Ontological support for managing non-functional requirements in pervasive healthcare

Nigel Koay[1]
Pavandeep Kataria[1]
Radmila Juric[1]
Patricia Oberndorf[2]
Gabor Terstyanszky[1]

[1] School of Informatics, University of Westminster
[2] Software Engineering Institute, Carnegie Mellon University

Ontological Support for Managing Non-Functional Requirements In Pervasive Healthcare

Nigel Koay, Pavandeep Kataria, Radmila Juric, Patricia Oberndorf[1], Gabor Terstyanszky
School of Informatics, University of Westminster, 115 New Cavendish Street, London, W1W 6UW, UK
[1]Software Engineering Institute, Carnegie Mellon University, 4500, Fifth Ave, Pittsburgh, PA 15213 – 3890, USA
N.Koay@student.wmin.ac.uk; P.Kataria1@wmin.ac.uk; juricr@wmin.ac.uk; po@sei.cmu.edu[1]; terstyg@wmin.ac.uk

Abstract

We designed and implemented an ontological solution which makes provisions for choosing adequate devices/sensors for remote monitoring of patients who are suffering from post-stroke health complications. We argue that non-functional requirements in pervasive healthcare systems can be elicited and managed through semantics stored in ontological models and reasoning performed on them. Our contribution is twofold: (i) we enrich the elicitation and specification of non-functional requirements within the requirements engineering discipline and (ii) we address the pervasiveness of healthcare software systems through the way of choosing devices embedded in them, and through user expectations in terms of having access to pervasive services personalized to their needs.

1. Introduction

Requirements engineering (RE) has come of age as a computer science discipline in spite of numerous discussions on "what exactly we engineer in requirements", or "how much is requirements engineering a research discipline" [1, 2, 3]. We have known for decades that we depend on correctly elicited, understood, analyzed, verified, agreed and document requirements, because they are the backbone of successfully implemented software systems [4, 5]. Unfortunately, requirements engineering communities have not created a silver bullet which could guarantee that, if we perform requirements analysis and requirements engineering as a discipline, it would lead to the delivery of software systems that fully meet documented requirements [6, 7, 8]. Furthermore, the changing nature of software applications, with the proliferation of web-based software systems plus our acceptance that ubiquity [9] and pervasiveness [10] are the way we compute in the 21st century, have put a greater burden on the problem of eliciting and managing requirements. The plethora of works in the field range from the Persona-Scenario-Goal methodology in user centered requirements engineering for a web applications system [11], requirements which deal with legislation, regulations, policy and regulatory compliance essential for requirements engineers and compliance auditors [12] and requirements for open service systems, which have to be updated at run time, [13] to works on requirements modeling in component-based systems [14], frameworks for requirements elicitations in integrated service creation environments in telecommunication networks [15] and applications of language patterns to an object oriented knowledge representation to improve the quality of requirements specification [16], to mention just a few.

Our research concentrates on the requirements problem in an area which spans the pervasiveness of software systems and the domain of healthcare. Pervasive systems are characterized by a multitude of computational, communication, wearable and handheld devices, which are embedded in a pervasive computational environment and which require users to become part of a corresponding software infrastructure, when plugging-in their devices. Therefore the emphasis is on non-functional requirements of pervasive systems [17, 18], because the developers of such systems need to generate requirements that

- enable analysis of their pervasiveness within the problem definition and according to user preferences and
- assist in delivering context awareness [19].

There is a generic list of requirements for pervasive software infrastructure in [20] in terms of assembling intelligent environments from distributed devices and software components. They range from extensibility, exchangeability and autonomy of devices/components/intelligent systems to decentralization and conflict resolution mechanisms. It is obvious that the issues of

(a) devices embedded in pervasive systems and

978-0-7695-3450-3/09 $25.00 © 2009 IEEE

(b) user's roles and expectations in terms of having access to pervasive services, personalized to their needs, which are non-intrusive and which operate seamlessly within the pervasive environment

require more comprehensive analysis of non-functional requirements than is possible using the traditional techniques for requirements analysis RE.

To illustrate the problem of managing non-functional requirements in pervasive healthcare, we give a scenario of remote monitoring of patients who are suffering from post-stroke health complications. Our goal is to elicit requirements that will enable us to choose the most appropriate devices for patient monitoring in terms of their functionality, obtrusiveness, interfaces, communicational and computational facilities, etc.

In section 2 we introduce the problem of non-functional requirements in pervasive healthcare and give a scenario that illustrates the purpose of building a remote patient monitoring system (RPMS) for patients suffering from complications of strokes. Section 3 gives an overview of related works. Section 4 and 5 describe ontologies for storing semantics of non-functional requirements for the RPMS and for classification of device/sensors characteristics that might fit these non-functional requirements. Section 6 describes the implementation process in terms of integrating the two ontologies and creating reasoning rules which help us to make decisions on the most suitable device for a particular use of the RPMS. We conclude in section 7.

2. The Problem

There are many works which address non-functional issues in requirements elicitation and management [21, 22, 7, 23, 24, 25]. Software performance, quality, satisfaction, constraints, representations and similar have been accepted in the requirements engineering community as the backbone of non-functional requirements. They have also been analyzed/categorized as part of a concern-based taxonomy of non-functional requirements in [26]. However, the same author laments that there is no consensus on what the non-functional requirements are and how we should manage them. Therefore, [26] is an excellent source for underpinning discussions on differentiating between functional and non-functional requirements and the way we may categorize the latter, which is outside the scope of our work.

The problem we would like to address here is that

(i) pervasive systems rely heavily on non-functional requirements [17, 18]; therefore they require early elicitation, analysis and verification
(ii) pervasiveness dictates (a) and (b) from the introduction, which means that the choice of embedded devices, which make up a pervasive environment, will have an impact on user adaptation and experiences of such systems and successful delivery of functionality built within them.

Therefore we deal with issues like: obtrusiveness, wearability, mobility, computations, ease of communication and exchange of information, interoperability versus heterogeneity, self-tuning and self-configuring, etc. In the next section we give a scenario that illustrates the problem from (i) and (ii).

2.1. The RPMS and its Purpose

This example of the RPMS concerns patients who are suffering from health complications caused by strokes. Therefore our RPMS is located at a patient's home, which is equipped with adequate monitoring devices, communication and computational devices, persistent storage and software applications that enable patients to be monitored remotely by healthcare professionals. The results of remote monitoring, which may include alerts, advices and decision making, may be stored within the home computing system, but are available for healthcare professionals, whose expertise is essential for the RPMS.

From that perspective, the main purpose of RPMS is to give an objective measure of the patient's status at any given time and to assist healthcare professionals in assessing the patient's current status, determining the patient's therapeutic strategy and its changes, managing alerts and abnormal situations that might occur when monitoring patients, etc. In order to give an objective measure of the patient's status, we need to measure a variety of conditions, experiences, feelings, disabilities, etc. for each patient. If we add to that the specificity of problems that patients suffering from post-stroke complications may have, then we have to add tasks of issuing alerts if symptoms of new/repeated stroke have been detected and assisting patients in following the prescribed medication and stroke rehabilitation therapies.

Therefore, MONITORING patients in RPMS means MEASURING a variety of conditions and situations, which are specific for such patients. We need to decide what exactly has to be measured and how. The first step is to systemize FACTORS which will determine what is to be MEASURED through

MONITORING within the RPMS. They are systemized in A-F below:

A. which **type of stroke** the patient had;
B. which **surgery** has been carried out after the stroke;
C. which **disability** has resulted from the stroke;
D. which **symptoms** patients may have while being monitored in post-stroke condition (this may include symptoms of new or repeated strokes)
E. which **medication** therapy has been prescribed by the consultant;
F. which **rehabilitation** therapy has been suggested and advised by the consultant (rehabilitation may be in the form of **therapies** and /or **learning skills**);
G. which **personal choice** each patient might exercise in terms of **using** the RPMS.

Apart from A-G above, the process of MONITORING patients and obtaining MEASURES of a patient's status could be done through TWO different MODES:

a) Patients can be monitored through **S**ENSORS, which are often incorporated into devices of varied communication and computational power, which are embedded into patient pervasive systems;
b) Patients can give their own, subjective input in terms of REPORTING to the RPMS their feelings/experiences/results of rehabilitation exercises/failures to take medications, etc.

Finally it is obvious that the monitoring is expected to be specifically tailored for a particular patient (personalized) and should include a set of devices and sensors/commutations/computations /interfaces which ***fit the personalized picture of the patient's needs within the RPMS***. Furthermore, we can add more personal choices to the RPMS by allowing patients to make certain decisions which will enable them to decide about their privacy, situations during the monitoring time/sessions, or improving their personalized RPMS to fit their needs better. Note: "privacy" in RPMS primarily means allowing patients to agree on what is to be monitored, when and how they will be monitored. It also means adding patients' requests in terms of devices' unobtrusiveness, interface expressiveness and similar. From that perspective "privacy" issues interfere with personalization of pervasive healthcare services within our RPMS.

The most important part of the RPMS is to ensure that any symptom(s) of another stroke are detected early and adequate actions taken immediately. The RPMS should have the ability to

(I) **Measure** potential loss of balance, dizziness, falls, and weakness/numbness of the muscles through a variety of sensors. This means that devices/sensors will generate data, which in turn can be analyzed and alerts issued as soon as "dangerous anomalies" have been detected, and

(II) Accept the patient's input based on his/her own perception of his/her "health status" in terms of **reporting** loss of sight, speech problems, headaches, loss of coordination and problems with any of his/her mental functions (confusion, understanding difficulties, etc.)

(III) Generate alerts in audio format, with an option to have a step-by-step "visual guide", through a user friendly GUI, on what the alert means and how to proceed.

2.2. Problem Formulation

In the previous section we have highlighted the basic functionality of RPMS, which is realized through specific measuring of a variety of conditions and situations for a particular patient. Therefore the factors that determine what is to be measured and factors that determine the nature of the devices that are suitable for such measurements are non-functional requirements, such as A.-F. and a) and b) from section 2.1. Furthermore, the objective measure of the patient status at any time may include feelings and experiences of patients, which are then adapted to situations the patient may happen to participate in. Measurements are also dependent on the patient's disabilities and medical conditions and are based on the patient's preferences or personal choices in terms of using the RPMS.

Therefore at the heart of RPMS is personalized monitoring and measuring, which cannot happen without correctly chosen devices/sensors. The choice of devices, as a consequence of semantics stored in non-functional requirements, will determine further requirements and their elicitation for the pervasive system and software applications that are supposed to carry out its functionality.

However, our problem is threefold:

I. There is a plethora of devices available for remote health monitoring in the marketplace, but they differ significantly in terms of their suitability for the RPMS requirements depicted in Section 2.1. Therefore it is difficult to choose correct devices/sensors without analyzing the semantics stored in RPMS requirements. For example, if we want to monitor a patient's vital signs, do we choose devices that measure each individual vital sign separately, or do we choose a single device that encompasses all measurements?
II. There are situations in remote monitoring, which may include cognitive measurements and personal

choices of patients that in turn might affect the functionality of such systems. Therefore the choice of devices/sensors involved in remote monitoring is directly responsible for building functionality of such pervasive systems. For example, if we want to monitor patient's 'mood' to assess his/her depression, then we can either use devices/sensors that support the management of "face expressions" or we can allow the patient to enter "cognitive data" into the system (see modes from a) and b) in Section 2.1.). The choice of devices will determine the functionality of the software applications which support the RPMS, because if we reject patient's individual input (see b) in section 2.1) we will have to build reasoning mechanisms for interpreting and managing data collected solely through sensors and then probably manage "face expressions" (amongst others!) to support the RPMS.

III. Non-functional requirements in RPMS are difficult to structure. As they are difficult to grasp initially, they are often expressed in non-standard ways; particularly in the case of choosing correct devices/sensors, non-functional requirements might be overlooked or understated. For example, we have tried to use bullets and numbering to structure non-functional requirements in Section 2.1. We have used letters (A-G), a)-b), (I) – (III) numbers 1-6, bullets, but we probably did not contribute significantly to the understanding on what non-functional requirements for the RPMS are. Furthermore, we could not place our knowledge of non-functional requirements within all these bulleted lists, because they cannot show relationships between list items, priorities, their significance, correlations, etc.

It appears that by encapsulating our knowledge of all kinds of non-functional requirements and relationships between them in RPMS, we may be able to address problems I.-III. This means that we need a semantically rich environment based on ontologies that can juxtapose requirements of the RPMS with characteristics of devices/sensors that may be used in such systems. This could be a basis for making correct decisions about the devices/sensors from which we build our pervasive systems.

3. Related Works

We list the works that might relate to ours. The readers who are more interested in the role of non-functional requirements in pervasive systems in general should look at [17, 18, 20]. For illustration of the latest works in pervasive systems and assessment of research done on requirements, we suggest reading the IEEE journal/conferences on pervasive systems published in 2007. The works which use ontologies for eliciting and managing requirements in general can be found in [27, 28, 29, 30]. We could not find any related work that would use ontologies in order to make correct choices of devices that make up pervasive environments in healthcare. Therefore we overview a few healthcare projects that are pervasive, i.e., they depend on devices embedded within them, they may cover remote patient mentoring, and they may use ontology for managing semantics of embedded devices and sensors in such pervasive systems.

R.B. Jan Borchers et al. from [31] have surveyed different mobile phone interaction techniques as input devices in a ubiquitous environment. They have used a taxonomy as a framework for their analysis of smart phones. The framework is structured around the tasks mobile input devices can perform and was adapted to suit "smart phones". A smart phone was then placed in various positions to gather data and assess how users interact with their environment. Therefore their taxonomy for analyzing features of mobile phones becomes a basis for deciding on features and functionalities of "smart phones".

F. Paganelli and D. Giuli [32] give an ontology-based context model for health monitoring and alert management for a system supporting patient care at home, through the use of devices and reasoning mechanisms built upon the devices. They focus on rule-based reasoning for analysis of measurement data values collected by devices. The reasoning on context is used mainly to trigger alarms, based on vital sign measurements and the situations in a home environment. Although reasoning is used to identify any important event, it can also be adapted to aid in selecting particular devices for home use.

E-J. Ko et al. [33], which proposes a context-aware framework for running applications on an embedded wearable system in ubiquitous healthcare environments, guarantees the independence of a services and devices in the system and mediates context information provided by sensors.

J. O'Donoghue et al. [34] present the Tyndall-DMS-Mote, a wireless, low cost sensing device for monitoring vital signs in a non-intrusive manner, inside or outside of a home. A rule-based system is run to trigger predefined actions for a particular scenario and the mobile client communicates wirelessly with the wireless device, collecting patient readings and sending them to the server.

None of these works above could relate firmly to our research in terms of addressing I-III from 2.2

through ontologies. There is only one paper that could support our decision to build ontologies and reason about them in order to make correct choices of devices/sensors when building pervasive healthcare environments like the RPMS. [28] describes an ontology for non-functional requirements that can be used to enhance RE practices in service-oriented systems. The ontology is used as a unified conceptual model for non-functional requirements specification. They argue that a non-functional requirements ontology helps to structure and express constraints as part of quality of service specifications in service-oriented systems. From that perspective, we share their views that

- non-functional requirements define the overall qualities of any system, particularly pervasive systems and
- non-functional requirements are of critical importance, therefore functional requirements may need to be sacrificed to meet them.

4. Ontology for Storing Semantics of Non-Functional Requirements

To address the problems from (I)-(III) in section 2.2 we have decided to use ontologies and create a reasoning mechanism upon them in order to choose the best monitoring device for a particular patient when using a RPMS. We started by creating two separate ontologies and name them as Req-ONTO and Dev-ONTO. Req-ONTO stores semantics of non-functional requirements for the RPMS as described in section 2. Dev-ONTO stores semantics of all characteristics of remote monitoring devices (and sensors) which might be needed when building a RMPS. By juxtaposing the semantics stored in these two ontologies we have been able to integrate them in terms of matching (a) particular requirement(s), which a patient/healthcare professional might have within the RPMS, with the choices of devices which fit those requirements. In our paper we use the term 'integrating ontologies' for finding this match between requirements of a RPMS and devices which can be used within it. In this section we give an overview of Req-ONTO. In Section 5 we give an overview of Dev-ONTO.

Figure 1 shows Req-ONTO basic classes and their subclasses. These main concepts are based on requirements described in section 2, i.e. the basic classes in Figure 1 mimic factors found under A-G labels in section 2.1. Req-ONTO consists of eight super-classes; *Disabilities, Medication, New-Stroke-Symptoms, Patient, Personal-Choice, Rehabilitation-Regime, Stroke-Types and Post-Stroke-Surgery*. The names of these basic concepts are self-explanatory. We give a rationale behind a few ontological concepts below.

In Section 2.1 we state that we need to measure a variety of patient's conditions. The semantics of such measuring is described through the following classes of Req-ONTO:

- Conditions are modeled through *Medication, Stroke-types* and *Post-Stroke-Surgery* ontological classes. This is because a combination of all these three classes is responsible for having a clear picture of the types of measurements we need to take and the types of devices that might be suitable for them.
- Experiences and feelings are modeled through the *Personal-Preferences* class which makes provisions for user/patient inputs.
- Disabilities are modeled through the *Disability* class that defines the semantic behind a number of disabilities that may happen as a consequence of stroke.

Determining a patient therapeutic strategy, as a consequence of a particular stroke, is modeled through a *Rehabilitation* class in Req-ONTO. The same class also defines rehabilitation therapies as a part of *Learning-Skills*. The semantics for cases of TIA (symptoms of possible new or repeated strokes) are modeled through the New-Stroke-Symptoms class. The same can be said for medications, which are modeled through the *Medication* class and the semantic of rehabilitation therapies are modeled through the *Rehabilitation-Program* class.

The *Rehabilitation-Program* class also holds details of a patient's rehabilitation program, listing their daily schedule of activities. One of the subclasses of the *Rehabilitation-Program* class is *Learning-New-Skills*, which gives all the different skills that a patient who has suffered from a stroke can go through to re-adapt to his or her normal life.

It is important to take note that the *Patient* and *Personal-Preference* classes hold personal data of the patient and the patient's personal monitoring preferences. In reality, the personal preferences of a patient are usually specified by the patient and computed by an application. In our model, we store these preferences within an ontology. The *Personal-Preference* class has two subclasses: *Environmental-Preference* and *Monitor-Time-Preference*. The *Environmental-Preference* lists the different places where the patient might be while using the monitoring device and the *Monitor-Time-Preference* class specifies the time of the day when the patient would like to be monitored and which type of measurements have been agreed to take place.

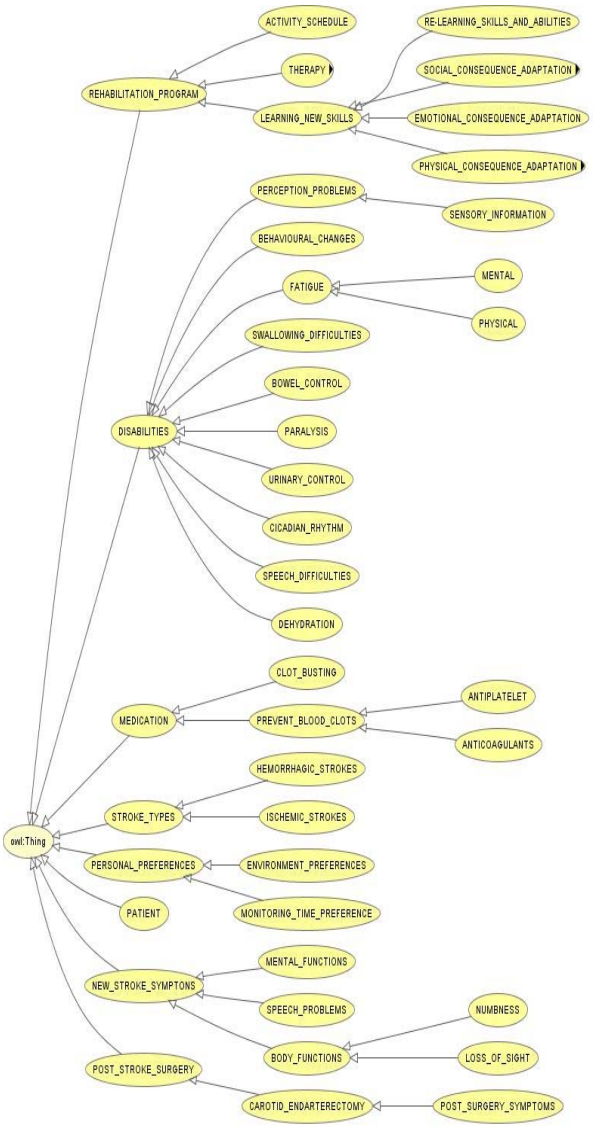

Figure 1: Req-ONTO: Ontology with Semantics of Requirements for RPMS

5. Ontology for Classifying Devices

The device ontology (Dev-ONTO) has been derived from categorizing a collection of healthcare devices used for monitoring various human functions. To underpin our modeling constructs, we had to read and analyze numerous academic papers, industrial white papers, web sites which advertise devices etc. Apart from providing a set of characteristics found in the different devices, Dev-ONTO stores semantics of our non-functional requirements given in Section 2 in order to classify Dev-ONTO's ontological concepts and relationships between them. For example, in (i) – (iv) below we show how we modeled the impact of our RPMS on patient's adaptation and experiences of using RPMS when classifying characteristics of devices and sensors:

(i) The *Characteristics-of-Sensor-Device* class has two subclasses: *Display-Type* and *Privacy*. The *Display-Type* subclass indicates all the different types of displays a device's interface may have. The *Privacy* subclass is seen as storing a characteristic of sensor-based devices, because it describes the level of 'physical' privacy that a device allows to a user (thus it has been modelled as a single 'node' in the Dev-ONT ontology). Therefore, it can hold the values of being 'invasive' (e.g. we may say the invasive devices heavily interact with the user and the user is very much aware of device presence in his/her everyday life) and 'Non-Invasive' (we may say that devices will perform their monitoring tasks, but without any interaction with the user). However, when creating ontological instances, there should be a consensus on what exactly this interaction means and how it affects the user's own privacy, i.e. should we consider CCTV as being non-intrusive and handheld devices (or sensors implanted below the skin) as being intrusive?

(ii) The semantics behind "wearability" is modeled as the *Embedded* subclass of the *Characteristics-of-User-Input-Device* class.

(iii) The semantics behind "mobility" of devices are modeled as the *Portability* subclass of the *Characteristics-of-User-Input-Device* class. Therefore, the *Characteristics-of-User-Input-Device* class has three subclasses: *Embedded*; *Input-Capabilities* and *Portability*. The *Embedded* class indicates where the device is located while being used. The *Input-Capabilities* class lists all the different methods that can be used by the user when entering any information into a RPMS through the device. The *Portability* sub-class indicates the ease with which a device can be moved.

(iv) The semantics behind "computations" and "communications" are modeled as the *Computational* subclass of the *Interface-Type* class. This is because the majority of devices are both communicational and computational. However, the semantics behind the communicational devices fit within the *Input-Capabilities* subclass of the *Characteristics-of-User-Input-Device* class. Furthermore the *Interactive* class is where a device interface allows the user to interact with it and ultimately with the RPMS. This can be in the form of buttons, communicational video uplinks or

microphone speaker. The *Computational* class is where a device interface has the ability to compute readings it has gathered. Note: we have decided to model wireless computation to indicate that a device can compute all readings by itself and non-wireless computation to indicate that a device must be connected to a base computer for computation to happen.

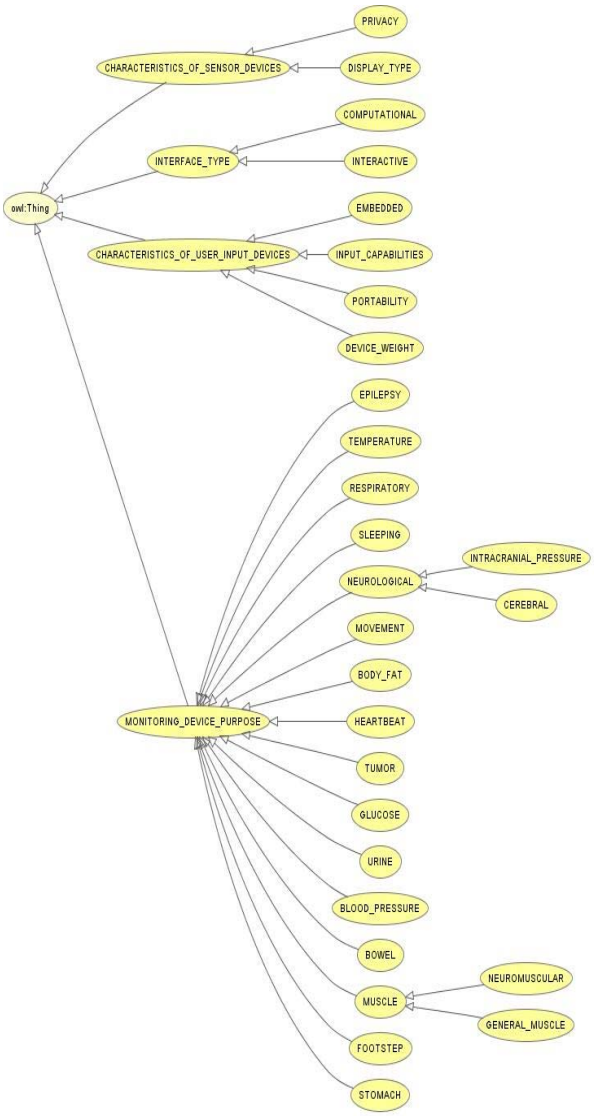

Figure 2 Dev-Onto: Ontology with Semantics of Device Characteristics

The *Monitoring-Device-Purpose* class models semantics equivalent to a variety of symptoms the device can measure (see D from Section 2.1), which may include symptoms of new or repeated strokes). The way these symptoms have been categorized as subclasses of the *Monitoring-Device-Purpose* class corresponds to symptoms and measuring, which can be found in (I)-(III) from Section 2.1.

6. Implementation

We have used OWL and its modeling constructs to create the two ontological models described in Sections 4 and 5. OWL uses the properties of classes to create restrictions based on the taxonomical relationships that have been defined between the classes and their properties. The ontological model was implemented using the Protégé 2000 Ontology Editor, version 3.4 [35]. The DIG Reasoner Inspector [36] was used to test the consistency of the ontology's classes/properties/instances. The SWRL rules were implemented through the SWRLTab plugin in Protégé, and executed using the Jess Rule Engine [37]. The implementation consists of:

1. Integration of two ontologies (section 6.1.) and
2. Performing reasoning upon integrated ontologies through rules written in SWRL (section 6.2)

6.1. Integrating the two ontologies

We have integrated the two ontologies to find a match between Req-ONTO and Dev-ONTO. For example, if in a particular ontological instance, Mr Smith is required to be constantly monitored for falls and weakness in muscles, he must have a device/set of devices that measure muscles weaknesses appropriately. Therefore a match between objective measure of the patient status at any given time (from Req-ONTO) and devices that can handle such measurements (from Dev-ONTO) is needed. The matching can be done by running reasoning rules.

6.2. Reasoning Rules upon Ontologies

We use the OWL-DL ontology language to model the semantics of ontological schemas and SWRL and SQWRL to model the rules needed for reasoning. SQWRL is a SWRL-based language that allows for more detailed querying of OWL ontologies. We illustrate 4 reasoning rules upon the Dev-ONTO and Req-ONTO ontologies. These are shown in Figures [3-6]. The rules can be altered to accommodate more/different semantics about the patients/devices and match instances from both ontologies.

Figure 3 describes a SQWRL rule that determines possible devices that a patient can use for monitoring a particular disability. The rule takes into account the personal monitoring preferences of the patient, the disability in question and characteristics of devices that the patient prefers. The rule then returns a set of devices that meet the criteria and therefore can be selected. The rule can be altered to accommodate any other type of disability or multiple disabilities.

PATIENT_DEVICE_CHOICE:

PATIENT (?a) ∧
Has_Disabilities (?a, Torso_Muscles)
∧ Environment_Preference (?a, On_Skin)
∧ Monitoring_Time_Preference (?a, Awake)
∧ Dev:NEUROMUSCULAR (?b)
∧ Dev:embedded_characteristics (?b, Dev:Person)
∧ Dev:device_weight (?b, Dev:Light)
∧ Dev:portability_characteristics (?b, Dev:Mobile)
∧ Dev:Description (?b, ?dd) ∧
swrlb:contains (?dd, "Torso Muscle Monitor")
∧ sameAs (?dd, Torso_Muscles)

→ query:select (?a, ?b, "Patient and Device Choices")

Figure 3. Patient Device Choice Rule

Figure 4 describes a SQWRL rule that determines possible devices a patient can use for monitoring their rehabilitation program. The rule takes into account the personal monitoring preferences of the patient, rehabilitation activity and characteristics of devices that are suited for the activity. The rule then returns a set of devices that meet the criteria and can be selected.

PATIENT_REHAB_DEVICE_CHOICE:

PATIENT (?a) ∧
Rehab_Activities (?a, Pelvic)
∧ Monitoring_Time_Preference (?a, Awake) ∧
Patient_Device_Preference (?a, ?y) ∧
∧ swrlb:matches (?y, Use_PC) ∧
Dev:FOOTSTEP (?b) ∧
Dev:computational_type (?b, Dev:Non_Wireless_Computation)
∧ Dev:portability_characteristics (?b, Dev:Mobile)
∧ sameAs (?y, Dev:Non_Wireless_Computation)

→ query:select (?a, ?b, "Patient and Rehab Device Choices")

Figure 4. Patient Rehab Device Choice Rule

Figure 5 describes a SQWRL rule that determines a particular device that monitors a patient's movement, based on preferences specified by the patient. This rule takes into account the device's embedded characteristics, weight and interactive functions. The rule then returns the particular device that monitors movement which matches all the preferences specified by the patient.

Figure 6 describes a SQWRL rule that determines a patient's preferences. The rule takes into account the monitoring environment, time and device preference of a patient.

PATIENT_SPECIFIED_FUNCTIONS_DEVICE_CHOICE

PATIENT (?a) ∧
Dev: MOVEMENT (?b) ∧
Dev:PRIVACY (?b, ?c)
∧ swrlb:matches (?c, Dev:Invasive) ∧
Dev:embedded_characteristics (?b, Dev:Person)
∧ Dev:device_weight (?b, ?d) ∧
swrlb:matches (?d, Dev:Light) ∧
Dev:INTERACTIVE (?b, ?e) ∧
swrlb:matches (?e, Dev:No_Alarm)

→ query:select (?a, ?b, "Patient and Movement Device")

Figure 5. Function Device Choice Rule

PATIENT_PREFERENCES:

Environment_Preference (?a, Under_Skin)
∧ Monitoring_Time_Preference (?a, Sleeping)
∧ Patient_Device_Preference (?a, ?x) ∧
swrlb:matches (?x, Use_PC)

→ PATIENT (?a) ∧ query:select (?x)

Figure 6: Patient Preferences Rule

7. Conclusions

In this paper we explore the possibilities of using ontologies and web semantic tools in order to deal with non-functional requirements in pervasive healthcare. We use a specific scenario of remote monitoring of patients who are suffering from post-stroke health complications to illustrate our research. The scenario generates a pool of non-functional requirements for such a system, which require making a correct choice of sensors/devices in order to monitor patients remotely. We have created two ontologies: Req-ONTO and Dev-ONTO which store semantics of non-functional requirements imposed on our RPMS and the characteristics of devices/sensors which may have been essential for creating such a system. The choice of (an) adequate device(s) is based on reasoning on both ontologies, i.e., through matching the characteristics of available devices with the non-functional requirements of our RPMS.

We have not found any similar work which deals with the selection of embedded devices in pervasive healthcare through semantically rich environments and ontologies for reasoning about them. However, non-functional requirements in pervasive systems are often of critical importance and should be taken seriously, because they may affect the delivery of their functionality. Therefore our idea to use ontologies and reasoning over them, in systemizing unstructured non-functional requirements and

ultimately choosing the most suitable devices for RPMS, proved to be a good starting point in building personalized pervasive services in healthcare.

The work given in this paper is a part of our research into semantic management of requirements in pervasive healthcare. We have been motivated to create semantically rich environments for managing requirements in pervasive healthcare because:

1. We need a systematic description and set of characteristics of pervasive computing environments (PCEs), which not only will give a consensus on what exactly our healthcare computational environments for remote patient monitoring are, but will also enable us to structure an initial set of requirements that must be met if we want to call our RPMS 'pervasive'. From this perspective a correct choice of devices that make up any PCE and the assessment of user acceptance of and participation in such environments are a starting point.
2. We need to look at the traditional division of requirements into categories of functional and non-functional in PCEs and see whether we really need that barrier between them, and if so, then address how we can articulate their relationships. The specificity of PCEs dictates the existence of many requirements, traditionally described as non-functional, which become the backbone of such systems, and which push functionality of PCEs into second place. Indeed, when dealing with the choice of devices that make up our RPMS we were more concerned with the user's acceptance of it than with its functionality. Thus there should be a classification of requirements for PCEs that takes into account the specific nature of computation in pervasive healthcare, the purpose of such spaces, and the communicational/computational outcome from them.
3. We need to look at the semantic management of requirements in PCEs and investigate the use of ontologies and mechanisms for reasoning over them, if we want to carry on working on a human conceptual level when reasoning about the specific requirements for RPMSs (and pervasive healthcare in general). However, we need to create a consensus on the feasibility, role and purpose of ontology pruning, alignment and merging in PCEs. In this work we have used the word 'integration' for finding a match between Req-ONTO and Dev-ONTO, but it is debatable a) how correct this term is (i.e., it is more likely that we have performed an alignment of ontologies) and b) how much the semantics available in and generated by the reasoning sentences in SQWRL could have been stored as the basic concepts of Req-ONTO and Dev-ONTO ontologies in the first place. In other words, more work should be done in terms of incorporating results of reasoning within ontological concepts when managing the semantics of requirements in pervasive healthcare. This is the immediate focus of our future work.

Acknowledgements

We wish to thank Dr Ivo Dukic from the Royal Blackburn Hospital, UK for his suggestions on the applicability of remote monitoring in patients suffering from post-stroke health complications.

8. References

[1] H. Akkermans, J. Gordijn, 2006, "What is This Science Called Requirements Engineering? ", Proc. of the 14th IEEE Intl Conference, Minneapolis, USA, pp. 273 – 278.

[2] A.M. Davis, A.M., Hickey, 2002, "Requirements Researchers: Do We Practice What We Preach?", Requirements Engineering Journal, June 2002, Vol. 7, no 2, pp. 107 – 111.

[3] R. Wieringa, 2005, "Requirements Researchers: Are We Really Doing Research?", Requirements Engineering Journal, Nov. 2005, Vol. 10, no 4, pp. 304 – 306.

[4] B. Nuseibeh, S. Easterbrook, 2000, "Requirements Engineering: A Roadmap", Proc. of the Conference on The Future of Software Engineering, Ireland, pp. 35-46.

[5] B.H.C. Cheng, J.M. Atlee, 2007, "Research Directions in Requirements Engineering", Proc. of 29th the Intl. Conf. on Soft. Engineering, Minneapolis, USA, pp. 285 – 303.

[6] B. Berenbach, 2006, "Requirements Engineering: An Industrial Perspective", Proc. of the 14th IEEE Intl Requirements Engineering Conference, Minneapolis, USA, pp. 260 – 265.

[7] O. Dieste, N. Juristo, F. Shull, 2008, "Understanding the Customer: What Do We Know About Requirements Elicitation? ", IEEE Software, Apr. 2008, 25(2), pp. 11-13.

[8] J. Robertson, 2002, "Eureka! Why Analysts Should Invent Requirements", IEEE Software, Aug. 2002, 19(4) pp. 20 – 22.

[9] M. Weiser, 1991, "The Computer of the 21st Century", Journal of Scientific America Special Issue on Communications, Computers and Networks, Feb 1991, 265(3).

[10] M. Satyanarayanan, 2001, "Pervasive Computing: Vision and Challenges", IEEE Personal Communications, Aug 2001, 8(4), pp. 10 – 17.

[11] M. Aoyama, 2007, "Persona-Scenario-Goal Methodology for User-Centered Requirements Engineering", Proc. of the 15th IEEE Requirements Engineering Conference, Delhi, India, pp. 185-194.

[12] P. N. Otto, A. I. Anton, 2007, "Addressing Legal Requirements in Requirements Engineering", Proc. of the 15th IEEE Intl Requirements Engineering Conference, Delhi, India, pp. 5-14.

[13] I.J. Jureta, S. Faulkner, P. Thiran, 2007, "Dynamic Requirements Specification for Adaptable and Open Service Systems", Proc. of the 15th IEEE Intl Requirements Engineering Conference, Delhi, India, pp. 381 – 382.

[14] G. Kotonya, J. Hutchinson, 2007, "A Service-Oriented Approach for Specifying Component-Based Systems", Proc. of the 6th Intl IEEE Conference on COTS Based Software Systems, Madrid, Spain, pp. 152-162.

[15] C. Bo, X-W, Meng, J-L. Chen, 2007, "An Adaptive User Requirements Elicitation Framework", Proc. of the 31st Annual Intl Computer Software and Applications Conference, Vol. 2, Beijing, PROC, pp. 501-502.

[16] A. Ang, 2007, "Improving The Quality Of Knowledge Representation For Requirements Engineering Through Natural Language Requirements Patterns", Proc. of the 11th IASTED Intl Conference Software Engineering and Applications, Cambridge, MA, USA

[17] L. Kolos-Mazuryk, G-J. Poulisse, P.van Eck, 2005, "Requirements Engineering for Pervasive Services.", In the OOPSLA-workshop on Building Software for Pervasive Computing, San Diego, USA, Oct. 16, 2005.

[18] E. Niemela, J. Latvakoski, 2004, "Survey of Requirements and Solutions for Ubiquitous Software", Proc. of the 3rd Intl Conference on Mobile and Ubiquitous Multimedia, Maryland, pp. 71 – 78.

[19] A.K. Dey, G.D. Abowd, 2000, "The Contect Toolkit: Aiding the Development of Context-Aware Applications.", In the Workshop of the 22nd Intl Conference on Software Engineering, Limerick, Ireland, pp. 434 – 441.

[20] M. Hellenschmidt, 2006, "Some Issues on Requirements for Pervasive Software Infrastructures", Proc. of the Intl Conference on Pervasive Computing, Dublin, Ireland, pp. 549-552.

[21] M. Glinz, 2005, "Rethinking the Notion of Non-Functional Requirements", Proc. of the 3rd World Congress for Software Quality, Germany, Vol. 2, pp. 55 – 64.

[22] A. Desillets, 2008, "Tell Me A Story", IEEE Software, March 2008, 25(2), pp. 14-15.

[23] N. Maiden, 2008, "User Requirements and System Requirements", IEEE Software, March 2008, Vol. 25(2), pp. 90-91.

[24] X. Franch, P. Botella, 1998, "Putting non-functional requirements into software architecture", Proc. of the 9th Intl Workshop on Software Specification and Design, Ise-Shima, Japan, pp. 60 – 67.

[25] J. Mylopoulous, L. Chung, B. Nixon, 1992, "Representing and Using Nonfunctional Requirements: A Process-Oriented Approach", IEEE Transactions on Software Engineering, June 1992, 18(6), pp. 483 – 497.

[26] M. Glinz, 2007, "On Non-Functional Requirements", Proc. of the 15th IEEE Intl Requirements Engineering Conference, Delhi, India, pp. 21-26.

[27] W. Liu, K-Q. He, J. Wang, R. Peng, 2007, "Heavyweight Semantic Inducement for Requirements Elicitation and Analysis", Proc. of the 3rd Conf. on Semantics, Knowledge and Grid, X'ian, PROC, pp. 206–211.

[28] G. Dobson, S. Hall, G. Kotonya, 2007, "A Domain-Independent Ontology for Non-Functional Requirements", Proc. of the IEEE Intl Conference on e-Business Eng., ICEBE '07, Hong Kong, China, pp. 563–566.

[29] H. Kaiya, M. Saeki, 2006, "Using Domain Ontology as Domain Knowledge for Requirements Elicitation", Proc. of the 14th IEEE Requirements Engineering Conference, RE '06, Minneapolis, USA, pp. 186-195.

[30] L. Liu, C. Liu, C-H. Chi, Z. Jin, E. Yu, 2006, "Towards A Service Requirements Ontology on Knowledge and Intention", Proc. of the 6th International Conference on Quality Software, QSIC '06, pp. 452-462.

[31] R.B. Jan Borchers, M. Rohs, J. G. Sheridan, 2005, "The Smart Phone: A Ubiquitous Input Device", Pervasive Computing, IEEE, March 2005, Vol. 5, pp. 70-77.

[32] F. Paganelli, D. Giuli, 2007, "An Ontology-Based Context Model for Home Health Monitoring and Alerting in Chronic Patient Care Networks", Proc. of the 21st Intl Conf. on Advanced Information Networking Applications Workshop, Ontario, Canada, Vol. 2, pp. 838-845.

[33] E-J. Ko, H-J. Lee, J-W. Lee, 2007, "Ontology-Based Context Modeling and Reasoning for U-Healthcare", IEICE Transactions on Information and Systems 2007, 90(8), pp. 1262-1270.

[34] J. O'Donoghue, J. Herbert, P. Stack, 2006, "Remote Non-Intrusive Patient Monitoring", Proc. of the 4th Intl Conference on Smart Homes and Health Telematic, ICOST 06, Belfast, Northern Ireland, pp. 180-187.

[35] The Protégé Ontology Editor, available at http://protege.stanford.edu/, (accessed August 2008).

[36] DIG Reasoning API, available at http://protegewiki.stanford.edu/index.php/ProtegeReasonerAPI, (accessed August 2008).

[37] JESS, The Rule Engine available at http://herzberg.ca.sandia.gov/, (accessed August 2008).