (through the location of the mobile device the user carries) and the sensor readings. The user is notified either if s/he is currently located at (near) his/her patient's bed-room or the vital signs of one or more of his patients is out of the normal range specified using the interface. All notifications received on the mobile device from the context refinement server are based on the parameters that decide the quality and relevance of context information specified by the user on the interface provided. The following real-life pervasive healthcare scenario demonstrates how the prototype implementation operates.

Scenario:

- In a certain hospital, doctor Kebede is in charge of following up three patients P1, P2 and P3 sleeping in rooms 101,201 and 103 respectively.
- In a morning of a certain day, he looks at the to-do list on his PDA and realizes that his first duty is to make a ward-round of his patients.
- He walks to room 101 and eventually approaches the gate of the room. Right after he gets in, his PDA notifies him with the history of patient P1. He uses the notification information to decide the type of measure to take concerning patient P1 and leaves the room.
- Then after, he walks to room 201 to visit patient P2. As soon as he reaches the doorstep of the room, his PDA alerts him again. This time, the patient history displayed on the screen is that of patient P2. He observed the status of the patient and left the room to get back to his office.
- While he is walking to his office, his PDA alerts him once again. He was wondering if he was around a room of the patients he is in charge of. However, when he examines what is displayed on the screen of the PDA, the notification indicates that blood pressure and temperature of patient P3 in room 103 is out of the normal range that he specified an hour ago. Hence, he quickly goes to room 103 so as to give urgent treatment to patient P3.

5.2. Tools and Technologies Utilized for Implementation

Several tools and technologies were utilized for the purpose of developing the prototype implementation. The following is list of the programming, communication, database management, context representation, context reasoning and operating environment used in the prototype implementation.

- Java 2 Micro Edition (J2ME) version 2.5.1 for CLDC (Connected Limited Device Configuration) of the MIDP (Mobile Information Device Profile) is used for developing the mobile device edition of the prototype [27], [28].

- Java 2 Standard Edition (J2SE) version 1.5.0_06 is used for developing the desktop machine edition of the prototype.
- MY SQL database server version 5.0.22 is used for persistent data management on the context refinement server.
- Java Servlet API [29] version 2.1 is used for bridging the mobile device edition and the desktop edition of the prototype application.
- Apache Tomcat version 4.1.12 is used as a web container for Java Servlets.
- Protégé Ontology Editor[11] Version 3.2 is used for developing pervasive healthcare ontology.
- Jena [30] Semantic Web framework version 2.5.2 is used for generating an RDF model and implementing the ontology and user-defined reasoning rules.
- Microsoft Windows XP operating system is used for the context refinement server.
- Palm Tungsten 5 PDA with Palm operating system version 3.5 and IEEE[12] 802.11 wireless LAN capability is used as a mobile device.
- IEEE 802.11 wireless LAN (Wi-Fi) is used as a communication infrastructure between the mobile device and the context refinement server.

5.2. Prototypical Setup

To support scenarios of the previous type (section 5.1), the prototype system is implemented as an application with the client-side on the mobile device (Palm PDA) and the server-side on context refinement server (Windows PC). The two sides communicate over a Wi-Fi LAN. On the client mobile device, the Service Parameter Specification Interface is implemented as a J2ME MIDlet while the Event Listener and the Device Tracker are implemented as J2ME threads that bind themselves to a certain port to communicate with the services on the server through a standard Hyper Text Transfer Protocol (HTTP).

On the context refinement server side, the Service Parameters Manager, the Reasoning and Decision Engine, the Context Acquisition Component and the Sensor are implemented as Java Servlets. The context knowledgebase and the context ontology are implemented as

[11] http:// www.protégé.stanford.edu
[12] Institute of Electrical and Electronics Engineers

RDF schema and OWL ontology document respectively. The EMR and the Service Parameters Database are implemented as MySQL relational databases. The detail of implementation for the major components is outlined in the following section.

5.3. Implementation Details

5.3.1. Modules on the Client Mobile Device

The Service Parameters Specification Module: It is implemented as a J2ME MIDlet class. It incorporates all the major service parameters including the identification, precision-based, range-based and special parameter types discussed in the previous chapter. Moreover, the specification of new, removal and update of existing service parameters are implemented as functionalities of this interface.

The Event Listener Module: It is a J2ME thread class that continuously listens to incoming messages from the decision module, which is part of the reasoning and decision engine. Before it starts continuous listening, it establishes an HTTP connection to the Decision component on the context refinement server and binds itself to a particular port. Up on arrival of a notification message, it automatically alerts the user by displaying the message on the screen of the PDA. The source code for this module is included in Appendix A.

The Device Tracker Module: This is a J2ME thread that periodically tracks the current context (mainly location, identity of user and time) and submits the current context to the context acquisition component on the context refinement server. Like the Event Listener, it opens an HTTP connection to the context refinement server and binds to a port number. Unlike the Event Listener, it periodically submits the current context of the device every five seconds. The source code for this module is included in Appendix A.

5.3.2. Modules on the Context Refinement Server

The Service Parameters Manager Module: This module handles the insertion, removal, update (from the mobile device through SPSI), and requests for recent service parameters from the decision component. It is a Java Servlet class that communicates with the J2ME MIDlet of the SPSI on the client mobile device and the service parameters database on the context refinement server.

The Reasoning and Decision Module (RD-module): This is the most complex module in the whole prototype as it is the core component in the proposed context information refinement architecture. The whole module is a Java Servlet class (in a form of continuously running server thread). It has two sub-modules: the Interpretation Component (IC) module and the Decision Component (DC) module, which are also Java Servlet classes.

The IC-module first receives the current context of the mobile device and the sensor readings from the Context Acquisition Component and invokes the pervasive healthcare ontology, the EMR and the context knowledgebase for undertaking a reasoning task that is performed by generating an RDF model from the ontology. The reasoning is done by querying the generated model using RDQL (RDF data Query Language). Based on the responses returned by the query, the IC-module enriches the context information with additional information retrieved from the EMR. At the same time, it also updates the context Knowledge base. Finally, it hands over the context information to the DC-module. The DC-module, as soon as it accepts the context information from the IC-module, it crosschecks the information with the recent service parameters set by the user. If the service parameters are in agreement with the current context information, it opens an HTTP connection to the Event Listener on the PDA and sends the context information as a notification through the Wi-Fi LAN. The Servlet source code is included in Appendix B.

The Context Acquisition Module (CA-module): It is responsible to periodically collect, aggregate (if need be) and submit context information that triggers the IC-module of the RD-module. It is implemented as a Java Servlet class in a form of continuously running server thread. Its main duty in the prototype is to continuously listen to the sensor and the Device Tracker to collect raw context data that is submits to the IC-module.

The EMR and Service Parameters Database: They are relational databases for the electronic medical record and service parameters repository, with interfaces to access the database. They are implemented as My SQL server relational databases while the interfaces to access them have been implemented as Java Servlet classes.

The Sensor Module: This module is used to simulate the operation of real sensors in such a way that it periodically generates a valid set of sensor values for a particular patient, attaches timestamp and submits the values to the CA-module. It has been implemented as a Java Servlet class with a thread characteristic as it is expected to run continuously.

The Context Knowledgebase (CKB): This is where the context knowledge that is accumulated overtime will be stored. The IC of the RDE mainly accesses it. It is implemented as an RDF schema document.

The Context Ontology: This is where all the terms (concepts), properties of concepts, relationship between concepts and constraints on the properties of concepts in the pervasive healthcare environment are modeled. It is implemented as a protégé OWL ontology document. Figure 5.1 shows the class hierarchy of the concepts modeled by the ontology. The complete OWL code for the ontology is included in Appendix C.

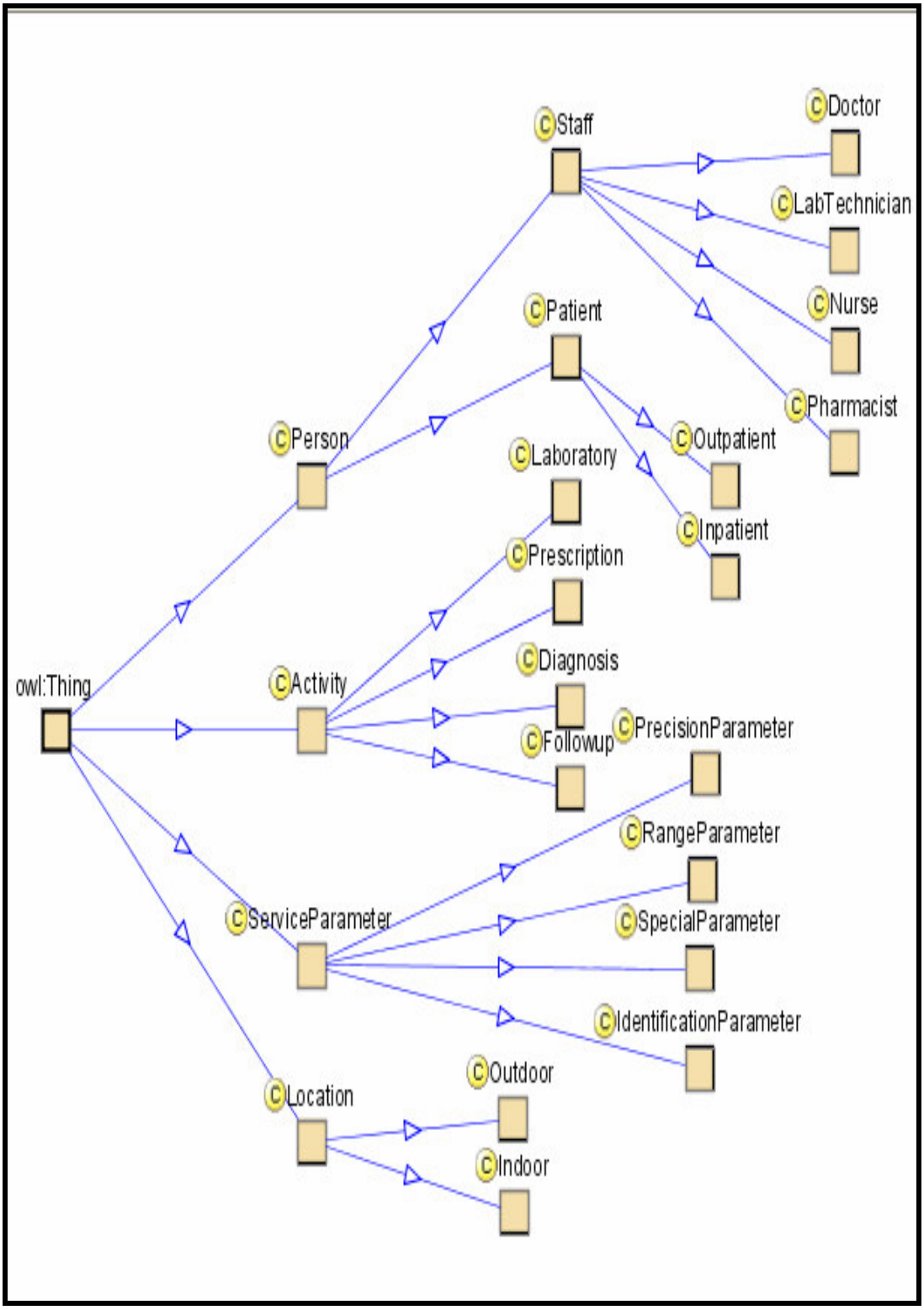

Figure 5.1: Class hierarchy for the developed pervasive healthcare ontology

5.4. Demonstration

For evaluating the implemented features, we created records of physicians and patients in the EMR and the equivalent instances of the ontology are generated. The patient-room-physician associations are also maintained. The SPSI on the mobile device is used to specify constraints on the services provided by the context refinement server including the patient and staff identification, a location accuracy constraint (precision-based parameter), acceptable range for heartbeat, temperature and blood pressure (range-based parameters) and the time at which the staff does not accept notifications (special parameter). Figure 5.2 shows a snapshot of the SPSI on the mobile device showing user entering service parameters for a patient. Due to the limitation on the display size of the mobile device, all the parameters are not displayed in the following snapshot.

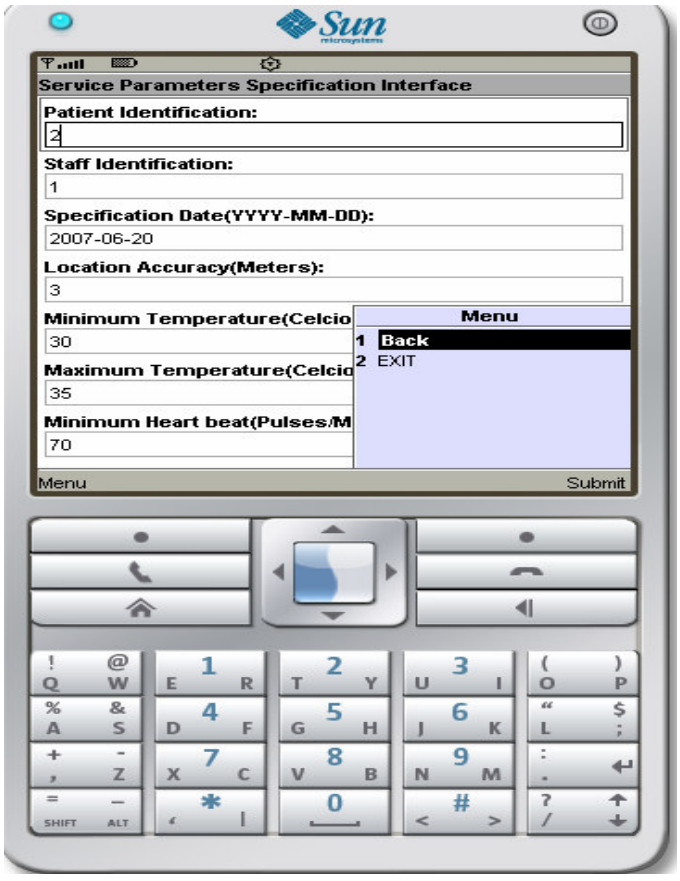

Figure 5.2: Mobile device interface for service parameters specification

After specifying the parameters, the context refinement server notifies the user either when he/she is at (near depending on the distance accuracy) the room of his/her patients or the sensor report indicates one or more of the vital signs to be out of the specified ranges. Figure 5.3 shows automatic notifications from the context refinement server indicating that the user is in room 101(left) at 3:45PM and in room 201 (right) at 3:55PM, i.e., after 10 minutes. The notification displays the history of the patient sleeping in that particular room. Such a notification assists the physician to make informed decisions.

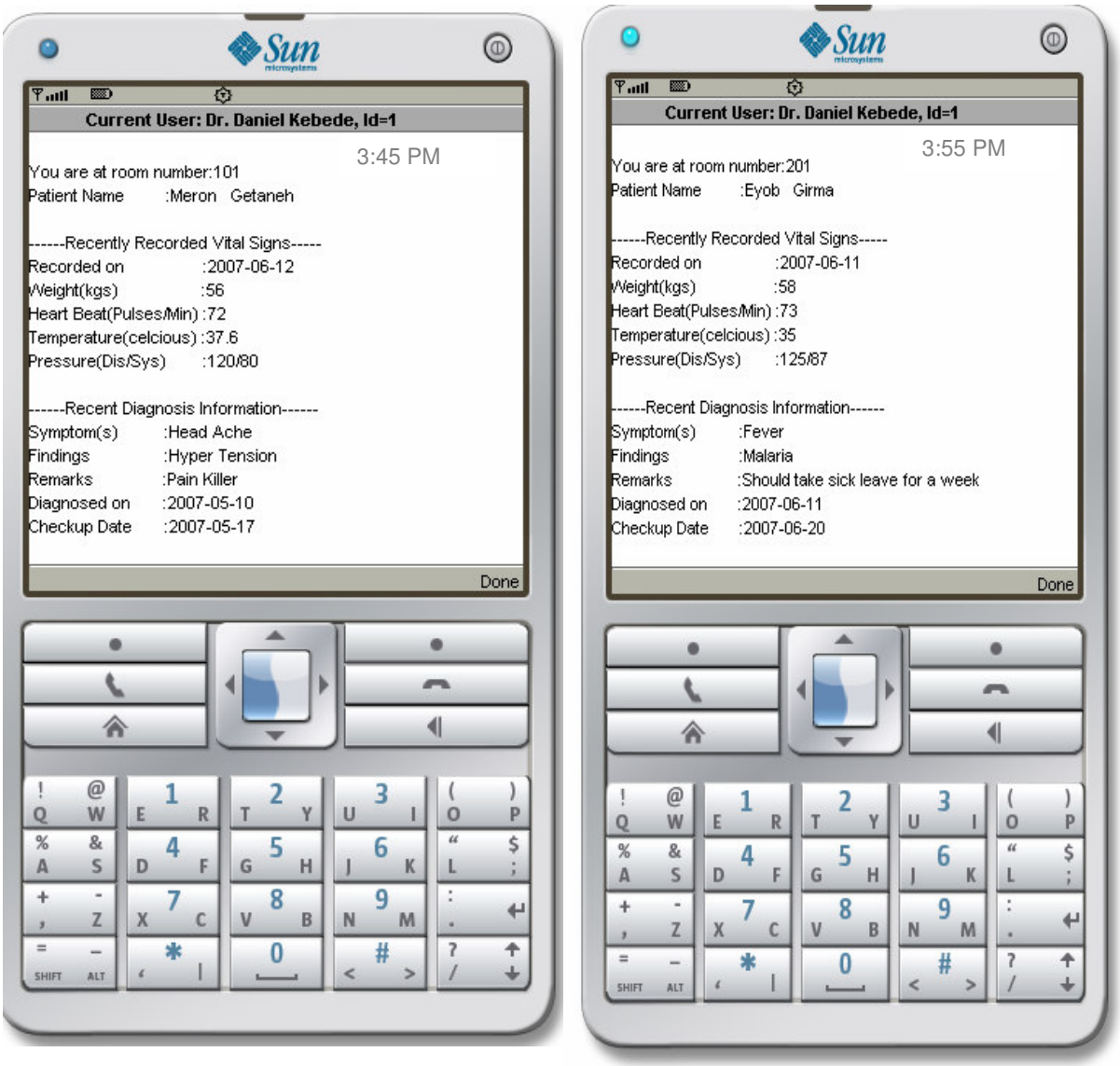

Figure 5.3: Automatic notification on the mobile device pertinent to user's location

The other kind of notification provided by the context refinement server is when the sensor report indicates abnormal vital signs of a patient. In such a case, the user is notified irrespective of his/her current location so as to take urgent measures. Figure 5.4 shows service parameter specification (left) for patient named "Yared Alemayehu" and the notification message (right) that alerts the user when the temperature and heartbeat of this patient are out of the ranges specified by the user.

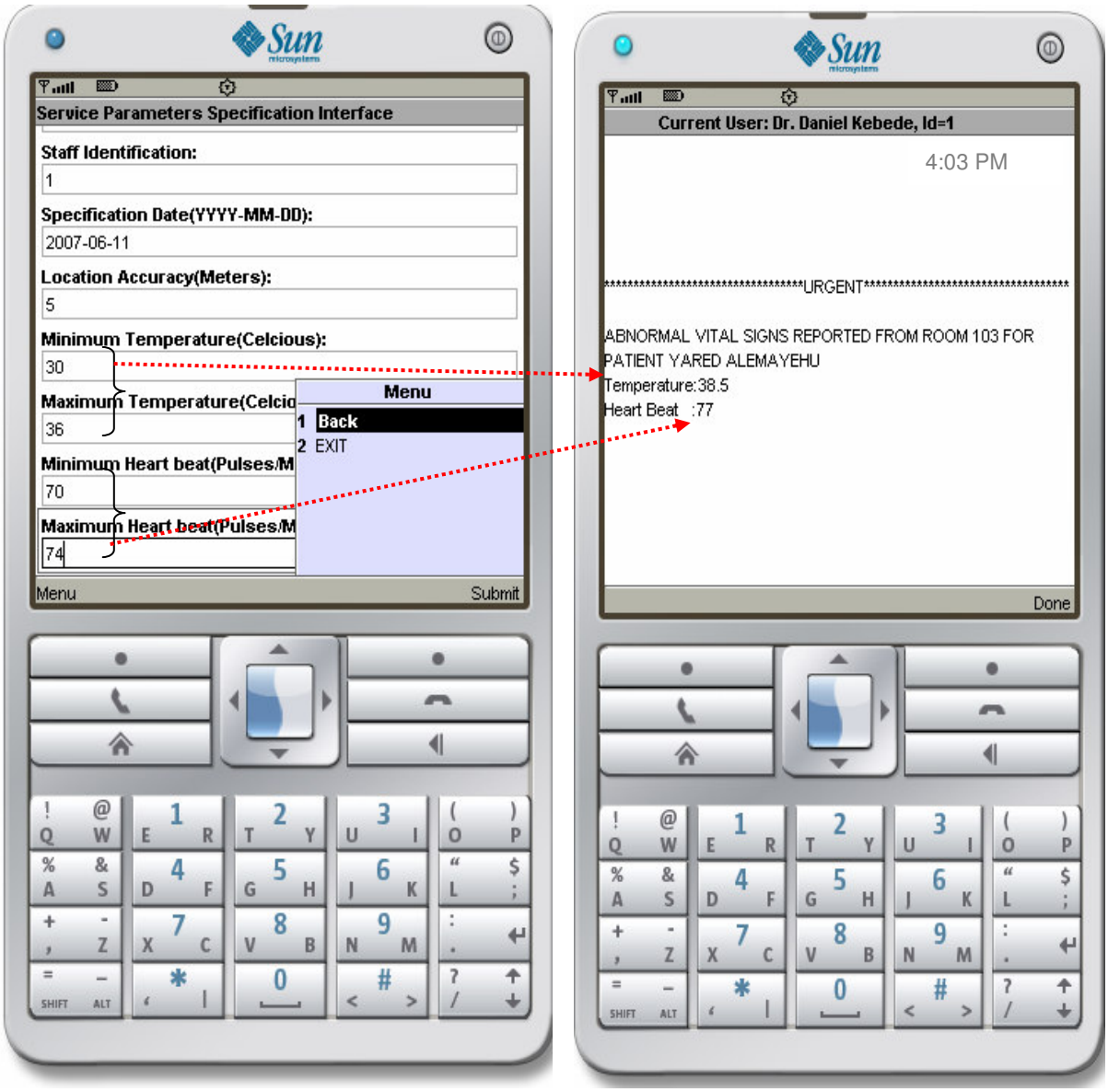

Figure 5.4: Automatic notification of abnormal vital signs reported from sensors

6. Conclusions and Future Work

In this thesis, we first identified the shortcomings that the existing context management models for pervasive medical systems experience with regards to context information refinement. These shortcomings were: quality of context information, relevance of context information and the incorporation of domain specific requirements for pervasive medical systems.

To address these shortcomings, we proposed (as explained in chapter 4) a context information refinement architecture that aims at facilitating and coordinating refinement of context information in a pervasive medical system. In the proposed architecture, we have made a research contribution to address the three issues as follows:

- We introduced a separate Service Parameters Specification Interface on the client mobile device that allows the user to explicitly put constraints on the quality parameters of context information. The counter part of this interface is the Service Parameters Manager on the context refinement server aimed at maintaining up-to-date service parameters that are used to decide on the delivery of context information to the user.

- The relevance of context information is ensured by the Decision Component on the context refinement server that crosschecks the context information that has been prepared by the Interpretation Component with the specific values of each service parameter indicating the preference of the user.

- The dynamic nature of pervasive medical systems is addressed by representing the domain concepts, properties and relationships between concepts and properties using a pervasive healthcare ontology that facilitates reasoning of context in such a domain where the context data of patients (E.g. vital signs reported from sensors) and medical professionals (E.g. current location) change frequently.

In order to demonstrate the validity of the proposed context information refinement architecture, we selected a pervasive healthcare scenario for which we developed a prototype implementation (explained in chapter 5). In the prototype, we have implemented the major components of the proposed architecture and conducted an encouraging initial evaluation of how the architecture facilitates refinement of context information by considering the quality and relevance parameters specified by medical professionals and the particular characteristics of the pervasive medical domain using the underlying pervasive healthcare ontology.

Although we tried our best to realize context information refinement architecture for pervasive medical systems under the objective of addressing the shortcomings of existing works, we do not believe that the architecture is generic enough to incorporate potential issues in pervasive medical systems. For instance, despite the importance of the issue, we have not considered the privacy and security aspect of context information in the course of context refinement since it was beyond the scope of this work. Therefore, we believe that the proposed architecture can be enhanced in such a way that the privacy and security of context is taken into account while refining context information.

The other line of improvement is regarding the complete implementation of the prototype. Even though we implemented many of the components in the proposed architecture, components like the Device Tracker and the Sensor were simulated based on the study we made on the operation of real location detection devices and sensors respectively. We preferred to simulate these components due to of lack of real devices. Hence, by taking this prototype as a platform, a more realistic implementation of the proposed architecture is another slot of improvement so as to maximize its usability in a real-life setting in the future.

References

[1]. Jakob E. Bardram and Henrik Bærbak Christensen, "Supporting Pervasive Collaboration in Healthcare — An Activity-Driven Computing Infrastructure", Center for Pervasive Computing, Department of Computer Science, University of Aarhus Aabogade 34, 8200 Aarhus N, Denmark, In Proceedings of the 2004 ACM Symposium on Applied Computing, 2004.

[2]. J. P. Black, W. Segmuller, N. Cohen, B. Leiba, A. Misra, M. R. Ebling, and E. Stern, " Pervasive Computing in Health Care: Smart Spaces and Enterprise Information Systems", IBM T. J. Watson Research Center, Hawthorne, NY 10532, In Proceedings of the Second International Conference on Mobile Systems, Applications and Services (MobiSYS2004), June, 2004.

[3]. John A. Flanagan,Johan Himberg1, Jani M¨antyj¨arvi, " A Hierarchical Approach to Learning Context and Facilitating User Interaction in Mobile Devices", VTT Technical Research Center of Finland, In Artificial Intelligence in Mobile System 2003 in conjunction with Ubicomp 2003, Seattle, USA, October, 2003.

[4]. Yngve Dahl, "Context-Aware Technology and Challenges of Hospital Wards", Department of Computer and Information Science, The Norwegian University of Science and Technology, Trondheim, Position Paper, 2006.

[5]. Jens H. Jahnke, Yury Bychkov, David Dahlem, Luay Kawasme, " Implicit, Context-Aware Computing for Health Care", University of Computer Science. 19th Annual ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA), 2004.

[6]. Thomas Buchholz, Alex Kupper, Micheal Schiffers, “Quality of Context-What It Is And Why We Need It?”, MNM-Team, Department of Informatics, Ludwig-Maximilians-University, Munich, Germany, In Proceedings of the workshop of the HP Open view University Association 2003 (HPOVUA 2003), Geneva, 2003.

[7]. Karen Henricksen, Jadwiga Indulska, and Andry Rakotonirainy, “Modeling Context Information in Pervasive Computing Systems”, School of Information Technology and Electrical Engineering, The University of Queens land, St Lucia QLD 4072, Australia, In Proceedings of the first International Conference on Pervasive Computing, Pervasive 2002, Zurich, Switzerland, August, 2002.

[8]. Matthias Baldauf, Schahram Dustdar, Florian Rosenberg,” A Survey on Context-Aware Systems”, Distributed Systems Group, Information Systems Institute, Vienna University of Technology, Argentinierstrasse 8/184-11040 Vienna, Austria, International Journal of Ad Hoc and Ubiquitous Computing, Inderscience Publishers, January 2006.

[9]. Guanling Chen, David Kortz, “A Survey of Context-Aware Mobile Computing Research”, Department of Computer Science,Dartmouth College, Dartmouth Computer Science, Technical Report TR2000-381, In ACM Publications, 2000.

[10]. Heinz-Gerd Hegering, Axel K¨upper,Claudia Linnhoff-opien, Helmut Reiser, “Management Challenges of Context-Aware Services in Ubiquitous Environments”, Munich Network Management Team, University of Munich, Department of Informatics, Oettingenstr.67,D-80538 Munich, Germany, In Proceedings of the 14th IFIP/IEEE Workshop on Distributed Systems, Operations and Management (DSOM), 2003.

[11]. DEY, A. K., "Understanding and Using Context", Personal and Ubiquitous Computing archive, Special issue on Situated Interaction and Ubiquitous Computing, ACM Publications, February, 2001.

[12]. AK Dey, GD Abowd, D Salber, "A Conceptual Framework and a Tool for the Rapid Prototyping of Context-Aware Applications", Human-Computer Interaction (HCI) Journal, 2001.

[13]. M.A. Razzaque, Simon Dobson and Paddy Nixon, "Categorization and Modeling of Quality in Context Information", In Proceedings of the IJCAI 2005 Workshop on AI and Autonomic Communications, 2005.

[14]. Mark Weiser, "The Computer for the 21st Century", Scientific American Journal, September 1991.

[15]. Uwe Hansmann, Lother Merk, Martin S, Nicklous, Thomas Stober, "Pervasive Computing-The Mobile World", Second Edition, Springer, 2003.

[16]. Petteri Nurmi, Patrik Floréen, "Reasoning in Context-Aware Systems", Helsinki Institute for Information Technology (HIIT), Basic Research Unit (BRU), Position Paper, 2004.

[17]. Surafel Lemma, "Semantic Description of Multimedia Content Adaptation Web Services", Master's Thesis, Addis Ababa University, Addis Ababa, Ethiopia, July 2005.

[18]. Jakob E. Bardram, " Applications of Context-Aware Computing in Hospital Work– Examples and Design Principles", Center for Pervasive Healthcare Department of Computer Science, University of Aarhus Aabogade 34, 8200 Aarhus N, Denmark. In Proceedings of the 2004 ACM Symposium on Applied Computing, 2004.

[19]. Jens H. Jahnke, “Towards Context-Aware Computing in Clinical Care”, Department of Computer Science, University of Victoria, Victoria, B.C. Canada, OOPSLA Workshop on Building Software for Pervasive Computing, San Diego, CA, USA, 2005.

[20]. Jorgensen Jens Bæk, Bossen Claus, “Executable Use Cases for Pervasive Healthcare”, In Proceedings of the Second International Workshop on Modeling of Objects, Components, and Agents (MOCA'02), pages 89-104, Aarhus, Denmark, August 26-27, 2002.

[21]. Heinz-Gerd Hegering, Axel K¨upper,Claudia Linnhoff-opien,Helmut Reiser, “Management Challenges of Context-Aware Services in Ubiquitous Environments”, Munich Network Management Team, University of Munich, Department of Informatics, Oettingenstr.67,D-80538 Munich, Germany. In Proceedings of the 14th IFIP/IEEE Workshop on Distributed Systems, Operations and Management (DSOM), 2003.

[22]. Cormac Driver, Éamonn Linehan, Mike Spence, Shiu Lun Tsang, Laura Chan and Siobhán Clarke, “Facilitating Dynamic Schedules for Healthcare Professionals”. In proceedings of Pervasive Health Conference and Workshops, 2006.

[23]. Sheetal Agarwal, “Context-Aware System to Create Electronic Medical Encounter Records”, Master’s Thesis, Computer Science and Electrical Engineering Department of University Of Maryland, Baltimore County, May 2006.

[24]. X. H. Wang, D. Q. Zhang, T. Gu and H. K. Pung, "Ontology based context modeling and reasoning using OWL", Workshop on Context Modeling and Reasoning at 2nd IEEE International Conference on Pervasive Computing and Communication (PerCom'04), 2004.

[25]. Harry Lik Chen, “An Intelligent Broker Architecture for Pervasive Context-Aware Systems”, PhD Dissertation, University of Maryland, USA, 2004.

[26]. Thomas Strang, Claudia Linnhoff-Popien," A Context Modeling Survey", Workshop on Advanced Context Modeling, Reasoning and Management as part of UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing, Nottingham/England, September, 2004.

[27]. Dreatmtech Software Team, "Wireless Programming in J2ME: Cracking the Code™ ", Published by Hungry Minds Inc., USA, 2002.

[28]. Vartan Piroumian, "Wireless J2ME™ Platform Programming", The Sun Microsystems Press-Java™ Series, Prentice Hall Publishers, 2002.

[29]. H. M. Deitel, P. J. Deitel, "Java™ How to Program, Sixth Edition", Prentice Hall Publishsers, 2004.

[30]. Philip McCarthy, "Introduction to Jena", http://www.ibm.com/developerworks/xml/library/j-jena, Accessed on February 15, 2007

[31]. M. Satyanarayanan, "Pervasive Computing: Vision and Challenges", IEEE Personal Communications, August 2001.

[32]. Roy Want, Andy Hopper, Veronica Falcao, Jon Gibbons. "The Active Badge Location System", Technical Report 92.1, Olivetti Research Ltd., ORL, 24a Trumpington Street, Cambridge CB2 1QA, 1992.

Appendices

A: J2ME MIDlet for Event Listener and Device Tracker on Mobile Device

```
class DeviceTrackerThread extends TimerTask implements Runnable {
        String url = "http://localhost:8080/getdevicecontext/getcontext";
        String currentLocation;
          String StaffId="1";
        String[] locations=new String[6];
       //Constructor
        public DeviceTrackerThread() {
              }
        public void start() {
            Thread t = new Thread(this);
            t.start();
        }
        public void run() {
          locations[0]="101";locations[1]="102";locations[2]="103";
          locations[3]="201";locations[4]="202";locations[5]="203";
          //pick system clock in milliseconds
          Random r=new Random();
          int index=Math.abs(r.nextInt())%6;
          currentLocation=locations[index];
          try {
          //Open an HTTP connection
         HttpConnection c = (HttpConnection) Connector.open(url);
         c.setRequestProperty("User-Agent","Profile/MIDP-1.0, Configuration/CLDC-1.0");
         c.setRequestProperty("Content-Language", "en-US");
         c.setRequestMethod(HttpConnection.POST);
         // Open data output stream and send(write) data
         DataOutputStream os =(DataOutputStream) c.openDataOutputStream();
         os.writeUTF(StaffId.trim());
         os.writeUTF(currentLocation.trim());
         os.flush();
         os.close();
         } catch (Exception e) {
               showLocationAlert(e.getMessage());
           }
        }
        /* This method passes the current location to the servlet */
        public void setContext() {
             this.StaffId=StaffId;
             this.currentLocation=currentLocation;
        }
                   /* Display alert On screen*/
                  private void showLocationAlert(String err) {
                       Alert a = new Alert("");
                      a.setString(err);
             a.setTimeout(Alert.FOREVER);
```

```
                //display.setCurrent(a);
        }
       }; //class DeviceTrackerThread
//-----------------------------------------------------------------------------------------------
class EventListenerThread extends TimerTask implements Runnable {
       String url = "http://localhost:8080/rde/reason";
         public MainMIDlet listenerMidlet;
       String patientId;
       private Display display;
       //Constructor
         public EventListenerThread(MainMIDlet listenerMidlet) {
        this.listenerMidlet = listenerMidlet;
         display = Display.getDisplay(listenerMidlet);
       }
       public void start() {
          Thread t = new Thread(this);
          t.start();
       }
       public void run() {
                 Random r=new Random();
                  int randomId=Math.abs(r.nextInt())%10;
                  int id=randomId%3;
              if(id==0)
                patientId="1";
              else if (id==1)
                patientId="2";
              else
                patientId="3";
                  StringBuffer sb = new StringBuffer();
          try {
       //Open an HTTP connection
       HttpConnection c = (HttpConnection) Connector.open(url);
       c.setRequestProperty("User-Agent","Profile/MIDP-1.0, Configuration/CLDC-1.0");
       c.setRequestProperty("Content-Language", "en-US");
       c.setRequestMethod(HttpConnection.POST);
       // Open data output stream and send
       DataOutputStream os =(DataOutputStream) c.openDataOutputStream();
```

```
 os.writeUTF(patientId.trim());
 os.flush();
 os.close();
 // Get response from servlet
DataInputStream is =(DataInputStream)c.openDataInputStream();
int ch;
sb = new StringBuffer();
while ((ch = is.read()) != -1) {
sb.append((char)ch);
}
 // here check whether the string buffer is empty or not. if empty display nothing
        String test="You are at room number:";
            if (sb.length()>23)
        {
         showEventAlert(sb.toString());
        }
        else
         ;//noop
        is.close();
        c.close();
    } catch (Exception e) {
        showErrorAlert(e.getMessage());
    }
 }
 /* This method passes patient ID to servlet */
 public void setPatientId() {
    this.patientId = patientId;
 }
/* Display error alert On screen*/
 private void showErrorAlert(String err) {
    Alert a = new Alert("");
    a.setString(err);
    a.setTimeout(Alert.FOREVER);
    display.setCurrent(a);
 }
   /* Display event alert On screen*/
```

```
    private void showEventAlert(String alert) {
        Alert a = new Alert("          Current User: Dr. Daniel Kebede, Id=1");
        if (alert.length()==23)
         {
        ;//noop
       }
         else
         {
         a.setString(alert);
         a.setTimeout(Alert.FOREVER);
         display.setCurrent(a);
       }
    }
     };//class EventListenerThread
```

B: Java Servlet Class for the Reasoning and Decision Engine

```
public class reasonDecide extends HttpServlet {
   public void init() {
    }
    public void doPost(HttpServletRequest request,HttpServletResponse response)
throws ServletException,IOException {
DataInputStream in =new DataInputStream((InputStream)request.getInputStream());
String patientId = in.readUTF();
// String message ="You are at room number:";
String
message="**********************************URGENT**********************
****************\n\n";
      try {
 message=message+connect(patientId.trim());//concatentate with the return value

      } catch (Throwable t) {
         message += t.toString();
      }
      response.setContentType("text/plain");
        response.setContentLength(message.length());
      PrintWriter out = response.getWriter();
      out.println(message);
      in.close();
      out.close();
      out.flush();
   } //doPost

   public void doGet(HttpServletRequest request,HttpServletResponse response)
throws ServletException,IOException {
      doPost(request,response);
   } //doGet
```

```
    // This method connects to MYSQL database
    private String connect(String patientId) throws Exception {
          String message="";
          int pId=Integer.parseInt(patientId);
          int sId=1;
          String currentLocation="";
          String roomNo="";
         // Establish a JDBC connection to the MYSQL database server.
         Class.forName("com.mysql.jdbc.Driver").newInstance();
         Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/emrdb","root","passme");
         Statement st= conn.createStatement();
         String sql1 = "";
         sql1 = "SELECT * FROM tblassociation WHERE PatientId="+pId;
         ResultSet rsAssociation = st.executeQuery(sql1);
           while(rsAssociation.next())
           {
           sId=rsAssociation.getInt("StaffId");
             roomNo=rsAssociation.getString("RoomNo");
           }
             String sql2="";
           sql2 = "SELECT CurrentLocation FROM tblcurrentlocation WHERE
           StaffId="+sId;
             ResultSet rsLocation = st.executeQuery(sql2);
           while (rsLocation.next())
           {
             currentLocation=rsLocation.getString("CurrentLocation");
           }
             if (currentLocation.equals(roomNo) )
               {
                  //crosscheck with the ontology about identity with location

                  String sql4="";
                  sql4 = "SELECT FirstName,LastName FROM tblpatient WHERE
                 PatientId="+pId;
                  ResultSet rsName = st.executeQuery(sql4);
                  String fullName="";
                  while (rsName.next())
               {
                  fullName="Patient Name        :"+rsName.getString("FirstName")+"
                  "+rsName.getString("LastName");
                }
                 rsName.close();
                  String sql3="";
                  sql3 = "SELECT * FROM tblfollowup WHERE PatientId="+pId;
                  ResultSet rsFollowup = st.executeQuery(sql3);
                  // send the notification
                  while (rsFollowup.next())
               {
                  message=message+currentLocation+"\n"+fullName+"\n\n"+"------
Recently Recorded Vital Signs-----\n"+"Recorded on
```

```
:"+rsFollowup.getString("FollowupDate")+"\n"+"Weight(kgs)
:"+rsFollowup.getString( "weight")+"\n"+"Heart Beat(Pulses/Min)
:"+rsFollowup.getString("HeartBeat")+"\n"+"Temperature(celcious)
:"+rsFollowup.getString("Temperature")+"\n"+"Pressure(Dis/Sys)
:"+rsFollowup.getString("PressureDys")+"/"+rsFollowup.getString("PressureSys")+"\n
\n";
              //crosscheck this resultset with service parameters for this patient
              //send to the EventListener if it worths notification
             }
             rsFollowup.close();
              String sql5="";
              sql5 = "SELECT * FROM tbldiagnosis WHERE PatientId="+pId;
                ResultSet rsDiagnosis = st.executeQuery(sql5);
                while (rsDiagnosis.next())
            {
                message=message+"------Recent Diagnosis Information------
\n"+"Symptom(s)         :"+rsDiagnosis.getString("Symptoms")+"\n"+"Findings
:"+rsDiagnosis.getString("Findings")+"\n"+"Remarks
:"+rsDiagnosis.getString("Remarks")+"\n"+"Diagnosed on
:"+rsDiagnosis.getString("DiagDate")+"\n"+"Checkup Date
:"+rsDiagnosis.getString("CheckupDate");
              //crosscheck this resultset with service parameters for this patient
              //send to the EventListener if it worths notification
             }
              rsDiagnosis.close();
           }
*/
                pId=3;
                String sql="";
              sql = "SELECT Temperature,Heartbeat FROM tblfollowup WHERE
PatientId="+pId;
                ResultSet rsAbnormal = st.executeQuery(sql);

                while (rsAbnormal.next())
            {
                String abnormalValues="ABNORMAL VITAL SIGNS REPORTED FROM
ROOM 103 FOR PATIENT YARED
ALEMAYEHU\n"+"Temperature:"+rsAbnormal.getString("Temperature")+"\n"+"Heart
Beat   :"+rsAbnormal.getString("Heartbeat")+"\n";
                  message=message+abnormalValues;
              }
      rsAbnormal.close();
       conn.close();
      return message;
      }
}
```

C: Protégé OWL Ontology Code for Pervasive Healthcare Domain

```
<?xml version="1.0"?>
<rdf:RDF
   xmlns="http://www.owl-ontologies.com/pervasivehealthcare.owl#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns:p1="http://www.owl-ontologies.com/assert.owl#"
   xml:base="http://www.owl-ontologies.com/pervasivehealthcare.owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="Laboratory">
   <rdfs:subClassOf>
    <owl:Class rdf:ID="Activity"/>
   </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Prescription">
   <rdfs:subClassOf>
    <owl:Class rdf:about="#Activity"/>
   </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Outpatient">
   <rdfs:subClassOf>
    <owl:Class rdf:ID="Patient"/>
   </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="RangeParameter">
   <rdfs:subClassOf>
    <owl:Class rdf:ID="ServiceParameter"/>
   </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#Patient">
   <rdfs:subClassOf>
    <owl:Class rdf:ID="Person"/>
   </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#Person">
   <FirstName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   ></FirstName>
   <owl:disjointWith>
    <owl:Class rdf:ID="Location"/>
   </owl:disjointWith>
   <LastName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   ></LastName>
   <owl:disjointWith>
    <owl:Class rdf:about="#Activity"/>
   </owl:disjointWith>
   <owl:disjointWith>
    <owl:Class rdf:about="#ServiceParameter"/>
   </owl:disjointWith>
```

```
</owl:Class>
<owl:Class rdf:ID="Outdoor">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Location"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Staff">
  <rdfs:subClassOf rdf:resource="#Person"/>
</owl:Class>
<owl:Class rdf:ID="IdentificationParameter">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#ServiceParameter"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Inpatient">
  <rdfs:subClassOf rdf:resource="#Patient"/>
</owl:Class>
<owl:Class rdf:about="#ServiceParameter">
  <owl:disjointWith rdf:resource="#Person"/>
</owl:Class>
<owl:Class rdf:ID="LabTechnician">
  <rdfs:subClassOf rdf:resource="#Staff"/>
</owl:Class>
<owl:Class rdf:ID="Indoor">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Location"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Activity">
  <owl:disjointWith rdf:resource="#Person"/>
</owl:Class>
<owl:Class rdf:ID="Pharmacist">
  <rdfs:subClassOf rdf:resource="#Staff"/>
</owl:Class>
<owl:Class rdf:ID="PrecisionParameter">
  <rdfs:subClassOf rdf:resource="#ServiceParameter"/>
</owl:Class>
<owl:Class rdf:ID="SpecialParameter">
  <rdfs:subClassOf rdf:resource="#ServiceParameter"/>
</owl:Class>
<owl:Class rdf:ID="Nurse">
  <rdfs:subClassOf rdf:resource="#Staff"/>
</owl:Class>
<owl:Class rdf:ID="Doctor">
  <rdfs:subClassOf rdf:resource="#Staff"/>
</owl:Class>
<owl:Class rdf:about="#Location">
  <owl:disjointWith rdf:resource="#Person"/>
</owl:Class>
<owl:Class rdf:ID="Diagnosis">
  <rdfs:subClassOf rdf:resource="#Activity"/>
</owl:Class>
<owl:Class rdf:ID="Followup">
```

```
  <rdfs:subClassOf rdf:resource="#Activity"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="CurrentLocation">
  <rdfs:range rdf:resource="#Indoor"/>
  <rdfs:domain rdf:resource="#Staff"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="LocatedAt">
  <rdfs:range rdf:resource="#Indoor"/>
  <rdfs:domain rdf:resource="#Inpatient"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="PId">
  <rdfs:domain rdf:resource="#Activity"/>
  <rdfs:range rdf:resource="#Patient"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="Treates">
  <rdfs:domain rdf:resource="#Doctor"/>
  <rdfs:range rdf:resource="#Patient"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="SId">
  <rdfs:range rdf:resource="#Staff"/>
  <rdfs:domain rdf:resource="#Activity"/>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="Weight">
  <rdfs:domain rdf:resource="#Followup"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="FirstName">
  <rdfs:domain rdf:resource="#Person"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="AdmissionDate">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#date"/>
  <rdfs:domain rdf:resource="#Patient"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="AssignedDate">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#date"/>
  <rdfs:domain rdf:resource="#Staff"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="HeartBeat">
  <rdfs:domain rdf:resource="#Followup"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="Pressure">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#Followup"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="PatientId">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
  <rdfs:domain rdf:resource="#Patient"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="Latitude">
  <rdfs:domain rdf:resource="#Location"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
```

```
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="Temperature">
  <rdfs:domain rdf:resource="#Followup"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="LastName">
  <rdfs:domain rdf:resource="#Person"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="StaffId">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
  <rdfs:domain rdf:resource="#Staff"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="ActivityDate">
  <rdfs:domain rdf:resource="#Activity"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#date"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="Height">
  <rdfs:domain rdf:resource="#Followup"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="Findings">
  <rdfs:domain rdf:resource="#Diagnosis"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="Longitude">
  <rdfs:domain rdf:resource="#Location"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="RoomNo">
  <rdfs:domain rdf:resource="#Indoor"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="Remarks">
  <rdfs:domain rdf:resource="#Diagnosis"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="Symptoms">
  <rdfs:domain rdf:resource="#Diagnosis"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<Inpatient rdf:ID="Patient2">
  <FirstName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Meron</FirstName>
  <PatientId rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
  >0</PatientId>
  <LastName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Getaneh</LastName>
  <AdmissionDate rdf:datatype="http://www.w3.org/2001/XMLSchema#date"
  >2007-06-10</AdmissionDate>
  <LocatedAt>
    <Indoor rdf:ID="Indoor1">
      <RoomNo rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
```

```
    >101</RoomNo>
   </Indoor>
  </LocatedAt>
</Inpatient>
<Indoor rdf:ID="Indoor2">
  <RoomNo rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >102</RoomNo>
</Indoor>
<Indoor rdf:ID="Indoor3">
  <RoomNo rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >103</RoomNo>
</Indoor>
<Doctor rdf:ID="Doctor1">
  <Treates>
   <Inpatient rdf:ID="Patient3">
    <AdmissionDate rdf:datatype="http://www.w3.org/2001/XMLSchema#date"
    >2007-06-01</AdmissionDate>
    <LocatedAt rdf:resource="#Indoor3"/>
    <FirstName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    ></FirstName>
    <PatientId rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >0</PatientId>
    <LastName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    ></LastName>
   </Inpatient>
  </Treates>
  <Treates rdf:resource="#Patient2"/>
  <AssignedDate rdf:datatype="http://www.w3.org/2001/XMLSchema#date"
  >2007-02-19</AssignedDate>
  <LastName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Kebede</LastName>
  <Treates>
   <Inpatient rdf:ID="Patient1">
    <AdmissionDate rdf:datatype="http://www.w3.org/2001/XMLSchema#date"
    >2007-06-19</AdmissionDate>
    <LastName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Haile</LastName>
    <PatientId rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</PatientId>
    <LocatedAt>
     <Indoor rdf:ID="Indoor4">
      <RoomNo rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >201</RoomNo>
     </Indoor>
    </LocatedAt>
    <FirstName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    ></FirstName>
   </Inpatient>
  </Treates>
  <StaffId rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
  >1</StaffId>
  <CurrentLocation rdf:resource="#Indoor1"/>
  <FirstName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
```

```
  >Daniel</FirstName>
</Doctor>
<Followup rdf:ID="Followup1">
  <Weight rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
  >79</Weight>
  <SId rdf:resource="#Doctor1"/>
  <Height rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
  >175</Height>
  <HeartBeat rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
  >73</HeartBeat>
  <Pressure rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >120-80</Pressure>
  <Temperature rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
  >37</Temperature>
  <ActivityDate rdf:datatype="http://www.w3.org/2001/XMLSchema#date"
  >2007-06-19</ActivityDate>
  <PId rdf:resource="#Patient1"/>
</Followup>
<Diagnosis rdf:ID="Diagnosis2">
  <ActivityDate rdf:datatype="http://www.w3.org/2001/XMLSchema#date"
  >2007-06-11</ActivityDate>
  <PId rdf:resource="#Patient1"/>
  <Findings rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Hyper Tension</Findings>
  <Symptoms rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Head Ache</Symptoms>
  <Remarks rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Pain Killer</Remarks>
  <SId rdf:resource="#Doctor1"/>
</Diagnosis>
<Indoor rdf:ID="Indoor6">
  <RoomNo rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >203</RoomNo>
</Indoor>
<Indoor rdf:ID="Indoor5">
  <RoomNo rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >202</RoomNo>
</Indoor>
<Diagnosis rdf:ID="Diagnosis1">
  <Findings rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  ></Findings>
  <Symptoms rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  ></Symptoms>
  <PId rdf:resource="#Patient2"/>
  <Remarks rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  ></Remarks>
  <SId rdf:resource="#Doctor1"/>
  <ActivityDate rdf:datatype="http://www.w3.org/2001/XMLSchema#date"
  >2007-05-10</ActivityDate>
</Diagnosis>
        </rdf:RDF><!-- Created with Protege (with OWL Plugin 2.2, Build 311)
                              http://protege.stanford.edu -->
```