Pontifícia Universidade Católica
do Rio de Janeiro

Vitor Pinheiro de Almeida

Patient-Buddy-Build: Customized Mobile Monitoring for Patients with Chronic Diseases

Dissertação de Mestrado

Dissertation presented to the Programa de Pós Graduação em Informática of the Departamento de Informática, PUC–Rio as partial fulfillment of the requirements for the degree of Mestre em Informática

Advisor : Prof. Edward Hermann Heausler
Co–Advisor: Prof. Markus Endler

Rio de Janeiro
October 2013

Pontifícia Universidade Católica
do Rio de Janeiro

Vitor Pinheiro de Almeida

Patient-Buddy-Build: Customized Mobile Monitoring for Patients with Chronic Diseases

Dissertation presented to the Programa de Pós Graduação em Informática of the Departamento de Informática, PUC–Rio as partial fulfillment of the requirements for the degree of Mestre em Informática. Approved by the following commission:

Rio de Janeiro , October 2^{nd}, 2013

Vitor Pinheiro de Almeida

Graduated in Computer Engineering from Pontifícia Universidade Católica do Rio de Janeiro (DI/PUC-Rio) in 2010.

Bibliographic data

Almeida, Vitor Pinheiro de

Patient-Buddy-Build: Customized Mobile Monitoring for Patients with Chronic Diseases / Vitor Pinheiro de Almeida ; advisor: Edward Hermann Heausler; co–advisor: Markus Endler. — 2013.
201 f. : il. ; 30 cm

Dissertação (Mestrado)-Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2013.
Inclui bibliografia

1. Informática – Teses. 2. Saúde Móvel. 3. Monitoramento. 4. Ontologias. 5. Aplicação ciente de contexto. 6. Sistemas Pervasivos. 7. Representação de conhecimento. I. Heausler, Edward Hermann. II. Endler, Markus. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. IV. Título.

CDD: 004

Acknowledgments

I would like to dedicate this dissertation to my brother Rodrigo, my mother Alcione and my dad Vitor. To all my family that gave me unlimited support during all the develop of this work. Without them, I would never be able to finish this work.

I would like to thank my advisor Professor Edward Hermann for the excellent guidance, for the patience to understand all my questions and difficulties. And especially for trusting in me and for having accepted to work with the theme of this dissertation.

I would like to thank my co-advisor Professor Markus Endler for the opportunity to work in the Mobile Health Net project, which is the project that this dissertation is inserted. The opportunity to work with the health area came form this invite made by Professor Markus Endler. Thank you very much for accept me as your student, for the patience of teaching me and for guiding me through every step of my learning.

I would like to thank Bruno Azevedo, who supported me in this work and was always available to help me with any subjects related to the health sector. Bruno‘s contribution was valuable for increasing the applicability of this work with the healthcare area.

I would like to thank my girlfriend Raissa and my friend Renato for helping me overcome difficulties during the development of this research.

I would like to thank my friend Doctor Fabio Gandour that always supported me and encouraged me to follow my dreams and specially for believing in me.

I would like to thank CAPES for the grant of a scholarship and for the support.

I would like to thank God for the opportunity of being able of finishing this master.

Abstract

Almeida, Vitor Pinheiro de; Heausler, Edward Hermann(advisor); Endler, Markus(co-advisor). **Patient-Buddy-Build: Customized Mobile Monitoring for Patients with Chronic Diseases**. Rio de Janeiro, 2013. 201p. MSc. Dissertation — Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

This thesis consists of the development of a tool for generating mobile applications that enables a customized form of remote monitoring of patients with chronic diseases. The customization is based on parameters and formal descriptions of patient preferences, the type of chronic disease, monitoring procedure required by the doctor, prescribed medication and information about the context (i.e. environment) of the patient, where the later is to be obtained from sensors. Bases on this data, the system will determine which information are more relevant to be acquired from the patient through questionnaires and sensors embedded or connected to the smart phone. Relevant information are information that best helps to identify possible changes in the monitoring process of a patient. This set of information will be sent by the mobile application to the responsible physician. The medical treatment and the kind of chronic disease will define the set of information to be collected. It should be stressed that the goal is not to support automatic diagnosis, but only to provide means for physicians to obtain updated information about their patients, so as to allow remote monitoring of patients.

Keywords

Mobile Health; Patient Monitoring; Ontologies; Context-aware application; Knowledge Representation; Pervasive Systems;

Resumo

Almeida, Vitor Pinheiro de; Heausler, Edward Hermann; Endler, Markus. **Patient-Buddy-Build: Acompanhamento remoto móvel customizável de Pacientes com Doenças Crônicas**. Rio de Janeiro, 2013. 201p. Dissertação de Mestrado — Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Este trabalho consiste do desenvolvimento de uma ferramenta para a geração de aplicativos móveis, que possibilita um monitoramento customizado, para o acompanhamento à distância de pacientes com doenças crônicas. A customização ocorre a partir de parâmetros e descrições formais, tais como: preferências do paciente, tipo da doença crônica, processo de acompanhamento desejado pelo seu médico, medicação prescrita e dados sobre o contexto (o entorno) do paciente, estes últimos obtidos de sensores. Com base nestes dados, o sistema irá determinar quais informações são mais relevantes para serem adquiridas do paciente através de questionários ou de sensores disponíveis no dispositivo móvel. Informações relevantes são informações que melhor ajudam a identificar possíveis alterações no processo de monitoramento de um paciente. Estas informações serão enviadas pelo dispositivo móvel, juntamente com os dados dos sensores, para o médico responsável. O processo de acompanhamento médico e a natureza da doença crônica definirão o conjunto de informações que serão coletadas. É importante ressaltar que o objetivo não é realizar diagnósticos, mas sim, prover informações atualizadas aos médicos sobre os seu pacientes, possibilitando assim, realizar um acompanhamento preventivo à distância.

Palavras–chave

Saúde Móvel; Monitoramento; Ontologias; Aplicação ciente de contexto; Sistemas Pervasivos; Representação de conhecimento;

Contents

List of Figures

"Stay Hungry. Stay Foolish."

Steve Jobs, *The Whole Earth Catalog.*

1 Introduction

From the 90's until now, we have witnessed a huge growth in the development of technologies for cellular mobile communication. The popularization of these technologies has enabled access to remote information anytime and anywhere, opening a wide range of new applications and services to users.

The number of mobile devices like smart phones grows increasingly in Brazil. According to data released by IDC, a consultancy firm specialized in technology and telecommunications market, approximately 15,4 million units of these devices have been sold in Brazil in 2012 (40).

In parallel, the coverage of mobile network also increases continuously; only at the first quarter of 2013, Brazil had 96,5 million broadband connections, an increase of 37% compared to March 2012. According to a survey of the Brazilian Telecommunications Association (Telebrasil), 26 million new access points have been activated in the past 12 months (10). It is expected that in the near future even low income citizens will have smart phones and access to mobile broadband in cities, as well as in many rural areas.

Hence, in this scenario there is an increasing urge of mobile application development for every field, including health care. Mobile Health characterizes the practice of medicine and healthcare through mobile devices (33). This area grows every day, and more and more hospitals and healthcare professionals are adopting to mobile technology.

Health care applications for mobile phones and mobile devices are expanding and changing the way of how and where medical care is done. Actively engaging patients and health professionals using sensor-rich mobile devices can help to monitor, prevent and treat diseases (23). Furthermore, patients that have difficulties to go to a hospital or their doctors, either because they live far away or do not have sufficient funds, may benefit from a more convenient, cheaper and yet effective medical care.

This thesis addresses the monitoring of patients with chronic diseases mainly because of the following (2):

- A characteristic of an ageing population.
- It typically involves high cost treatment.

– The affected set of people may include high risk patients.

Thus, the remote monitoring of these patients can help to prevent their health conditions from worsening and avoid individual health crises. (46).

1.1 Objective

The goal of this thesis is to develop Patient-Buddy-Build (PBB)(31, 32), a framework for generating mobile applications enabling a customized form of remote monitoring patients with chronic diseases.

In order to make possible the development of such an application, first there are some requirements that need to be satisfied and which are also the focus of this thesis:

– Representation of medical knowledge and of context information using ontologies (14).
– Automatic generation of the set of information to be monitored from the patient by using a knowledge base that contains information about the patient, available mobile sensor data, the disease and the monitoring process. By automatic generation, we mean to create an automatic procedure (a simulation of a diagnosis) that helps the system to choose which set of information is more relevant to acquire from the patient depending on the situation that he/she is in.

The purpose is not to develop a complete and readily usable mobile application, but to use it as a proof of concept of the purposed model for describing monitoring process from formal descriptions (including patient and doctor preferences, symptoms of disease, treatment method, etc) and to identify benefits and limitations of the approach.

As a prove of concept of our purposed representation for context information, we develop an example application using the PBB framework to remote monitor patients with Atrial Fibrillation[1].

[1]Atrial Fibrillation description: `http://www.webmd.com/heart-disease/atrial-fibrillation/heart-disease-atrial-fibrillation-basics`

1.2 General idea of this work

The PBB framework aims at the generation of context-aware mobile client-server applications, where each mobile application is customized and than used for monitoring a patient.

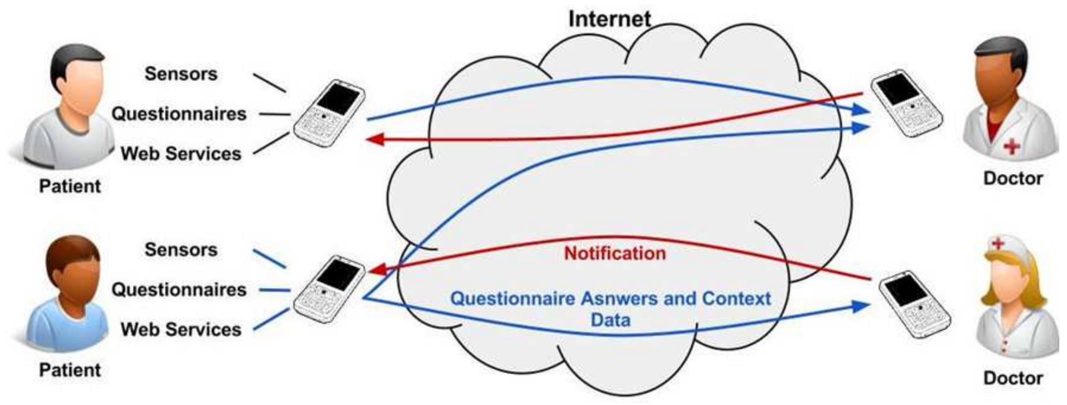


Figure 1.1: The scenario that PBB was designed for.

Figure 1.1 illustrates the three ways that a information can be acquired from a patient:

1. Using sensors attached to the patient or carried by him.

2. From web services. (e.g.: A weather forecast service, if the doctor needs to monitor the weather conditions at the patient location.)

3. From mobile application questionnaires about the patient's health condition (generated automatically) and answered by the patient.

Sensors attached to the patient will be used to monitor a variety of health conditions. For example, using a sensor attached to the patient chest, it is possible to monitor heart rate, breath rate, body temperature and the electrocardiogram signal.

Web services are another source of information that may be necessary for identifying the patient's environment context. In this case, PBB will communicate with another software designed to support a machine-to-machine interaction and provide some services over a the internet. As mentioned before, the doctor could be interested in the weather conditions at the patient's location. For example, asthmatic patients are more susceptible to sudden climate change. Their breath capacity can get worse if the environment temperature suddenly decreases.

The questionnaires of the mobile application are intended for acquisition of information about the user's health state via user input. Every time that a patient is not feeling well and he/she wants to notify his/her physician about it, the patient will press a button and a questionnaire will be generated according to the patient's context.

The idea is that the questionnaire can be answered in a quick and practical way, without excessive input text and prevalence of multiple choice answers. For example, some questions can be used to check if the patient has some difficulties to sleep or if he/she is feeling tired more often. All information collected in any of these three possible ways (sensors, web services and questionnaires) are sent to and persisted in the server.

The main goal of the patient mobile application is to collect data from the patient using sensors, web services and questionnaires and to decide how to use and act upon it. This application will have a self configurable behaviour that will be able to adapt itself depending on the patient situation and activities. For example, the patient mobile application could be configurable to change the frequency in which a certain data is monitored whenever the patient starts doing a physical activity (running or walking).

The patient mobile application is also able of deciding which set of information is relevant to collect from the patient depending on the situation that he/she is in. For example, as soon as the patient reaches a place with an altitude higher than a certain limit, the patient mobile application can automatically configure itself to be able to ask the patient a different set of questions (e.g.: if he is feeling any symptoms related to the altitude). While before the altitude limit, it may not have been relevant to the application to ask about symptoms related to high altitude.

This kind of behaviour helps the doctor in the task of choosing which information to collect from the patient on each moment of his monitoring. The patient mobile application is also capable of taking actions based on the patient's situation. For example, if the patient has an anti-coagulated prescribed medication and he just enters a place where the altitude is higher than a certain limit, the application will detect this pattern and send an alert to the patient alerting him about the dangerous of it. Staying in a high altitude place, can make events like nose bleeding be very easy to happen, and when uncoagulated patients stars to bleed in that kind of altitude, they may not able to stop it.

Before using the PBB framework, it must be first customized. The customization stage is where it is defined all the doctor preferences related to the monitoring of his/her patients. The customization is not made directly by the doctor, it is made by an ontology engineer by editing a set of ontologies that together defines for the PBB framework how the process of monitoring the patient will be done.

The following items are a resume of the possible customizations:

– **Sensors used:** Which sensor will be used for the monitoring process. Maybe it is not necessary to use all the available sensors with a specific patient. Concerning that the mobile device has low resources, it may be important to disable a sensor in order to save battery.

– **Conditions monitored:** Which conditions the doctor wants to be monitored and how each of them will be acquired. The monitored conditions can be any condition (e.g.: heart rate or breath rate, feeling headache) that is possible to be acquired or derived using information from questions, sensors or web services. The patient's activity, for example, can be a condition derived from the accelerometer and the inclination of the patient.

– **Monitoring process:** Which situation is considered critical and the doctor must be notified. Which patient situation the application should record more intensively the electrocardiogram signal. Which symptoms are interested to be monitored in each of the patient's context (e.g.: Altitude symptoms are not interested to be monitored if the patient location's altitude is less than 2400 meters). When the patient should be warned about a specific monitored condition? (e.g.: It is important to warn the patient about the dangerous altitude? Or if he travels, to remember him about the cover area of his health plan?). Which information must be persisted by the system and in which case? (e.g.: During normal monitoring states the doctor only wants to record the electrocardiogram signal in 6/6 hours by 10 seconds. After a physical activity the doctor could want to record one minute of electrocardiogram signal.)

– **Questionnaire:** The doctor could want to ask questions to the patient and choose the answering form of the question. For example, a doctor could prefer using multiply answers to a headache question that must have only one answer. Other doctor could give the patient the possibility of choosing more then one answer. Or maybe the doctor wants that the patient uses free text box to give the patient the possibility of answering whatever he/she likes.

This thesis is being developed within the scope of the MobileHealthNet[2] project (43), a joint effort of the Distributed Systems Lab at UFMA (25) and the Laboratory for Advanced Collaboration at PUC-Rio (24), with support from the University Hospital (HUUFMA) of UFMA (Federal university of Maranhão). In particular, the development of the Patient-Buddy-Build (PBB) is being supported by two units of the HUUFMA: The Assistance Program to Asthmatic Patients (PAPA) and the "Casa da Dor", which specializes in treating patients suffering with chronic pain.

The PBB framework relies on the MobileHealthNet communication middleware which has authentication and cryptographic mechanisms already implemented in it. Therefore issues regarding security will not be discussed in this thesis.

The remainder of the thesis is organized as follows: Chapter 2 gives a theoretical background about important concepts used in this thesis and lists approaches to represent context information that already exists in the literature. In chapter 3 we describes an ontology-based method to represent context information that was used by this thesis and how it was applied to it. Chapter 4, we explain our knowledge base, how it is organized nad how it can be customized to monitor different patients with different diseases and with different monitoring process. Chapter 5 we explain how the PBB framework was implemented and witch technology were used. In chapter 6 we describe an example application developed using the PBB framework. In chapter 7 we compare the modeling approaches listed in the previous chapter with the proposed by this thesis focusing on applying it to the remote monitor of patients with chronic diseases. And finally, in chapter 8, we draw some conclusions about the work developed in this thesis.

[2]Mobile Social Networks for Health Care: `http://www.lsd.ufma.br/~mbhealth/`

2 Theoretical Background

2.1 Context aware applications

There are several definitions regarding context, but one of the most referenced is given as follows (12): “Any information which can be used to characterize the situation of an entity. An entity is a person, a place or an object which is considered relevant for the interaction between a user and an application, including the user and the application”.

When a user enters a new context, it is desirable that the applications on his devices are able to adapt themselves to the new situation and the environment. They should be able to adapt its services to the presence of the new user.

Abowd in (1) proposed the notions of primary context (e.g, localization, identity, activity and time) and of secondary context, where the latter can be deduced from the primary context and may be used for making adaptation decisions at a higher level of abstraction.

Context awareness applications are applications that have the ability of sensing (context information) from the current environment and autonomously performing appropriate adaptations in regard to its optimal operation, general behaviour and user interaction.

Remote mobile monitoring is an example of scenario where context aware applications could be used. The context in which each patient is and the way it is measured a certain information is relevant for a better interpretation of the information collected. For example, measuring the heartbeat of a patient sleeping has a different interpretation compared to when he is walking. Also, monitoring a patient when he is at work may involve different concerns in relation to the monitoring if he or she is at home.

This kind of applications must understand, describe and manage context information, in order to do this a *model for representing context information* is required. The propose and creation of this model is the focus of this thesis, the system to manager the context information (the context aware application) is only implemented to be a prove of concept of the proposed model for representing context information and is not the focus of this thesis.

The model for represent context information encapsulates some requirements needed to implemented context-aware applications, and because of this they reduces the complexity of their implementation and improves their maintainability.

To create a model for represent context information, several requirements must be attended (7, 42, 17). In the next section we listed the main requirements for creating models for represent context information.

2.2 Modeling Requirements

Distributed Composition When developing context-aware applications, they can be applied to different environments and to different types and sources of context information. Data collected from sensors, such as GPS position, can be outdated very quickly and usually have to be interpreted before use. Context data collect from profiles of social networks, in the other hand, have the characteristic to be more static and usually do not need additional interpretation. Thus, the context information model must adapt to the needs of the environment that it is in.

Partial Validation and Reasoning Modeling context information is not a simple task, the complexity of the relations among context information can make the modeling attempts error-prone. There are a variety of types of context information and each of these types have different properties and constrains that must be respected when modeling. Partial validation is a requirement that helps detect errors within the context data model by using consistency verifications. This consistency verifications can be executed in terms of structure and in instance level. Reasoning is desirable to, for example, generate secondary context information from primary(basic) context information. When modeling context information interrelations, implicit knowledge can become explicit by using a reasoning process.

Richness and Quality of Information Due to the heterogeneous nature of the sources of contextual information, richness and quality information must be considered and represented by the context information model. Different sensors can provide information with different precisions/accuracy (e.g., few Celsius degrees of imprecision about the measurement of a temperature). The user of these sensors should be able to choose from which sensor to retrieve context information (42). Sensors can differ on the rate that they send new context information. For example, a user should that wants the most up to date information related to the geographical position of a person, should be able to choose the sensor with the high freshness rate.

Incompleteness and Ambiguity Besides the quality of the information, two or more sensors can provide the same context information with different values (e.g., the weather of a city can be considered to be rainy and sunny at the same time by two different sensors). This ambiguity should be captured by the model and early resolved by using conflict resolution techniques. Another problem is that the context information can be incomplete, especially when derived from sensors. A sensor that counts how many trucks are located in a specific city, can miss some truck.

Level of Formality Level of formality is the capacity that a context information model have to express contextual facts and the relationships among them. The more precise the representation of context information is, more different types of situations and tasks can be represented with it. For example, for the application understand that: "If a patient's body temperature is higher than a certain limit, it is considered that this patient has fever.", each of the terms must be precisely defined by the context model. What is "body temperature" or what does "higher than a certain limit" means. This requirement also describes how well a context model can define a specification for knowledge sharing and reuse (e.g., if the context model is extensible or if it is abstract enough to be used by another application domain).

Applicability to Existing Environments Is the capability that a model to represent context information have to be implemented within different existing infrastructures (44). Independently of the formality level, the reasoning process or the different types of context information that a context model can represent, it must be possible to implement and apply the context model. The infrastructure in which the context model will apply can vary depending on the environment on which the system should be implemented. For example, if the system should be all implemented using mobile devices, the fact that this devices have low resources has to be considered.

2.3 Modeling Approaches for context information

Several authors (7, 19, 42, 44, 47) defined relevant modeling approaches for context information. The first two models are to be described as follows are early approaches:

2.3.1 Key-value context information models

The key-value model is the most earlier context modeling approach among the others described in this thesis. The main characteristic of this model is its simplicity, it uses key-value pairs to describe context information used by context-aware applications. Every context information has a key and an associate value. This characteristic enchants the applicability to existent environment requirement.

– person(name $\rightarrow$ “Silva”)

– person(responsibleDoctor $\rightarrow$ “Menezes”)

There is no notion of ordering or hierarchy that is already expressed from within the model. It is possible to create a property that means order, that associate a context information with a number, for example. But the interpretation of the ordering is expressed in the application level. There is no constructor or native property in the key-value model that can indicate a hierarchy order. We can see this context model as a flat information model, because all the context information are either a key or a value. In terms of formalization, the determination of if a patient has fever or not (defined if the patient's body temperature is higher than a certain limit) may be possible. In the other hand, formalizing the fact that an application should record five minutes of electrocardiogram if and only if the patient just stopped doing some physical activity and he/she is not feeling any atrial fibrillation symptom, then it might be better use another context modeling approach.

2.3.2 Markup based context information models

Markup based context information models use a variety of markup languages (e.g., XML), and they have a native hierarchical data structure. The hierarchical relations and attributes are represented using markup tags. Markup tags can be represented recursively by using another markup tag. This context information model approach was the first to use RDF and include elementary constraints and relationships between types. For example, when using a XML serialization, it is possible to create XML Schemas, that can validate (in a schematic level) the content of the markup tags (e.g., the model can verify if the content of a markup tag is in a numerical value or a text value format).

Most of the markup based context information models are defined as extensions to the Composite Capabilities / Preference Profile (CC/PP) (45), which is a W3C standard for description of mobile devices. With this standard it is possible to express, for example, the hardware platform where a specific software is executing or define characteristics of a specific application, such as a browser. This information can be useful, for example, to know the appropriate data format to be sent to a specific client. It is also possible to extend the CC/PP, an example is the markup based approach (named: *CC/PP Context Extension*) created by Indulska (22). Indulska defined a set of CC/PP concepts and attributes that allow to express additional context information types and relationships between them.

The following approaches to context modeling are more recent and are considered most prominent approaches (7, 44, 42).

2.3.3
Graphic/Object-role based models of context information

The object-role based model approach is a fact-based approach. With the object-role based models, it is possible not only to categorize context information in a hierarchical form, but also to identify the roles that each of them play in the system. By doing so, the model becomes more expressive and pieces of the application code could now be used for iterating thorough objects that not only are of the same hierarchy level but also that plays the same role within the system.

The newer approaches of object-role based models focuses on the Object-Role modeling (ORM) (15). ORM is represented in a graphically way, where each entities are represented as ellipses containing a name, and an optional reference scheme describing the representation of instances of the entity type. Entity types that are simple value types, do not require a reference scheme and are shown as dashed ellipses.

Fact types are shown as sequences boxes, each box attached to an entity type. The fact type are described with labels, that contains text written with natural language. This labels are designed to be written by humans, thus the semantic of the model is directed to humans, not yet to machines. We made a small example that illustrates how to express, using ORM, that a question can be made to a person at some time during some activity.

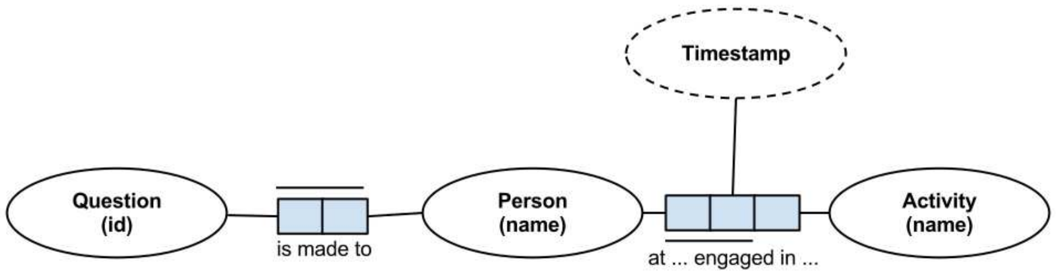


Figure 2.1: Using ORM to express that a question can be made to a person at some time during some activity.

In order to do so, figure 2.1 represents two fact types:

- The fact *"can be made to"*, that relate the entity **Question** with the entity **Person**.
- The fact *"at .. engaged in"*, it associates the entity **Person** to the **Time** that the person started a specific activity.

As the other context information models, the ORM can be extended too. In order to give an example of which kind of extension can be created, we will resume an initiative of Henricksen (16). Henricksen extends the ORM to enchant the richness and quality indication requirement of context information models. The extension created by Henricksen have four different context information types:

- **Sensed**: Context information acquired from sensors that are usually frequently changing and can suffer from problems of inaccuracy. For example, the heart rate measure, a GPS position or any measure taken from a eletrocardiogram.
- **Static**: Context information that do not change during the time. In the remote monitoring scenario, could be the chronic disease of a patient.
- **Profiled**: These are context information that are supplied by the user, its change frequency rate is characterized as slow because it is not common to expect users to change their information, for example, very hour. This type of context information could be unknown, incomplete or out of date.
- **Derived**: Context information derived from one or a set of primary context information. The derived context information type could inherit error from the primary context information (used to create the derived context), or even inherit errors from the process that was used to create the derived context.

The differentiation of the four classes of context information is useful, for example, on the instance level for context management purposes. Each of the four classes of context information requires different treatment, for example in terms of conflict detection, update management and trustiness.

Henricksen extended model also defined when a fact type is *ordinary* (do not have possibility of imperfection) or *alternative* (have possibility of imperfection). *Alternative* facts (e.g., They can be ambiguous or imprecise) and are represented as possible true, using a three valued logic (16).

Regarding the capabilities of supporting reasoning using this extension, the formality of ORM supports the evaluation of simple assertions. For example, using the "**is doing activity**" fact type, it is possible to instantiate it in two different forms:

First form:

1. Silva ***is doing activity*** Running

Second form:

1. Silva ***is doing activity*** Running
2. Silva ***is doing activity*** Walking

The "**is doing activity**" fact type addresses a sensed context information, that can be determined by more than one sensor. Thus, it is represented as an *alternative* fact because this information have the possibility, for example, to be ambiguous. Two different sensors can provide different information. The first form do not represents any possible ambiguous information. However, the second form, the assertion expressed by the two facts represents an ambiguous information.

The main focus of the object-role modeling approach is on the structure level (42). The extension of ORM presented above is an example of how the object-role model approach can be used to describe the structure of contextual knowledge.

2.3.4 Spatial context information models

Spatial context information models gives more importance the spatial context and most of them are fact-based models. The spatial location can be used, for example, to determine *the resources that are nearby*, *if there is someone near*, *if a person is located in his/her work place* (12, 36, 37). This information are very common among context-aware applications that are focused in the human. Spatial information is not only the location of a person in the real world, for example, a disease can be seen as a “place”, where a person have different properties when related to it. That way, a place can be used as a metaphor for non physical context information.

There are two types of spatial context information models:

Symbolic coordinates Symbolic coordinates positioning systems are systems that do not need to use explicit positioning (e.g., latitude and longitude information). They uses spatial identifiers to organize real world areas (e.g., by only using country names) or even to spatially separate different contexts that an entity may be related with (e.g, when a person is depressed, he/she can be related to different properties) (6).

An example of a context-aware application using symbolic coordinates would be a smart system to turn off air conditioners. For example, in an university, there is a lot of class rooms equipped with air conditioners. It is very common that people forget to turn the air conditioner off when leaving the room. It is possible to develop a system that will use cameras and sensors that can recognize a change in the room environment if a person enter a room. This system would not need precise location information like latitude and longitude. For example, in order to infer that there is no people in a room, a camera could be pointing to the main door of the room, and when someone enters the room the system increments a counter. With this information, whenever there is no people in the room, the air conditioner could be turned off. The location information needed are only the room number or name in order to the system identify which rooms have the air conditioner on or off at a specific moment of a time.

Another example of application that uses this positioning system is the **Stick E Notes** application that is described within this thesis in section 2.4.2.

Geometric coordinates Geometric coordinates positioning systems are systems that represent points or areas in a metric space, because they need the precise location on an object. They provide the system with the exact point in the space where an object is located in (e.g., providing the latitude and longitude information). One example of this type of coordinate system are systems that manages a group of trucks.

For example, a company that delivers food all over the state needs a system to manage and assist communicate with the trucks, using context information for their benefit. The idea is that the system could use information provided by sensors installed within the trucks to help them with the job of delivering the food. For example, based on a location and on the information that it is raining provided by a specific truck, the system could warn the nearest trucks to avoid pass through that location. Another information that trucks can exchange is when they detects that a specific road is blocked (e.g., because of an car accident). By sending the exact position of the truck, it is possible for all other trucks to know where is the accident and to avoid that road.

2.3.5 Object-oriented context information models

The object-oriented modeling approach employ the advantages of encapsulation and reusability of any object-oriented approach to better attend the modeling requirements for represent context information (42).

The object oriented paradigm is capable of grouping together data and tasks in entities know as objects (this is called encapsulation). With this technique, the system can categorize data and identify similar data. For example, it is possible to create an entity(class) that represents all the data that is monitored from a patient.

By doing so, the system will interpret all instances (objects) of that entity(class), as a monitored data from the patient. New types of contextual information (classes) as well as new updated instances (objects) may be handled in the system in a distributed way. The entity (represented as classes) encapsulates the details of context processing from the other entities of the system. The access of contextual information is provided through interfaces, provided by the entities.

An example of this approach is presented by Bouzy in the article named “Using the object oriented paradigm to model context in Computer Go” (9). Bouzy assumes that contextual knowledge in complex domains aims at organizing knowledge for efficient use. Based on that, Bouzy proposed an object-oriented paradigm mechanisms to represent and organize context information, by representing contextual knowledge using classes. The context information was divided in four classes: temporal, goal, spatial and global.

An example of class created by Bouzy was the temporal context information class. It was a more general class to identify relevant temporal aspects of the game Go (The beginning of the game, the middle and the end of the game). By categorizing these temporal aspects, Bouzy creates classes that represents the beginning, middle and the end of the game. Using the division, the system can better behave according to each part of the game. For example, a common strategy in the game Go, is to fill the corners at the beginning. If the system is able to detect that the game is at the beginning, it can better behave according to a strategy that behaves differently at the beginning.

Bouzy used the object-oriented paradigm to organize the contextual information within the system, and justified it with its inheritance and reutilization capabilities. Using this technique, the programmer avoids having to re-specify properties of general classes (e.g., the temporal class proposed by Bouzy) when creating specialized classes.

There are other solutions that used the object-oriented paradigm, aiming to provide organization for contextual knowledge and processing by using the benefits of this approach: encapsulation, reusability and scalability. For instance, we have the MoCA middleware (35).

MoCA's Context Service Architecture is based on an object-oriented model paradigm with an abstraction level for representing context information. This approach have been primarily driven by the requirement of distributing composition to being able to manage greater varieties of contextual information while maintaining scalability.

2.3.6 Ontology based models of context information

In order to define what are the ontology-based context information models, first we must define ontology. Ontology, in philosophy, is the science that studies the subject of existence. It is basically a way of understanding entities and group of entities called classes. In informatics, we call ontologies a common understanding or classification of entities, which can be abstract information or physical objects (real world objects). It is a model that represents concepts and relationships of a domain.

There are several reasons for creating context models based on ontology:

– *Knowledge reuse*: Since ontologies can define very abstract concepts, when expanding the system, it is possible to reuse them on different domains. It is not necessary to start from scratch to compose a large-scale context ontology.

– *Knowledge Sharing*: As we mentioned before, ontologies refers to a common understanding of some domain. When using it in a context management system, it enables each part of the context management system to have a common understanding about context, while interacting with one another. Regarding the distribution composition requirement this characteristic is useful.

– *Logic Inference*: There are various types of computational logic and reasoning mechanisms to deduce complex context information from primary context. Context-aware systems can exploit and use ontologies to solve inconsistent context knowledge due to imperfect sensing (context information acquired from sensors) or even inconsistencies within the created model to represent what the application knows about context structure representation.

Ontologies can be written using different standards. The Resource Description Framework (RDF) is an example of a standard that provides data model specification to facilitate data merging even if the underlying schemas are different. Another standard is the Web Ontology Language (OWL) that enables the definition of domain ontologies and sharing of domain vocabularies.

From a formal point of view, OWL can be seen to be equivalent to description logic (DL) (3). DL is a family of logic-based knowledge representation language that can be used to represent the terminological knowledge of an application domain in a structured way.

Ontology-based models of context information exploit the representation and reasoning capability of this logic for multiple purposes:

1. *Level of formality*: Used to describe complex context data that cannot be represented by languages like CC/PP(22). Complex context data can be represented by structured OWL-DL (a subset of the OWL language) expressions. This data typically include complex expressions about a user's preferences or activities. In order to illustrate this complex context data definition capability, we can consider the following example: "Menezes' smart phone must be configured to turn the vibrate mode on whenever he is attending a medical meeting at his hospital." This example could be expressed using description logic with the following expression: $\mathbf{SetMenezesPhoneToVibrate} \sqsubseteq \mathbf{Activity} \sqcap \geq \mathbf{2\ hasActor} \sqcap \forall \mathbf{actor.MedicalStaff} \sqcap \mathbf{hasLocation.Hospital}$

2. *Reasoning tools*: Can be used to check the consistency of the set of relationships that describes a context scenario. It also can be used to derive implicit information from explicit information. For example, using description logic, we can create a rule that says that every woman who has a child is a mother too.

   $\mathbf{Mother} \sqsubseteq \mathbf{Woman} \sqcap \geq \mathbf{1hasChild}$

   If the context-aware system capture the fact that, for example, Regina is a woman that has a child, the ontology reasoning system will infer that she is also a mother. The statement "Regina is a mother" is implicitly defined, and is explicit generated when the reasoning executes.

2.4 Related Work

In this thesis we develop a context aware application but with the only purpose to serve as a prove of concept to our purposed model for representing context information. Therefore, in this section, we present existing models for representing context information aimed to provide tools for building context aware applications. Their descriptions in this thesis is mostly focusing on the approach they use to deal with the representation of context information.

2.4.1 CC/PP based context management architecture

This work (22) consists in a three layer architecture that has the ability to dynamically reconfigure itself in order to adapt to new requirements or a changing environment. The context representation model of this work is a markup-based. They extended the basic CC/PP vocabulary by a number of components in order to describe a wide range of context information. Such components include network interfaces of devices, quality of service, location, disconnection status and application requirements. Composite Capabilities/-Preference Profiles (CC/PP) is a W3C proposed (markup-based) standard for describing device capabilities and preferences with focus on wireless devices such as PDAs and mobile phones.

The architecture is structured into three autonomous layers:

– Context-aware application layer: By using a subscriber language, named Elvin, the context-aware application can subscribe to a set of events of interest that are provided from context sources that are capable to process context information from sensors.

– Awareness module layer: This module is where the context sources are located. Awareness module is a specialized module that provides a particular functionality, they transform information from sensors into a format that can be understood by the context manager layer.

– Context manager layer: It manages the persistent repository (database) of context information, receives context updates from the awareness module layer, detects context changes and provides notifications about the change to all interested clients.

All the context information gathered from sensors are processed and converted to a RDF model and persisted (serialized) in XML format in a centralized form (in the context manager layer).

2.4.2 The Stick-e Note Architecture

The Stick-e Note Architecture(29) is a proposed solution for capturing, representing and processing contextual data. It provides a framework that can be used by application to adapt its user-interface or perform some action based on the current context.

The approach uses spatial information as main context, and proposes the idea of attaching information to objects within the user's physical environment. Depending on spatial relations, for example, if a person meets either Glenn or Helena, a stick-e note may be triggered with specific information that belongs to this context.

The architecture represents the context information using an object-role approach and, for each entity, there are a variety of contextual states that are attached to them (stick-e notes). Depending on the location that the user is, a different stick-e note can be attached to the entity.

A centralized class (designed using the singleton object oriented template) receives context information from all operational devices attached to it to provide data to potential clients. Potential clients are classes from the object-oriented paradigm that may not actually require access to specific types of context information.

Within the potential clients, exists a controller element that is responsible for continual trigger-checking process of new context information (notes) and the routing of triggered notes to displays (the screen of the user's device). The triggering condition of a stick-e note consists in a singular or compound context.

2.4.3 Context-aware eTourism application

The context-aware eTourism application (27) is based on a hybrid approach named Context Distribution and Reasoning(ConDoR) that aims to combine the advantages of object-oriented and ontology-based modeling. The ConDoR model consists of two sub-models: the object-oriented Context Acquisition and Distribution Data Model (CADDM) and the ontology-based Context Reasoning Data Model (CRDM).

The CADDM gives the architecture the capability of understanding context information from a variety of categories. It defines five general categories: Location, Environmental, Activity, Instrumental and Social context. These categories help distribute and organize the acquired context information from the sensors. They help accommodate different application purposes, and can be extended by adding other appropriate categories or by refining the existent ones using advantages from the object-oriented paradigm (i.e., specialization for refining existent categories and the use of well defined interfaces to create new ones.).

The CRDM aims to highlight semantic relations among ontological classes, while the CADDM represents mainly syntactical relations. Each of the contexts items which describes the context of an entity according to syntactical rules defined in the CADDM model is here translated into an ontology. For example, the fact "A *Place isPopulatedBy* some *Persons*" is expressed as a OWL statement in order to be analysed and processed by inference engines. The CRDM persists all the context information with an OWL-DL ontology and description logic reasoners can be used in order to determine concept satisfiability, class subsumption, consistence and instance checking.

3
Contextualized Ontologies and how it was applied on PBB

The main reason for this thesis to choose the use of ontologies, is to be able to take advantage and use the algebra of contextualized ontologies. Contextualized ontologies are proposed and defined on the work entitled: "Specifying Ubiquitous Systems through the Algebra of Contextualized Ontologies" (11), which describes a formal framework to represent context information.

Before defining the algebra of contextualized ontologies it is important to define the terms: *component* and *vocabulary*. A *vocabulary* is a set of concepts and relations used to describe a certain domain (area of concern).

The difference between the terms *ontology* and *vocabulary* is that the term *ontology* does not necessarily contain only concepts and relations, but it can contain a more formal description about them. For example, ontologies can use a logic-based knowledge representation language (e.g.: Description Logic) to achieve a higher formalism. While in the other hand, the term *vocabulary* is only used when such strict formalism is not necessarily used or only in a loose sense (e.g.: by using a schema validation).

And finally, a *component* is a term that we will use in this thesis to refer to concepts and/or relations of an ontology or vocabulary.

The algebra of contextualized ontologies is based on three central principles, quoting (11):

– *Homogeneous* description: The adoption of a unique mechanism for representing knowledge. The usage of ontologies for representing both entities and contexts. This enhances the flexibility of the framework avoiding to determine a priori the role of an ontology: an ontology may represent an entity, a context or even both an entity and a context.

– *Independent description*: Any entity or context is independent with its encapsulated attributes. The framework puts the focus on the relationship among the components of a systems and not on the components themselves. In this way, the internal constitution of an entity is hidden, and descriptions are built in a modular and reusable way (28).

– *Net of relationships*: Consists of semantic links between entities and contexts. These links define which ontologies represent the entities (the domain of the link) and which represent the context (the codomain of the link). It is also possible for an entity to have several contexts (represented as several links with the same domain), or for a context to contextualize several entities (represented as several links with the same codomain). These structures can be arbitrary extended and be composed in a associative way. They can form a net of entities and context that all together defines a complex situation.

3.1 Diagrams of Device, Entity and Ontologies

In order to specify the Patient-Buddy-Build framework we used a layered division of the algebra of contextualized ontologies. This layered division facilitates the process of system specification where context information must be taken into account (i.e., context-aware applications). This method consists in the usage of three kinds of diagrams: diagram of devices, entities and ontologies. The diagrams helps dealing with different levels of abstractions at the same time.

(i) **Diagram of Devices:** The purpose of this diagram is to define all the computational devices that compose the system and the kind of information that is exchanged among them. We can see this diagram as a graph, where each node represents a computational device and the links among the nodes represent the type of information exchanged between devices.

(ii) **Diagram of Entities:** After defining the devices that compose the system, it is important to specify each entity that will interact with each device. Entities can be people or even softwares that will interact with any device. For example, a person that must be monitored differently when located at a region with high altitude. This person will interact with a device (e.g., a smart phone) and generate some prodecure executed by the device. Each device of the diagram of devices is related to a diagram of entities. In this diagram there is no detailed information about how each of the entities are connected.

(iii) **Diagram of Ontologies:** Each entity of the diagram of entities will be detailed and described by an ontology. The links/connections among the entities will be specified in this diagram too, in order to be possible the alignment of two ontologies that represent entities.

3.1.1 Diagram of Devices

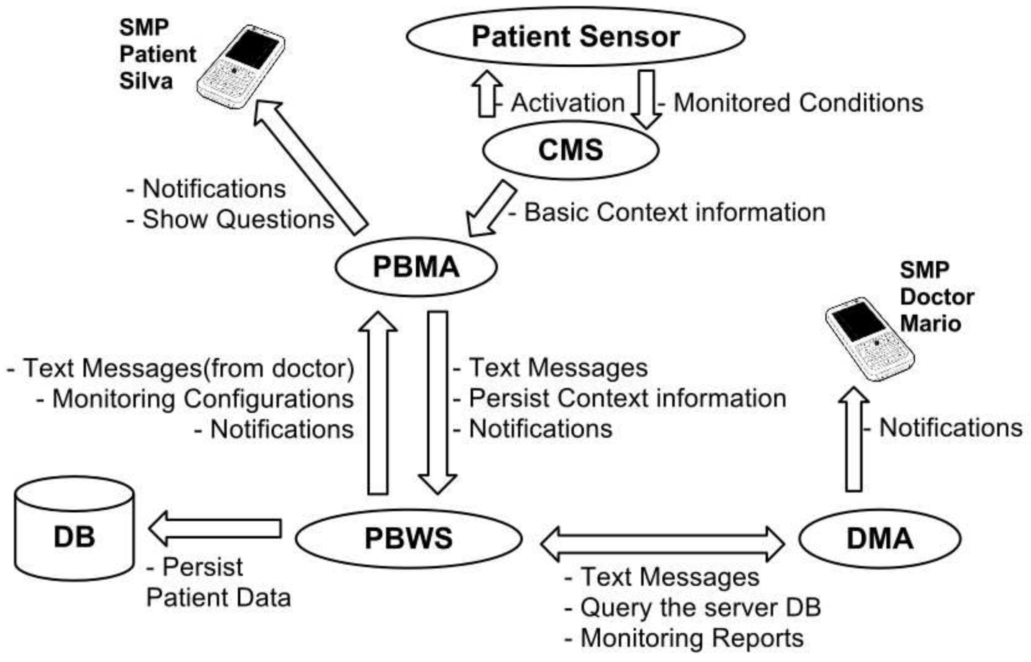


Figure 3.1: Diagram of Devices: Remote monitoring patient Silva.

Figure 3.1 composes the diagram of devices of the scenario described in section 3.3. It shows the type of information that PBWS, PBMA, CMS and DMA interchange in order to monitor patient Silva.

– CMS (*Context Management Service*): In PBB the CMS is responsible for collecting information from the sensors attached to the patient. The set of sensors that will be used for monitoring a patient can be chosen. The doctor can choose which sensor to use, CMS is the service that activates the set of sensors needed to monitor the patient.

– PBMA (*Patient-Buddy Mobile Application*): The mobile application responsible for monitoring Silva. This application changes the monitoring configuration depending on the context information and the situation that Silva is in. For example, the frequency that a certain information is sent by the mobile phone to be persisted in the web server is configurable. This application is also responsible for sending alerts to the patient and SMS messages to the people responsible for him.

– DMA (*Doctor Mobile Application*): Is an mobile application from which the doctor can request reports on the monitoring data of his patients. He can also receive text messages from patients and send messages to the patients.

– PBWS (*Patient-Buddy Web Server*): Is responsible for persisting the monitored context data from the patient's mobile device. PBWS knows all the monitoring process of all the patients. The monitoring process contains information related to: (a) In what frequency the context data will be monitored from the patient. (b) Depending on the situation that the patient is, what set of context data will be monitored. (c) If the mobile device must send a SMS message to the doctor warning him/her about his/her patient's health situation. The PBWS is responsible for sending a configuration file that contains information about how the Patient-Buddy Mobile Application (PBMA) will behaviour depending on the situation that the patient is in. This configuration file is called a monitoring configuration. The monitoring process has information about how the PBMA will monitor the patient during all situations defined by the doctor. A monitoring configuration file only tells to the PBMA how it must monitor the patient during only one situation. The monitoring configuration file is send to the PBMA every time the patient changes his/her situation (e.g., starts doing a physical activity or arrived in a place that is considered to have a high altitude).

Before we describe the other two diagrams, first we must define what is a contextualized ontology. After this, we will describe the scenario for which this application was designed to and then describe the diagrams of entities and ontologies.

3.2 Defining Contextualized Ontologies

Contextualized ontologies are described by (11) as structures that persist a link between two ontologies. The source of the link is the entity and the target is the context. In the remote monitoring scenario, we can think about the entity as the patient. The context can be thought as the monitoring process to which the entity (i.e., patient) can be related to. The representation of the patient ontology(entity patient described in an ontology), monitoring process ontology(context monitoring process described in an ontology) is self contained. This self contained characteristic gives a high degree of mobility, however the links between the entity and the context must be made explicitly defined. The link shows to the system how the entity can be viewed from within a specific context.

The following definitions are given and are described as they are shown in (11) as following:

Definition 1 (Ontology structure): An ontology structure is a tuple (C,R,H^C, rel,A). The components, in the same order that appear in the tuple, are: Concepts, Relations, which are disjoint sets. $H^C \subseteq C \times C$ is a hierarchy of concepts - a taxonomic relation. rel:R $\rightarrow 2^{C \times C}$ is a function that relates concepts non-taxonomically. Axioms specify other properties of concepts and relations. We assume that H^C is a partial order. By $(x_1, x_0) \in H^C$ we mean that x_1 is a subconcept of x_0.

Definition 2 (Link Between Ontology Structures): A link between ontology structures is a triple (f,g,h): O $\rightarrow O'$ where O = (C,R,H^C, rel,A) and $O' = (C'$,R',$H^{C'}$, rel',A') are ontology structures, f : C $\rightarrow C'$, g : R $\rightarrow R'$ are functions such that (i) if $(c_1, c_2) \in H^C$ then (f(c_1), f(c_2)) $\in H^{C'}$, for $C_1, C_2 \in$ C; (ii) if(c_1, c_2) $\in$ rel(r) then (f(c_1), f(c_2)) $\in rel'$(g(r)), for $c_1, c_2 \in$ C and r $\in$ R; and h : A $\rightarrow A'$ exists if and only if (iii) for all a $\in$ A there exists $s' \in$ Th(A'), where s' is the sentence that results from the translation of a to the vocabulary of O' by f and g.

By condition (i), links preserve hierarchy of concepts. By (ii), links preserve relations. By (iii), axioms of the domain ontology structure, when properly translated to the vocabulary of the codomain ontology structure, hold for the codomain. The process of translation is a canonical process over the structure of the sentence, The authors in (11) denotes by $trans_{fg}$(a) the translation of an axiom a by f and g. By Th(A') we mean the set of sentences that are provable from axioms A'. From (iii), we see that a link between ontology structures induces a mapping between theories:

Let O = (C,R,H^C, rel, A) be an ontology structure for Ω, and (f,g,h) : O $\rightarrow O'$ be a link between ontology structures, where $O' = (C'$,R',$H^{C'}$, rel',A'). Then, by (iii) of definition 2, if a $\in$ A then $A' \vdash tranfs_{fg}(a)$. O' induces an ontology Ω' whose axioms are A' and translation of the axiomatization of H'. Then there is a syntactical morphism between Ω and Ω': for all a $\in \Omega$, $\Omega' \vdash tranfs_{fg}(a)$.

Definition 3 (Contextualized Ontologies): A contextualized ontology is a triple (e,l,c), also represented by e $\rightarrow$ c, where l is a link between the ontologies e(domain ontology) and c, codomain ontology.

3.2.1 Operations

This section describes operations that can be made with contextualized ontologies, in order to compose and decompose them in several different ways.

In order to give a more practical example to explain what capabilities this operations can provide to the system, figure 3.2 shows how the heart rate can be seen under the point of view of different contexts. When the heart rate is interpreted under the disease point of view, it acquires properties like: description as a medical data and why it is related to the disease. In the other hand, when the heart rate is seen under the point of view of the monitoring process, it acquire properties like: the frequency that the application will record this information, how critical is this symptom (when the heart rate is too high, the monitoring process must change to a critical state?).

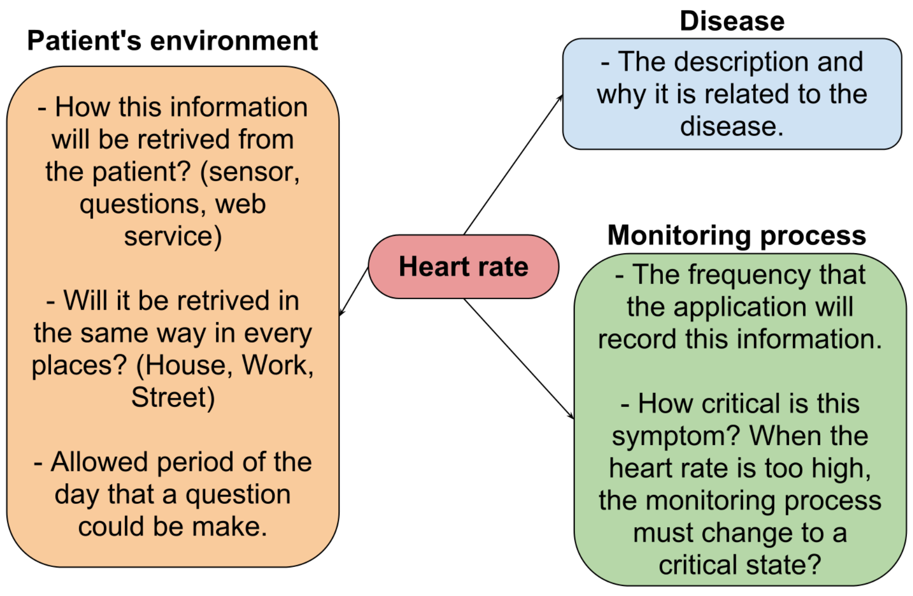

Patient's environment
- How this information will be retrived from the patient? (sensor, questions, web service)
- Will it be retrived in the same way in every places? (House, Work, Street)
- Allowed period of the day that a question could be make.
Heart rate
Disease
- The description and why it is related to the disease.
Monitoring process
- The frequency that the application will record this information.
- How critical is this symptom? When the heart rate is too high, the monitoring process must change to a critical state?

Figure 3.2: Context data representation

In order to give the application this capability, of seen a concept (heart rate) under the point of view of different contexts, we use the algebra of contextualized ontologies. By using it, the heart rate is seen as an entity and represented with a ontology. And each of the contexts that this entity can be contextualized is represented as an ontology.

The links between ontologies mentioned earlier in this chapter, are the mappings between the entity ontology and the context ontology. This links shows to the system how the ontologies will be aligned.

In this section, we show operations with contextualized ontologies that integrate/align both context and entity ontologies. The manipulation of information made by the operations in order to compose and decompose the contextualized ontologies produces a result that is coherent to be used by computer systems. It is called contextualized ontologies because the ontologies only make sense when seen as a pair (Entity $\rightarrow$ Context). The heart rate, in figure 3.2, when it is not seen under any context, it is only an abstract concept.

In the moment that it is contextualized it gain an interpretation that can be used by the application.

The defined operations in (11) are: **Alignment**, **Context Integration**, **Collapsed Union**, **Entity Integration** and **Relative Intersection**, quoting (11):

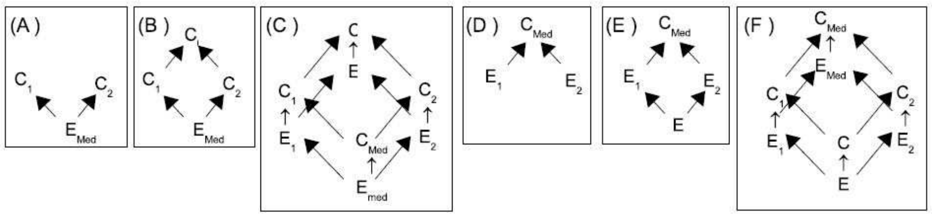


Figure 3.3: (A) Alignment (B) Context Integration (C) Collapsed Union (D) Coalignment (E) Entity Integration (F) Relative Intersection. Figure taken from (11)

Alignment (Figure 3.3-A): A situation where an entity has more than one context is an alignment: $C_1 \leftarrow E_{Med} \rightarrow C_2$. By defining a binary relation, the alignment makes the partial mapping between contexts possible. This feature enables dealing with situations where a concept of a context does not make sense in the other context. On the other hand, the entity must be totally mapped on both contexts: all concepts of the entity must be understood in both contexts.

Context Integration (Figure 3.3-B): These operations consider situations where a single entity E_{Med} has more than one context (C_1 and C_2): $C_1 \rightarrow E_{Med} \leftarrow C_2$. The context integration produces a new context $C_1 \rightarrow C \leftarrow C_2$ that combines information of the original context preserving the coherence with the entity. This operation can be used in situations where a single entity can be viewed in many ways, according to the considered context. The integration performs the amalgamated union of contexts, collapsing components that are images of the same component in the original entity.

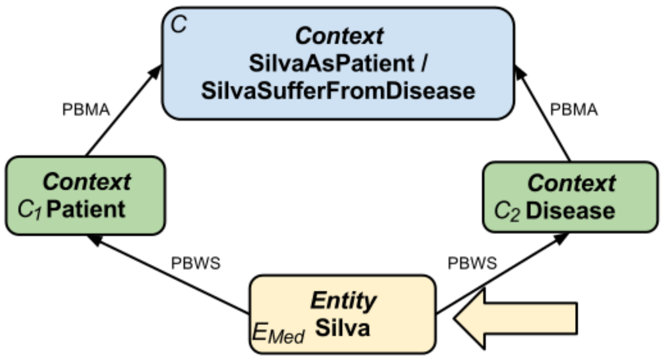


Figure 3.4: Example of Context Integration in the Patient-Buddy.

To exemplify this, figure 3.4 shows an example of Context Integration in the PatientBuddy framework. This context integration generates an ontology (C) that has information about both Silva as a Patient and Silva's Diseases. With the generated ontology (C), the system is able to obtain details about Silva's diseases as well as knowing who is the doctor responsible for him.

The arrows in the figure 3.4 represent the links between ontologies. The new context (C) combines information of C_1 and C_2 preserving the coherence with the entity E_{Med}. The resulting context contains all information of the original contexts (C_1 and C_2), but identifies parts related by the mediator entity (E_{Med}).

Collapsed Union (Figure 3.3-C): Is the amalgamated union of two contextualized ontologies mediated by a third contextualized ontology. It is the combined composition of entities and contexts, where the ontology links ensure the preservation of structure, relations, and axioms of each ontology and coherence of each entity with respect to contexts. It produces a new contextualized ontology with all components of the original ones, but collapsing components that have the same source in the mediator.

Coalignment (Figure 3.3-D): It is a mechanism to establish a correspondence between vocabularies of two ontologies by the use of an intermediate target ontology: $E_1 \to C_{Med} \leftarrow E_2$. It configures a binary relation between the two ontologies, where related components are those mapped in the same component of the intermediate target. By defining a binary relation, the coalignment allows the partial mapping between entities, meaning that not all concepts of one entity make sense for the other entity. Both entities, however, must be totally mapped in the context.

Entity Integration : (3.3-E) Giving a coalignment $E_1 \rightarrow C_{Med} \leftarrow E_2$, the entity integration produces a new entity E contextualized by the original ones (and by transitivity, by the original context C_{Med}). The entity integration performs the semantic intersection of the entities under the mediation of the context, that is, the new entity will embody all, and nothing more than, information of the original entities that are related by the coalignment $E_1 \rightarrow C_{Med} \leftarrow E_2$.

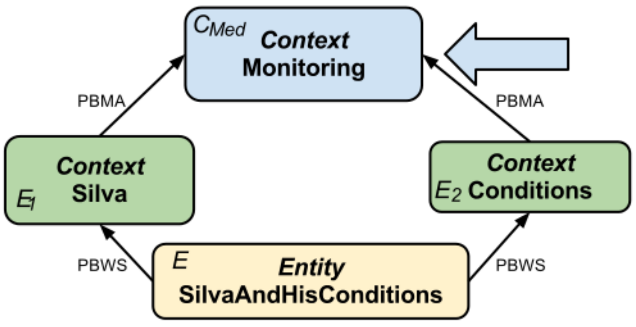


Figure 3.5: Example of Entity Integration in the Patient-Buddy.

An example of entity integration in the Patient-Buddy-Build framework is shown in figure 3.5. The entities Silva and Conditions are integrated under the mediation of the Monitoring context. The integration results in the new entity SilvaAndHisConditions (E). The new entity has information about Silva and as well as his monitored conditions. We use the term monitored conditions as a synonym of monitored context information.

Relative Intersection (Figure 3.3-F): Is the intersection of two contextualized ontologies mediated by a third contextualized ontology. It produces a new contextualized ontology having just the components of the originals that are mapped in the mediator.

In the appendices we describe the algorithms used by Patient-Buddy-Build framework to compose and decompose the contextualized ontologies. It is important to say that in this stage of the work this algorithms are not yet implemented, and because of that they were executed manually in order to create the prototype application (further explained in chapter 6).

In the following subsection we describe a simple scenario using the Patient-Buddy-Build framework, which illustrates the use of context in an ubiquitous environment. The numbers in italics between brackets are used to identify situations that will be further referred in section 3.4.

3.3 Scenario

Further in chapter 6 we describe the example application created using the Patiet-Buddy-Build framework. This application was created to monitor patients with Atrial Fibrillation. In this scenario, Silva is a patient that has Atrial Fibrillation and needs to be monitored by his doctor Mario.

Doctor Mario prescribed an anti-coagulant to Silva, in order to prevent his organism of the appearance of a coagulum. Silva carries with him a smart phone, which hosts the mobile context-aware application (described in section 5.2). The Patient Mobile Application responds to different situations, according to the monitoring process defined by Mario, environment conditions and health conditions of Silva.

As soon as Silva turns on his smart phone, the Patient-Buddy Mobile Application (PBMA) and the Context Management Service (CMS) starts. PBMA is the mobile application that will be running in Silva's smart phone (further explained in section 5.5), and CMS[1] is a service and software framework that manages the gathering, processing and distribution of any type of context data. With the Wi-fi or 3G connection enabled, the smart phone sends a request to the Patient-Buddy Web Server (PBWS) for the initial configuration(A). As the configuration is received, the CMS will start all the context providers according to the sensors that must be used to monitor patient Silva (B).

Doctor Mario defined that the smart phone application of the patient will alert Silva whenever Silva is located in a place that has an altitude higher than 2400 meters. Mario defined this action to be taken only if the patient has any prescribed anti-coagulant medication. Silva was on his way to Atacama Desert in Chile by car. (C). With the GPS enabled, when Silva's altitude passed the limit of 2400 meters, PBMA sends an alert to Silva's smart phone. The alert was sent to explain how it is dangerous to be in an altitude greater than 2400 meters when being a patient that takes anti-coagulant medication(D).

Silva read the alert from his smart phone, but decide to continue his travel. He was too excited about the possibility of seeing the city of Atacama. Knowing that Silva's altitude is greater than 2400 meters, the application is now configured to monitor new kinds of symptoms regarding the altitude. The application wants to take care of Silva's health, helping him to monitor his conditions.

Silva started to feel headaches, and pressed the report button that generates a questionnaire based on the current monitored information. Through

[1]Context Management Service (CMS): `http://www.lac.inf.puc-rio.br/software/context-management-service-cms`

this questionnaire, Silva answered yes to the headache question. PBMA placed that question there based on the fact that Silva was in an altitude higher than 2400 meters(E).

Some minutes later the sensor attached to Silva sends information via Bluetooth about the RR interval[2] of his electrocardiogram signal. The interval was not constant, as it should be, and also was variating more than 500 milliseconds. When PBMA receives this information, it concludes that Silva is fibrillating (F). The information required to deduce that Silva is fibrillating was defined by doctor Mario. It was only concluded because of the fact that Silva is also a intermittent patient. Based on Silva's monitoring process, PBMA sends an emergency message to the Silva's doctor and to his emergency contact (G) asking for emergency medical assistance within 6 hours.

3.4 Formalizing the Scenario

In this section we will apply the layered division approach to formalize the scenario described in the previous section. The goal is to use the scenario to show how the ontologies of entities and contexts were created. Only the (F) and (G) situations won't be formalized, since the objective here is to explain how we applied the framework developed in (11) and the previous situations already demonstrates the idea. The situation (F) and (G) are only to illustrate some functionalities and possible situations that the PBB framework could implement.

First we make a short description of each ontology of entity and context created for the PBB framework:

Ontologies of contexts:

– **Disease**: Contains information about one or more diseases, and what conditions are related to each of the diseases. Each information monitored from a patient is a condition. For example, heart rate and electrocardiogram signal information are relevant conditions to be monitored if a patient has Atrial fibrillation disease.

– **Environment**: Each of the monitored conditions has a form to be acquired from the environment of the patient. This ontology describes how each of the monitored conditions will be acquired. E.g.: If the heart rate will be acquired from a sensor attached to the patient or from a question.

[2]RR interval is measurement of the distance between two consecutive heart beats. It is measured from the electrocardiogram signal.

– **Questionnaire**: Contains information about the question and how it should be answered. For example, if the question should be answered by multiply choice (radio box or check box) or by using a free text (text box).
– **Person**: Personal information about the patient that is relevant, concerning the monitoring process. The personal information concerned in this framework was the city and country that the person lives, his/her name and smart phone number. E.g.: The name of the city and the country that the patient lives can be useful to determine if the patient current location is covered by his/her health plan. The smart phone number information is useful to send emergency messages about the health state of a patient during a monitoring process.
– **Patient**: This ontology contains information about which diseases a specific patient has and who is the responsible doctor and emergency contacts related to a patient. It also contains information about prescribed medications and specific characteristics of the patient (e.g.: If a patient has Asthma, a specific characteristic could be if he/she is more susceptible to have asthma attacks in a environment with high temperatures.).
– **Monitoring**: This ontology contains information related to the monitoring process. This ontology tells the application which are and how to identify the relevant situations to the monitoring of a patient (e.g.: The doctor can define to monitor a patient only if he is doing a physical activity and also describe how the application will determine whenever the patient is doing a physical activity). This ontology also determines in which situations the mobile application should send an alert message to the patient or to the doctor.

Ontologies of entities:

– **Conditions**: This ontology contains all the conditions that the Patient-Buddy-Build framework can monitor.
– **Questions**: All the questions that can be done by the Patient-Buddy-Build framework.
– **SMPPerson**: An ontology that contains the smart phone number of a person.
– **Monitoring Team**: The group of people that is responsible for the remote monitoring of a person.
– **Patient Data**: This ontology holds all the possible diseases and patient characteristics that are represented within the Patient-Buddy-Build framework.

Using this ontologies of entities and context we created the following contextualized ontologies:

1. **Conditions → Environment**
2. **Conditions → Questionnaire**
3. **Questions → Environment**
4. **Questions → Questionnaire**
5. **SMPPerson → Monitoring**
6. **SMPPerson → Person**
7. **Monitoring Team → Person**
8. **Monitoring Team → Patient**
9. **Monitoring Team → Monitoring**
10. **Patient Data → Disease**
11. **Patient Data → Patient**

All the ontologies, can be customized to any person. Within this section the reader will notice that some ontologies are described as customizations of the ontologies listed above. Since in this scenario we are instantiating the PBB framework to remote monitor a specific patient (for Silva). For example, "SilvaAsMonitoredPerson" is a customization of the *Monitoring ontology* and it represents the monitoring process of Silva.

It is important to highlight that in PBB, ontologies of context can be seen as an entity and an entity can also be seen as a context. Detailed information about each of the ontologies can be found in the next chapter and in the appendices. In order to determine these ontologies of entities and contexts, we use the three layered approach proposed by (11). The first diagram, the diagram of devices, was described in the previous subsection 3.1.1. The diagrams of entities and ontologies are described in the next subsections.

3.4.1 Diagram of Entities

In order to detail the devices of the diagram of devices, each of them have a diagram of entities. Using the example scenario described in section 3.3, we will show the entities of each device. The numbers in brackets bellow are referencing situations described in the scenario section 3.3.

Diagram of entities of PBWS: First at (A), when Silva turns on his smart phone. The *Patient-Buddy Mobile Application* (PBMA) sends a request to the Patient-Buddy Web Server (PBWS) for the initial configuration of the monitoring process. For the application determine which is the monitoring process for patient Silva, it is required the integration of the ontologies *Silva* and the *Monitoring* with respect to the smart phone of patient Silva. There is an ontology that represents the smart phone of patient Silva: *SMPSilva*. This means that the concepts and relations of *SMPSilva* will be linked into correspondents of *Silva* and *Monitoring*, respecting the structure of the ontologies. This will result in a context integration, *SMPSilva* will act as mediator of *Silva* and *Monitoring*: *Silva ←PBWS- SMPSilva -PBWS→ Monitoring*. As a result of the integration a new ontology appears and it is named as *SilvaAsMonitoredPerson*.

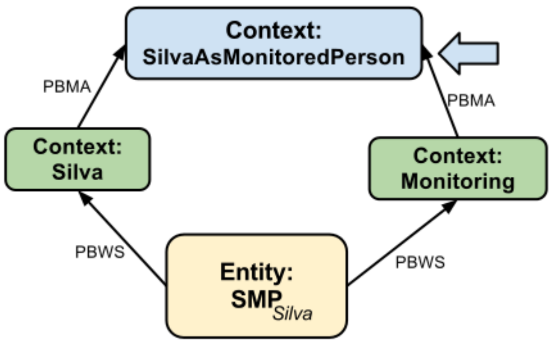


Figure 3.6: Considering the ontologies *Silva* and *Monitoring*, PBWS generates the *SilvaAsMonitoredPerson* ontology.

With this integrated context, PBMA will have enough information to know which is the right monitoring process to be configured for Silva's smart phone.

Diagram of entities of CMS: After, in (B), CMS will start all the context providers required to monitor Silva. To know which are the sensors that need to be active, CMS will need more than only the information contained in *SilvaAsMonitoredPerson* context, but will also need information about Silva's environment. Figure3.7, shows the context integration that CMS will execute in order to be able to initialize all the context providers required for Silva's monitoring.

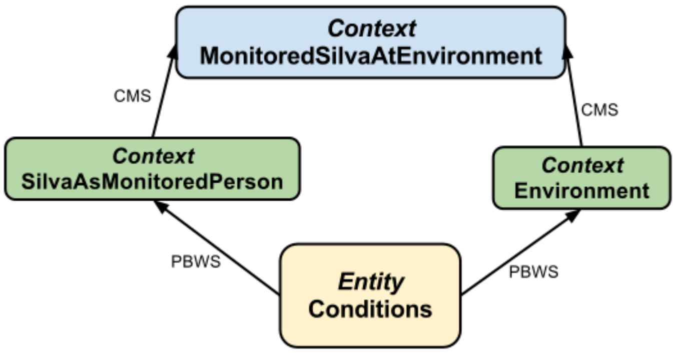


Figure 3.7: Considering the ontologies: *SilvaAsMonitoredPerson* and *Environment*, PBMA generates the *MonitoredSilvaAtEnvironment* ontology.

Diagram of entities of PBMA: Later, the patient Silva is travelling to Atacama Desert in Chile. During his way, Silva's location altitude passed the limit of 2400 meters(C). To detect this situation, PBMA needs to know how the altitude information is acquired. But according to the described scenario, the altitude condition only exists because Silva takes an anti-coagulant. Thus, PBMA also needs to know about Silva's prescribed medication.

There is a very simple ontology that contains the conditions that could be monitored using the PBB system. This ontology is called *Conditions*. *Silva's conditions* is an ontology that contains information about Silva and the list of conditions that are being monitored from him.

With the integration (*Silva's Conditions* ← *PBMA* - *Conditions* - *PBMA*→ *Environment*) we generate a context ontology (*SilvaAtEnvironment*). With the *SilvaAtEnvironment* ontology, PBMA can properly know how the altitude information is acquired (i.e.: from sensors or questions). But the altitude information is relevant to be monitored only if Silva takes an anti-coagulant. The information about Silva's medication (e.g.: if he takes an anti-coagulant or not) comes from the context of Silva as a patient(*SilvaAsPatient*). Thus, the information about Silva as a patient must also be taken into account for setting the Silva's monitoring configuration properly.

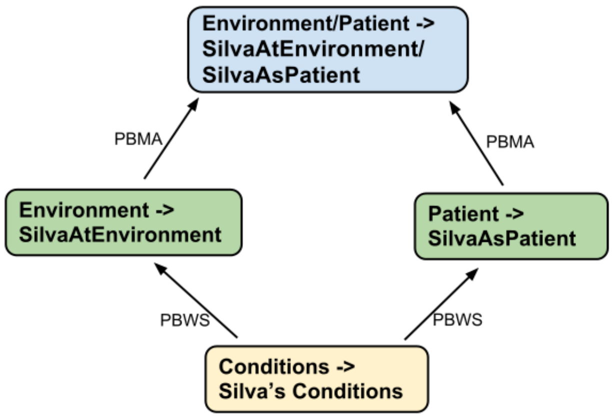

Environment/Patient -> SilvaAtEnvironment/ SilvaAsPatient
PBMA
PBMA
Environment -> SilvaAtEnvironment
Patient -> SilvaAsPatient
PBWS
PBWS
Conditions -> Silva's Conditions

Figure 3.8: Collapsed Union Example.

The context integration (*Environment ←PBMA- Conditions -PBMA→ Patient*) generates a context where PBMA can find what information of the environment of Silva is important to monitor and information about Silva as a Patient (e.g.: The prescribed medication of Silva). This generated context *SilvaAtEnvironment/SilvaAsPatient* is represented in the base square of 3.8.

On the top square of figure 3.8, is represented the context integration (*SilvaAtEnvironment ←PBMA- Silva's Conditions -PBMA→ SilvaAsPatient*). This integration generates a context where PBMA can find not only information about Silva's prescribed medication or about how to acquire the altitude information, but also information about all the monitored conditions of Silva. Note that figure 3.8 also pictures a combined integration: the collapsed union of the contextualized entities *Patient → SilvaAsPatient*, *Environment → SilvaAtEnvironment* mediated by *Conditions → Silva's Conditions*.

Now, Silva's smart phone must alert him about the altitude, but to do that, PBMA must have information about the context of *SilvaAsMonitoredPerson* (D). The context integration *Patient ← Conditions →Monitoring* generates it. In this context there is information about the actions that must be taken if Silva's location altitude passed the limit of 2400 meters. In figure 3.9, the collapsed union of the contextualized entities is shown:

– *Conditions → Silva's Conditions* (Mediator)

– *Patient → SilvaAsPatient*

– *Environment → SilvaAtEnvironment*

– *Monitoring → SilvaAsMonitoredPerson*

With this integration, it will generate two specific contextualized ontologies: (i) *Environment/Patient → SilvaAtEnvironment/SilvaAsPatient* and (ii) *Patient/Monitoring → SilvaAsPatient/SilvaAsMonitoredPerson*. With these two contexts, we apply the Context Integration operation under the mediation of *Patient* → *SilvaAsPatient* to generate the context represented in the green box (located in the top of the figure 3.9). PBMA will now have necessary information to send an alert to Silva. And also, to explain to Silva what is the danger of the high altitude to users of an anti-coagulant medication.

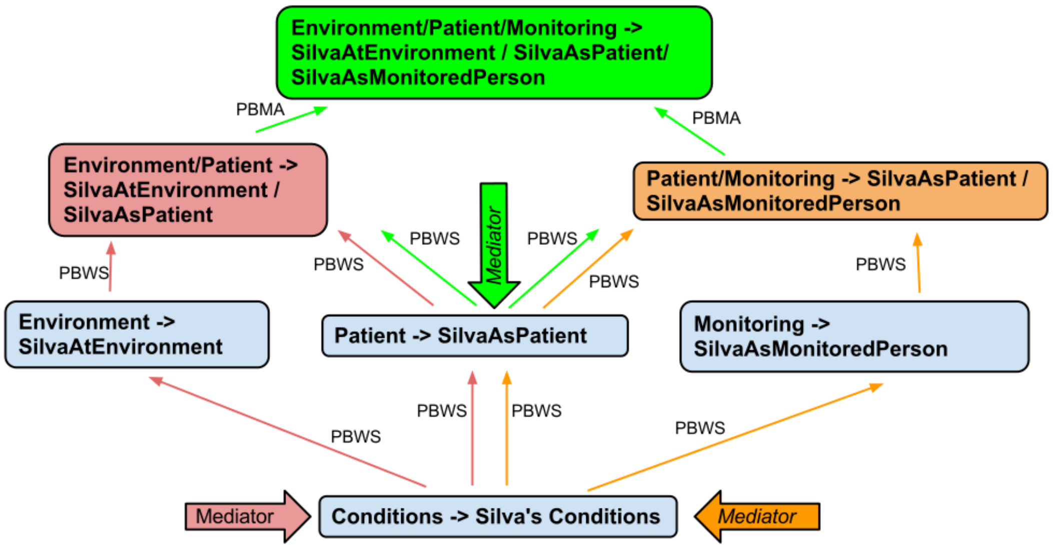

Environment/Patient/Monitoring -> SilvaAtEnvironment / SilvaAsPatient/ SilvaAsMonitoredPerson
PBMA
PBMA
Environment/Patient -> SilvaAtEnvironment / SilvaAsPatient
Patient/Monitoring -> SilvaAsPatient / SilvaAsMonitoredPerson
Mediator
PBWS
PBWS
PBWS
PBWS
PBWS
PBWS
Environment -> SilvaAtEnvironment
Patient -> SilvaAsPatient
Monitoring -> SilvaAsMonitoredPerson
PBWS
PBWS
PBWS
PBWS
Mediator
Conditions -> Silva's Conditions
Mediator

Figure 3.9: The Collapsed Unions: **(1)** *Patient* → *SilvaAsPatient*, *Environment* → *SilvaAtEnvironment* mediated by *Conditions* → *Silva's Conditions*(arrows with pink color). **(2)** *Monitoring* → *SilvaAsMonitoredPerson*, *Patient* → *SilvaAsPatient* mediated by *Conditions* → *Silva's Conditions*(arrows with orange color). **(3)** *Environment/Patient* → *SilvaAtEnvironment/SilvaAsPatient*, *Patient/Monitoring* → *SilvaAsPatient/SilvaAsMonitoredPerson* mediated by *Patient* → *SilvaAsPatient*.

After this, in (E), Silva started to feel a headache and pressed the report symptom button of his smart phone. This button generates a group of questions that are related to the current monitoring state and the disease of Silva. To assemble an ontology that has all the information required to generate this questionnaire, we will need the following contextualized ontologies:

1. *Conditions* → *Silva's Conditions*
2. *Disease* → *SilvaSufferFromDisease*
3. *Environment* → *SilvaAtEnvironment*
4. *Monitoring* → *SilvaAsMonitoredPerson*
5. *Questionnaire* → *SilvaQuestionnaireForm*

First, PBMA needs to know about all the disease's conditions that Silva suffers from. The context integration *Patient ←PBMA- Silva -PBMA→ Disease* will generate a new ontology that we will name *SilvaSufferFromDisease*. It will embody all components of *Patient*, all components of *Disease*, and will have the images of concepts of the mediator *Silva* collapsed. The *SilvaSufferFromDisease* context, represented in 3.10, will have the information about the internal conditions that affect Silva. When we say internal condition, it means it is a condition that comes from within the body of Silva, for example, his body temperature or his heart rate.

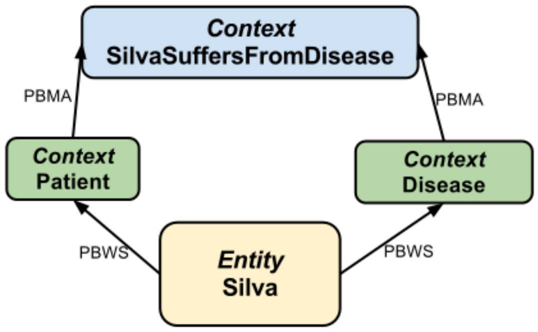


Figure 3.10: The context integration from the ontologies *Silva*, *Patient* and *Disease*. It generates the SilvaSufferFromDiease context.

The external conditions that affect Silva, for example the altitude of the place, must also be taken into account for generating his questionnaire. Thus, if Silva is located in a place with an altitude higher than 2400 meters, the system must ask questions related with altitude symptoms. In order to do that, we use the integrated context *SilvaAtEnvironment*, generated by the integration: *Environment ←PBMA- Silva -PBMA→Patient*. The integrated context *SilvaAtEnvironment* will also have information about which of the conditions are acquired from the question.

With *SilvaSufferFromDiease* and *SilvaAtEnvironment* ontologies, PBMA will have enough information to know which are the conditions that can affect Silva and which of them are acquired from questions. In order to generate a questionnaire, PBMA needs information about Silva's monitoring process, to know which of the conditions are important to be acquired given Silva's monitoring state. The integrated context (*SilvaAsMonitoredPerson*) that contains this information is acquired from the integration (*Patient←PBMA- Silva -PBMA→Monitoring*).

In the Patient-Buddy-Build framework, the questionnaire can be customized for each person. For example, doctor Menezes can configure a specific question to be answered in text form, using radio boxes or check boxes.

Because of that, the information about the questionnaire must also be integrated. PBMA generates it with the integrated context (*Person* ←*PBMA-Silva-PBMA*→ *Questionnaire*) that produces the ontology *GenerateQuestionnaire*.

Using a collapsed union of the contextualized ontologies:

– *Environment* → *SilvaAtEnvironment*

– *Questionnaire* → *SilvaQuestionnaireForm*

– *Monitoring* → *SilvaAsMonitoredPerson*

Under the mediation of *SilvaSufferFromDisease*, PBMA generates a context that is represented in the green box of figure 3.11 (SilvaAtEnvironment/SilvaQuestionnaireForm/SilvaAsMonitoredPerson). With this generated context, PBMA will have enough information to generate the questionnaire for Silva. The questionnaire will be targeting all the conditions that are important to be acquired from Silva and will be created using Silva's preferences for answering it (e.g.: using a free text or a multiply choice answer).

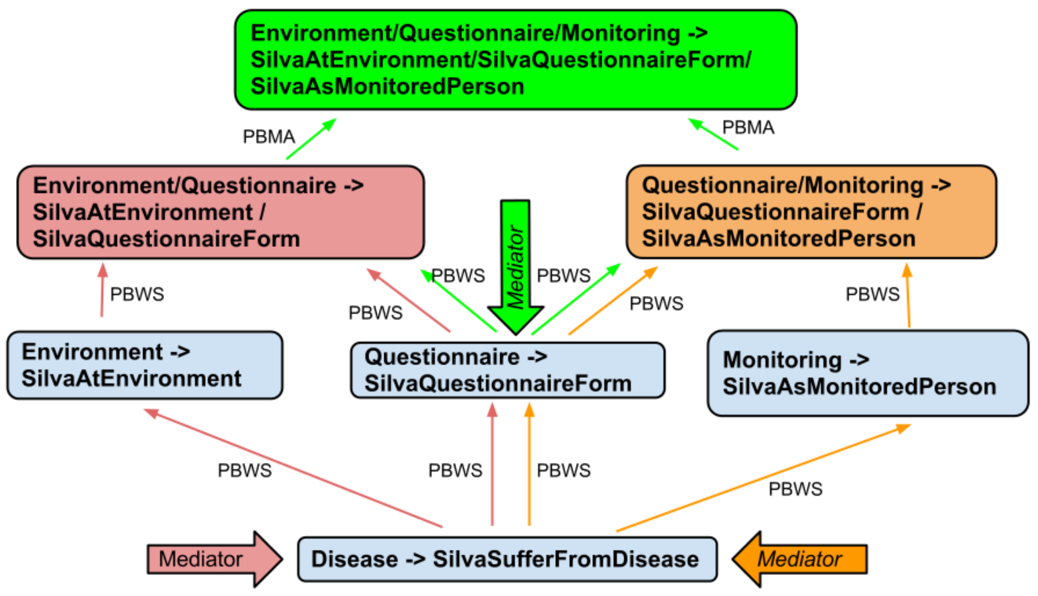

Environment/Questionnaire/Monitoring -> SilvaAtEnvironment/SilvaQuestionnaireForm/ SilvaAsMonitoredPerson
PBMA
PBMA
Environment/Questionnaire -> SilvaAtEnvironment / SilvaQuestionnaireForm
Mediator
Questionnaire/Monitoring -> SilvaQuestionnaireForm / SilvaAsMonitoredPerson
PBWS
PBWS
PBWS
PBWS
PBWS
PBWS
Environment -> SilvaAtEnvironment
Questionnaire -> SilvaQuestionnaireForm
Monitoring -> SilvaAsMonitoredPerson
PBWS
PBWS
PBWS
PBWS
Mediator
Disease -> SilvaSufferFromDisease
Mediator

Figure 3.11: The Collapsed Unions: (1) *Questionnaire* → *SilvaQuestionnaireForm*, *Environment* → *SilvaAtEnvironment* mediated by *Disease* → *SilvaSufferFromDisease*(arrows with pink color). (2) *Monitoring* → *SilvaAsMonitoredPerson*, *Questionnaire* → *SilvaQuestionnaireForm* mediated by *Disease* → *SilvaSufferFromDisease*(arrows with orange color). (3) *Environment/Questionnaire* → *SilvaAtEnvironment* / *SilvaQuestionnaireForm*, *Questionnaire/-Monitoring* → *SilvaQuestionnaireForm* / *SilvaAsMonitoredPerson* mediated by *Questionnaire* → *SilvaQuestionnaireForm*.

3.4.2 Diagram of Ontologies

We selected the first two situations of the scenario described to show how the framework used in this thesis provides the required information to adapt services or behaviours according to the context changes.

In situation (A), the system must send back to the smart phone the requested initial configuration for the Silva's monitoring process. We have information about Silva's smart phone that comes to the context of Silva as a Person. The system now knows personal information about the owner of the smart phone (Silva). This information will be used in a different context, the Monitoring context, where Silva will have a specific configuration for his monitoring process.

The association that makes Silva have a monitoring process is written in OWL (Ontology Web Language).

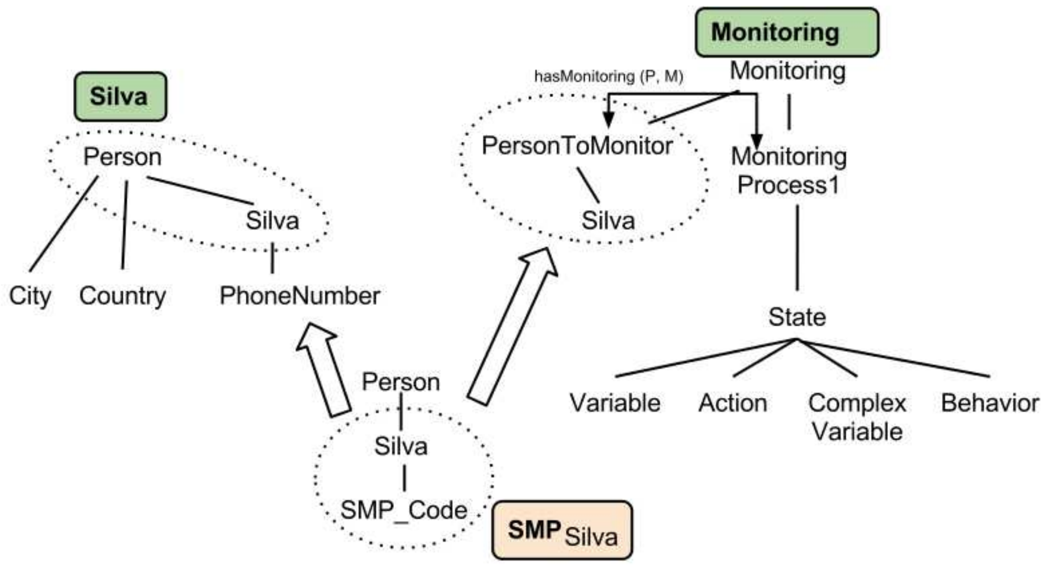


Figure 3.12: Zooming the alignment of *Silva* and *Monitoring* ontologies under the mediation of *SMPSilva*.

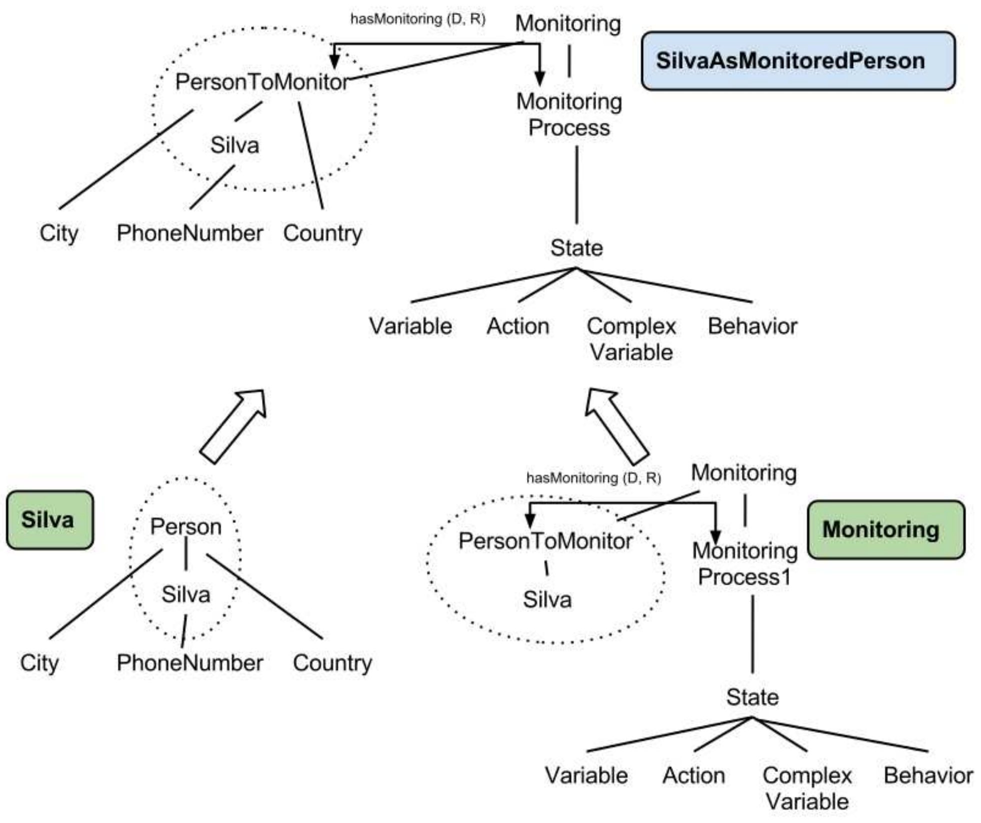


Figure 3.13: The context integration of the alignment of figure 3.12. The information about which is the monitoring process of Silva and his phone number are available.

Considering figure 3.6, the mediator of the context integration (*Silva* $\leftarrow$*PBMA* - SMPSilva - *PBMA* $\rightarrow$ *Monitoring*) must capture the fact about Silva as a Person and properly map this information into the *Monitoring* ontology. This is represented in alignment shown in 3.12, between the ontologies of *Silva* and *Monitoring*.

The integrated context resulted is the *SilvaAsMonitoredPerson* ontology, that will hold information about the monitoring process of Silva, his phone number and also personal information about Silva. Using this integrated context, PBMA will have enough information to send to Silva's monitoring process configuration to his smart phone.

Later, in the situation (B), where the system need to know which context providers have to be started in order to monitor all the conditions of Silva. The *Environment* ontology has all context providers, and each of them is responsible for monitoring a set of conditions. Figure 3.15 shows the resulting entity, the integration between *SilvaAsMonitoredPerson* and *Environment* ontology that is represented in figure 3.14. It only contains the set of conditions that belongs to both of the integrated ontologies. Thus, with this integration the system will have the set of conditions of Silva that are monitored from context providers. The CMS will now be capable to activate only the necessary context providers.

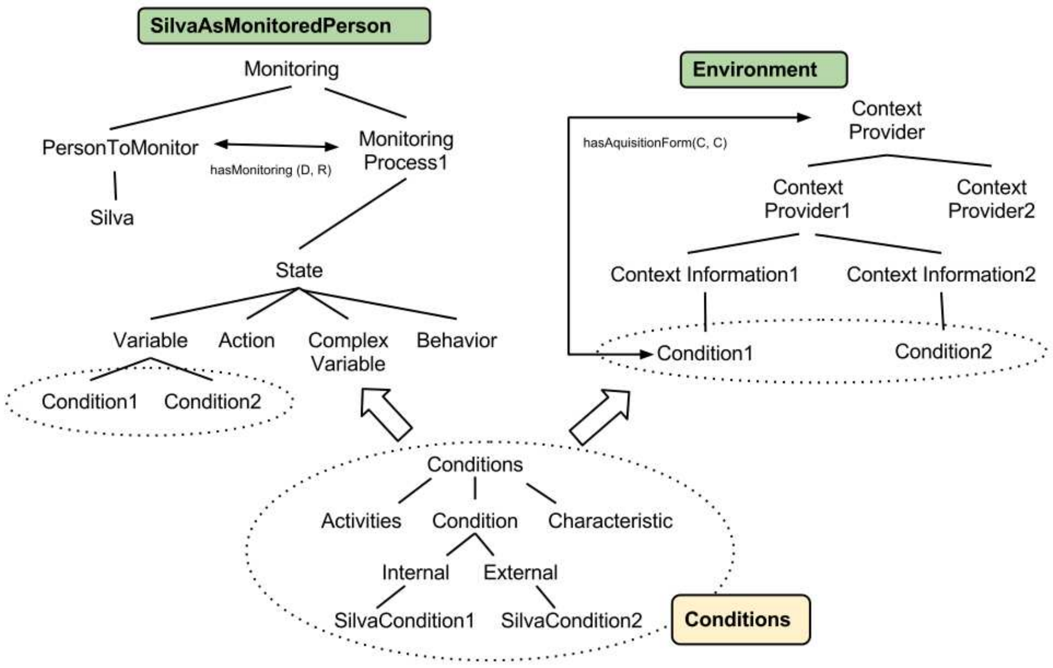


Figure 3.14: The alignment of *SilvaAsMonitoredPerson* and *Environment* ontologies with respect to the monitored conditions of Silva.

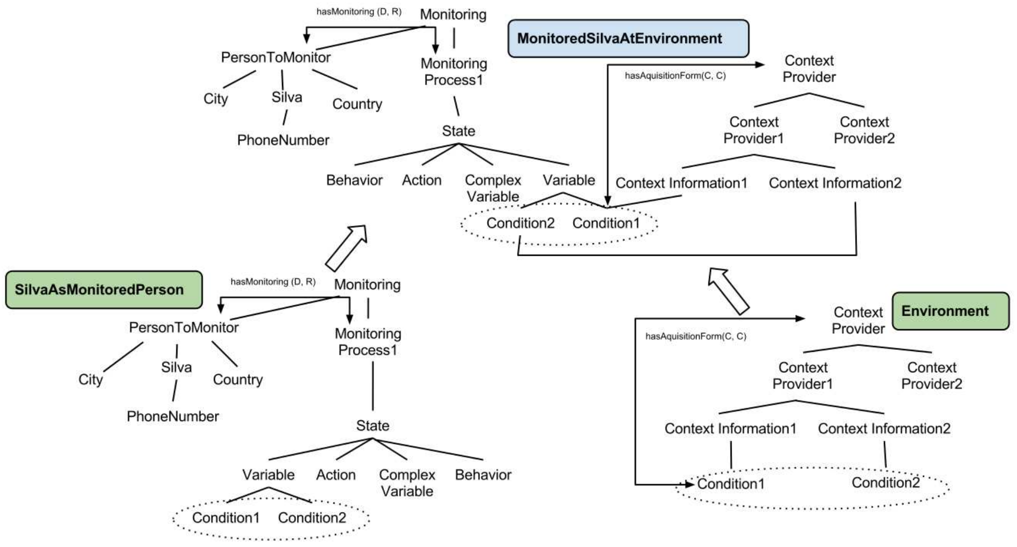


Figure 3.15: Integrating the context *SilvaAsMonitoredPerson* with *Environment* and generating the *MonitoredSilvaAtEnvirnment* ontology.

4 Patient-Buddy-Build Ontologies and Customizations

There are two ways to customize an application instantiated using the PBB framework: through the customization of the ontologies or the extension of two Java classes: *ContextProvider* and *PBB_ContextConsumer*. The extension of this two classes gives the possibility to the user to add new sensors and/or to use new web services (e.g.: to use new web services from the web to identify the whether of the place that the patient is located or to add a new sensor to monitor the level of glucose in the blood).

All the rest of the code of the PBB framework stays the same for every instantiated application, thus concentrating all the other customizations by through editing ontologies.

To explain each ontology and how they can be customized, we will use a lot the word "'*condition*". *Conditions*, in this thesis, are every context information that are considered relevant to the remote monitoring of the patient. Thus a condition can be, for example, a symptom of the patient (e.g.: fever, headache), a data from the patient like his/her heart rate or blood pressure or even the altitude of the place where the patient is located, if this affects his/her monitoring.

4.1 Disease Ontology

The disease ontology contains the knowledge about what is a disease to the PBB framework. For the PBB, in what concerns a remote monitoring scenario, a disease is a set of *conditions* and *characteristics* that must be monitored from the patient. Both *conditions* and *characteristics* are context information about the patient, the only difference between them is that the *characteristics* represents the static context information while the *conditions* represents a more dynamic type of context information. For example, a *characteristic* could be the type of blood of the patient and a *condition* could be the patient's heart rate.

Therefore, for each context information to be monitored by the system, it must be created a *condition* or a *characteristic* that represents it in the disease ontology. The disease ontology is illustrated by the figure 4.1 bellow.

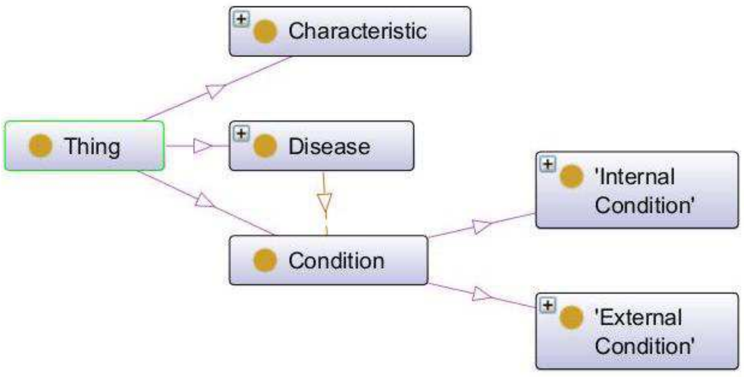


Figure 4.1: Disease Ontology Structure

All context informations that must be monitored by the system will be an instance of the class **Condition** or an instance of the class **Characteristic**. And will be related to zero or more diseases. The condition can be divided into internal and external condition. **Internal Conditions** are information related to the body of the patient (e.g.: The heart rate, breath rate, if he is feeling headache). **External conditions** are acquired from the environment that the patient is inserted (e.g.: Patient's location altitude or climate).

The diseases are instances of the class *Disease*. To relate any condition to an instance of a *Disease* class only use the object property *has_condition*. Similary, to relate any characteristic to an instance of a **Disease** class only use the object property *has_characteristic*. As represented in the figure 4.1, the object property *has_condition* has the class *Disease* as its domain and the class *Condition* as its range. For example, if the ontology engineer wants to customize the disease ontology to associate the heart rate to the atrial fibrillation disease, he/she must do the following steps:

– Create an instance of the Disease class to represent the atrial fibrillation disease.

– Create an instance of the Internal Condition class to represent the heart rate condition.

– Use the object property *has_condition* to associate the atrial fibrillation disease with the heart rate instance.

The process for associating a characteristic to a disease is similar, but it uses the object property *has_characteristic* and the characteristic must be an instance of the *Characteristic* class.

All the ontologies have some assertions made using description logic with the objective to help the ontology engineer on the task of customizing the ontologies for a new remote monitoring of a patient. Figure 4.2 shows an example of this expressions made with description logic. All the expressions about all the ontologies are showed in the appendices‘ chapters.

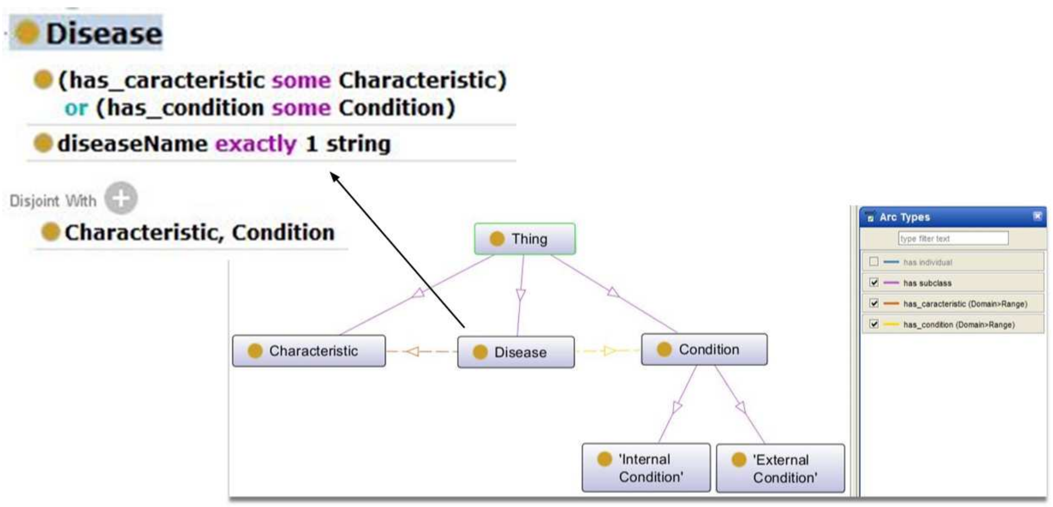

Disease
(has_caracteristic some Characteristic) or (has_condition some Condition)
diseaseName exactly 1 string
Disjoint With
Characteristic, Condition
Thing
Characteristic
Disease
Condition
'Internal Condition'
'External Condition'
Arc Types
type filter text
has individual
has subclass
has_caracteristic (Domain>Range)
has_condition (Domain>Range)

Figure 4.2: Disease Ontology

The expressions showed in the figure are: (A) A disease must have a *has_caracteristic* property with some **Characteristic** instance or has_condition property with some **Condition** instance, (B) A disease must have a *diseaseName* property with exactly 1 name and it must be a string; (C) The **Disease Characteristic** and Condition classes are disjoint. The expression (A) says that any disease must have at least one condition or characteristic associated to it. In the remote monitor scenario, there is no meaning to add a disease that has no context information to be monitored associated to it. Expression (B) says that a disease instance must have a name. This expression was inserted in the ontology, because the context-aware application developed with this thesis uses the disease name to generate reports about the patient. So it is convenient to the ontology engineer to not forget to use the *diseaseName* property, because it will give a name to the disease.

4.2 Questionnaire Ontology

For each information that the doctor wants to acquire from the patient where it is acquired from a question, the doctor can define the form that the question will be presented to the patient. Figure 4.3 below, represents the structure of the questionnaire ontology.

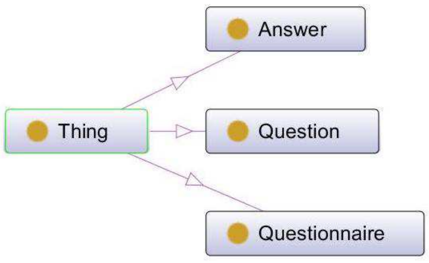


Figure 4.3: Questionnaire Ontology Structure

Instances of the class **Questionnaire** can have one or more instances of the class **Question** associated to it. And instances of the class **Question** can have zero or more instances of the class **Answer** associated to it

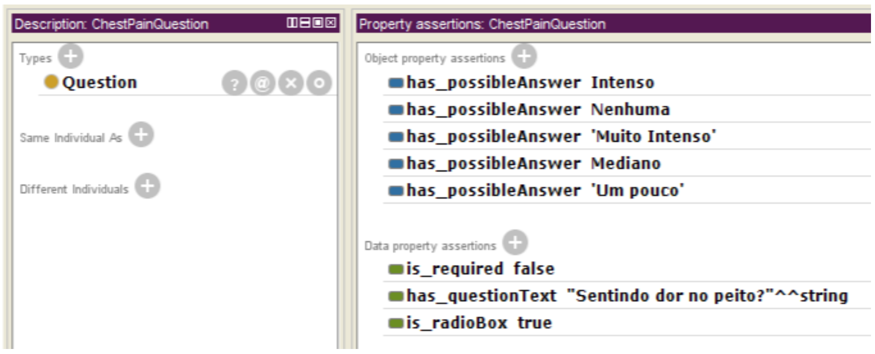
Each of the classes have properties that allows the following customizations for the questionnaires:

1. A question can be configured to have multiply choice possibility that the patient can choose multiple answers. These will be presented to the patient in the form of Check Boxes.

2. A question can be configured to have multiple choices answers that the patient can only choose one of them. This option will be presented to the patient in the form of Radio Boxes or Combo Boxes (depending on the doctor's preferences).

3. A question can be configured to be answered by writing a free text.

Figure 4.4 below exemplifies the creation of a question using the questionnaire ontology.


Description: ChestPainQuestion

Types
Question

Same Individual As

Different Individuals

Property assertions: ChestPainQuestion

Object property assertions
has_possibleAnswer Intenso
has_possibleAnswer Nenhuma
has_possibleAnswer 'Muito Intenso'
has_possibleAnswer Mediano
has_possibleAnswer 'Um pouco'

Data property assertions
is_required false
has_questionText "Sentindo dor no peito?"^^string
is_radioBox true

Figure 4.4: Feeling chest pain question.

The question is: "Are you feeling chest pain?" and it was configured to be answered by multiple choice and with only one possible answer.

As a first step to create this question in the **questionnaire ontology**, the ontology engineer have to create an instance of the class Question. In the case of the example showed in figure 4.4, the instance name is **Chest-PainQuestion**.

To attach to question text the data property **has_questionText** is used. Radio box was the form chosen for this question to be answered, by using the data property **is_radioBox** and selecting **true** as the property's value. By choosing to use radio boxes, the answer automatically will be presented as multiple choice and with only one possible answer.

The object property **has_possibleAnswer** is used to attach in the question all the instances of the class **Answer** that can be an answer for the question. Finally, the value of the data property **is_required** is false and it means that the question is not mandatory for the patient to answer, it is only optional (the value can be true, if the question is mandatory or false if it is optional).

4.3 Environment Ontology

The *environment ontology* represents the knowledge that PBB have about the monitored patient in the context of his/her surroundings. Some conditions are only defined in this ontology because they are not related to a specific disease. For example, the city where the patient lives is a condition that is not related directly to a disease but can be important to be monitored (i.e.: to know when the patient is covered by his/her health care plan).

For each condition, the ontology engineer can defines the following:

1. Define how the condition will be acquired. If it will be acquired by questions, by sensors or by a web service.

2. Select the context providers that the application will load in order to monitor the desired conditions.

The *environment ontology* structure is represented by the figure 4.5 below.

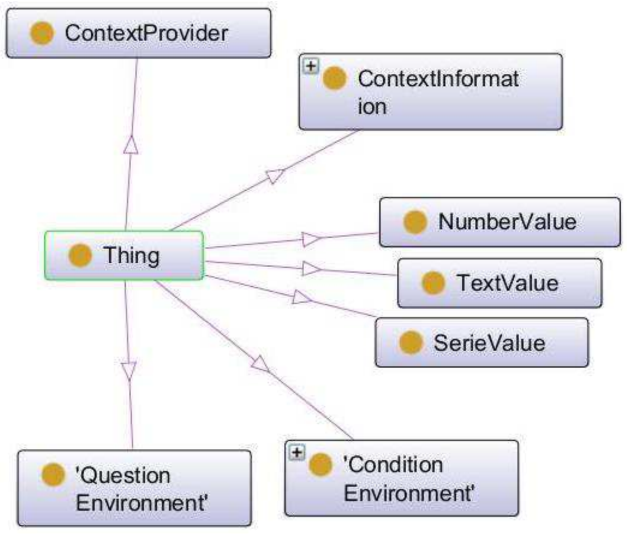


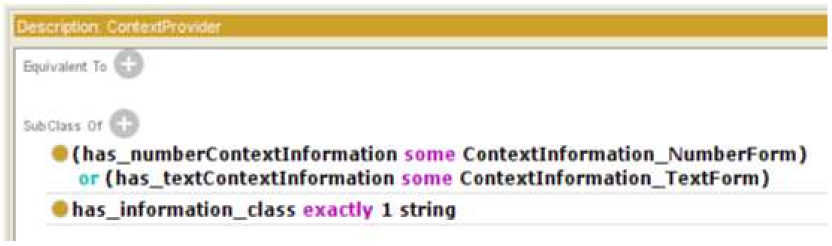
Figure 4.5: Environment Ontology Structure

Earlier in section X, we introduced the CMS (Context Management Service) and explained that this is a service that PBB uses to manager how the context information provided by the sensors and the web services will be acquired. Context Providers encapsulates the complexity of how to communicate to sensors and web services, and expose to PBB only the information acquired.

The PBB framework includes some context providers that were implemented for the remote monitoring application of patients with atrial fibrillation, explained in chapter 6.

By creating instances of the **ContextProvider** class of the environment ontology, the ontology engineer can configure the PBB to load only the necessary Context Providers from CMS in the mobile device. Each instance of the environment ontology‘s class **ContextProvider** represents one Context Provider from CMS.

As each Context Provider from CMS can monitor one or more conditions from a patient, the environment ontology lets the ontology engineer associate each condition to a **ContextProvider** instance, by using the class **ContextInformation**. Each context provider must be associated with at least one instance of the class **ContextInformation**, this is represented by the description logic expression showed in figure 4.6 4.6.

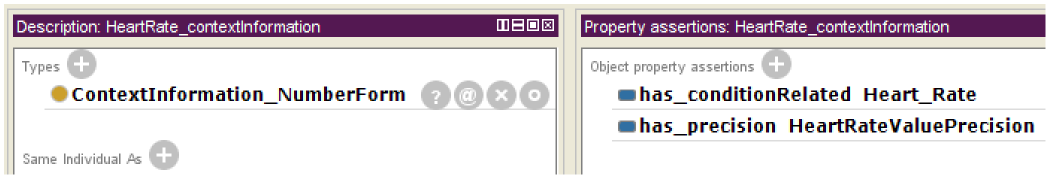


(has_numberContextInformation some ContextInformation_NumberForm)
or (has_textContextInformation some ContextInformation_TextForm)
has_information_class exactly 1 string

Figure 4.6: Context Provider class restrictions

The **ContextInformation** class from the environment ontology, represents a condition that a Context Provider can acquire. As so, when creating an instance of a **ContextProvider** class, it is necessary to associate to which instances of the class **ContextInformation** it is related.

The **ContextInformation** class have two sub classes. When the condition that will be acquired from the Context Provider is a number, this condition must be related to an instance of the class **ContextInformation_NumberForm**. And if the condition is a string, it must be related to an instance of the class **ContextInformation_TextForm**.

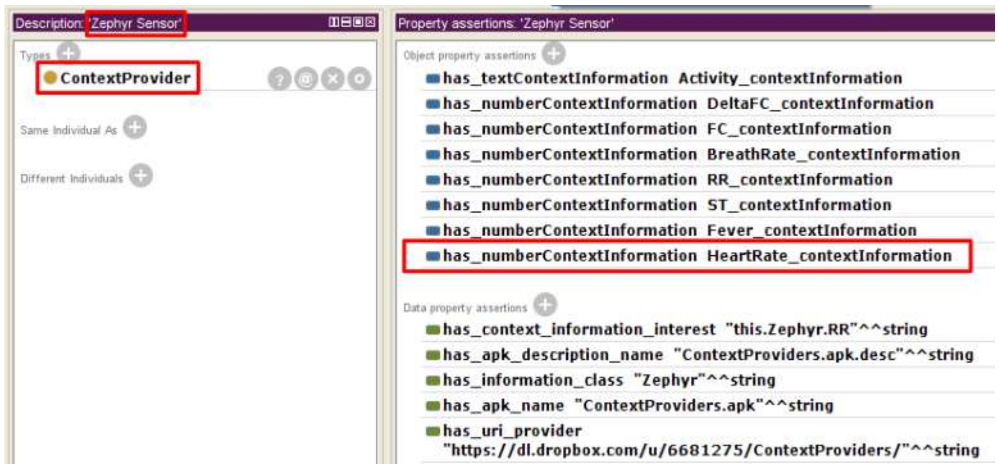

Description: HeartRate_contextInformation
Types
Same Individual As
Property assertions: HeartRate_contextInformation
Object property assertions

ContextInformation_NumberForm
has_conditionRelated Heart_Rate
has_precision HeartRateValuePrecision

Figure 4.7: Representing the heart rate condition as a context information acquired from the Zephyr context provider.

For example, the CMS have a context provider that uses a sensor (we will name this sensor by Zephyr) to acquire the heart rate of the patient. Figure 4.7 shows the first step that the ontology engineer will do to represent it in the environment ontology. First the ontology engineer will create an instance of the class **ContextInformation** to represent the heart rate condition, the instance is showed by figure 4.7 by the name **HeartRate_ContextInformation**). This instance will contain informations associated to the heart rate value that is acquired by the CMS context provider:

- The precision: How precise is the heart rate value acquired by the sensor X.
- An condition related: The instance of the **ConditionEnvironment** that is related to this instance of **ContextInformation**, in this case the **HeartRate** instance.



Description: 'Zephyr Sensor'

Types

ContextProvider

Same Individual As

Different Individuals

Property assertions: 'Zephyr Sensor'

Object property assertions

- has_textContextInformation Activity_contextInformation
- has_numberContextInformation DeltaFC_contextInformation
- has_numberContextInformation FC_contextInformation
- has_numberContextInformation BreathRate_contextInformation
- has_numberContextInformation RR_contextInformation
- has_numberContextInformation ST_contextInformation
- has_numberContextInformation Fever_contextInformation
- has_numberContextInformation HeartRate_contextInformation

Data property assertions

- has_context_information_interest "this.Zephyr.RR"^^string
- has_apk_description_name "ContextProviders.apk.desc"^^string
- has_information_class "Zephyr"^^string
- has_apk_name "ContextProviders.apk"^^string
- has_uri_provider "https://dl.dropbox.com/u/6681275/ContextProviders/"^^string

Figure 4.8: The instance of the class **ContextProvider** that represents the Zephyr context provider.

Figure 4.8 shows the second step that the ontology engineer will make to represent that the heart rate will be acquired from the Zephyr context provider. He/she will create an instance of the **ContextProvider** class to represent the Zephyr sensor (the instance named **Zephyr**) and use the property **has_numberContextInformation** to associante the **Zephyr** context provider instance to the **HeartRate_ContextInformation** instance.

4.4 Monitoring Process ontology

It is important to highlight that since the vocabulary to describe how the remote monitoring will be done is limited, this restrings its usage for a limited set of chronic diseases. The vocabulary was designed to solve the problem of patients with atrial fibrillation. However, many of the abstractions made in the construction of this model, can be useful for representing others chronic diseases monitoring process.

The monitoring process, can be seen as a diagnosis procedure (48), that will contain information about how will the knowledge (of the disease, the monitored context information, about prescribed medications) be applied to monitor a patient.

The monitoring process ontology represents the knowledge that the application have about what is a process to monitor patients with any disease. This ontology defines a vocabulary that provides a way for the doctor (with the help of an ontology engineer) to explain to the application how the monitoring process of his/her patient will be done.

To be able to develop a system that can monitor different patients with different diseases, we must give the doctor the capability of choosing how the system will apply its knowledge to monitor each of his/her patients.

In the health sector, there is no strict protocol that all the doctors will follow to monitor all patients. Each patient, is a different patient with a different health history and set of diseases, allergies and etc. Thus, even if two patients have the same disease, they might have different monitoring process. Two doctors can have different views regarding the treatment about the same patient too. They might want to monitor the same patient with different approaches. Another way to justify the need of customization for an application that remote monitor patients with different diseases is citing the medical guidelines. A medical guideline is a document with the aim of guiding decisions and criteria regarding diagnosis management and treatment in specific areas of healthcare. Such documents have been used for many years during the history of medicine. This documents are not protocols to be restrictedly followed, their objective is only to guide the doctor on how to manage and treat each disease. Thus, each doctor must have a way for customize his/her monitoring process about each patient and each disease.

In order to define a remote monitoring, it was developed a model that can be viewed as a *state* machine (an automata). The word *state* will be used constantly in this thesis; it represents a situation that a patient can be in. Situations in the real life of the patient that the doctor wants to monitor or worry about concerning the monitoring process.

Each *state* has three types of *variables*, *actions* and *behaviours* linked to it. Figure 4.9 shows an example of *state*. This *state* is part of the set of *states* created with the example application for monitoring atrial fibrillation and is explained in chapter 6.

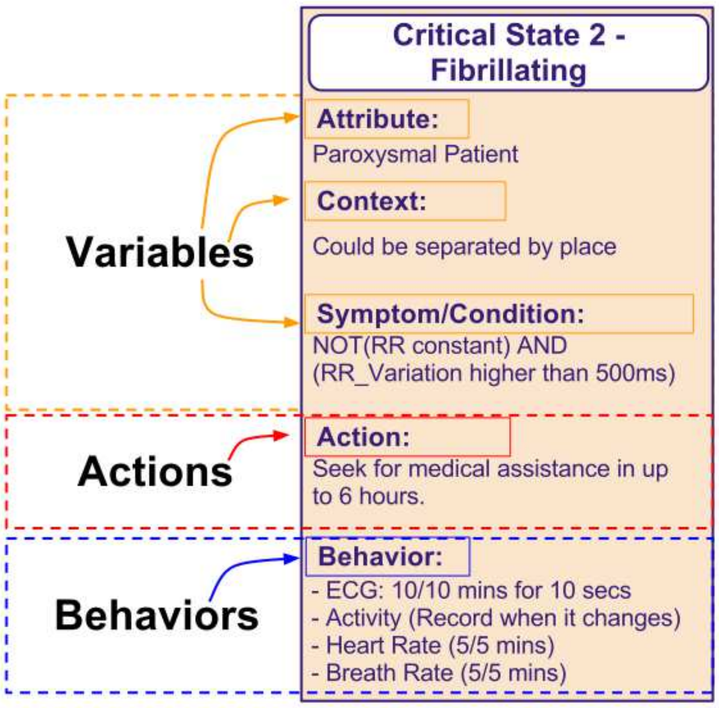

Critical State 2 - Fibrillating
Variables
Attribute:
Paroxysmal Patient
Context:
Could be separated by place
Symptom/Condition:
NOT(RR constant) AND (RR_Variation higher than 500ms)
Actions
Action:
Seek for medical assistance in up to 6 hours.
Behaviors
Behavior:
- ECG: 10/10 mins for 10 secs
- Activity (Record when it changes)
- Heart Rate (5/5 mins)
- Breath Rate (5/5 mins)

Figure 4.9: *Critical State 2* from the set of *states* defined in the atrial fibrillation example application developed for this thesis.

The *variables* are represented in the figure 4.9 above with the orange color. They define if the patient is consider to be in the *Critical State 2* or not. If all of the *variables* are evaluated as true, it is considered that the patient current situation or activity is the one represented by the *variable's state.* When any *variable* is evaluated as true, it asserts a minor situation about the monitored patient. For example, if his difference between R waves in an electrocardiogram is varying more than 500 milliseconds, if he is doing any physical activity or even if he is located in a place with altitude higher than 2400 meters.

The vocabulary for representing the *variables* are represented by the OWL classes: **AttributeVariable**, **ContextVariable**, **ConditionVariable** and **ComplexVariable**.

Attribute, Context, Condition and Complex variables

1. Attribute variable

It is represented by the class **AttributeVariable** and is designed for representing static context information about the patient. When we say static, it implies that this context information will not be modified by the application during the monitoring process. This context information can only be modified by the ontology engineer that customize the set of ontologies of the patient.

Figure 4.9 shows an example of attribute variable: "The fact that the patient must be a paroxysmal patient for this variable become true". Only to explain why this is a static context, it is important to say that regarding the constancy of the fibrillations, there are two kinds of patients. One is the paroxysmal patient that only fibrillates in specific periods during the day and the other is the permanent patient, which fibrillates all the time.

Since this characteristic does not change frequently during the monitoring process, it is considered as a static context and is modelled as an attribute variable.

2. Context variable

It is represented by the class **ContextVariable** and is designed to address conditions that belong to the environment/surroundings of the patient that affects in some way the monitoring process. For example, his location, the temperature of the environment, the city that the patient is located. In some diseases, like asthma, the temperature of the environment can be a factor that contributes to trigger a crises in a patient. Even the city that the patient is located, can be a factor that implies whenever he has a health plan coverage or not. In figure 4.10 we have an example of a state that has a context variable.

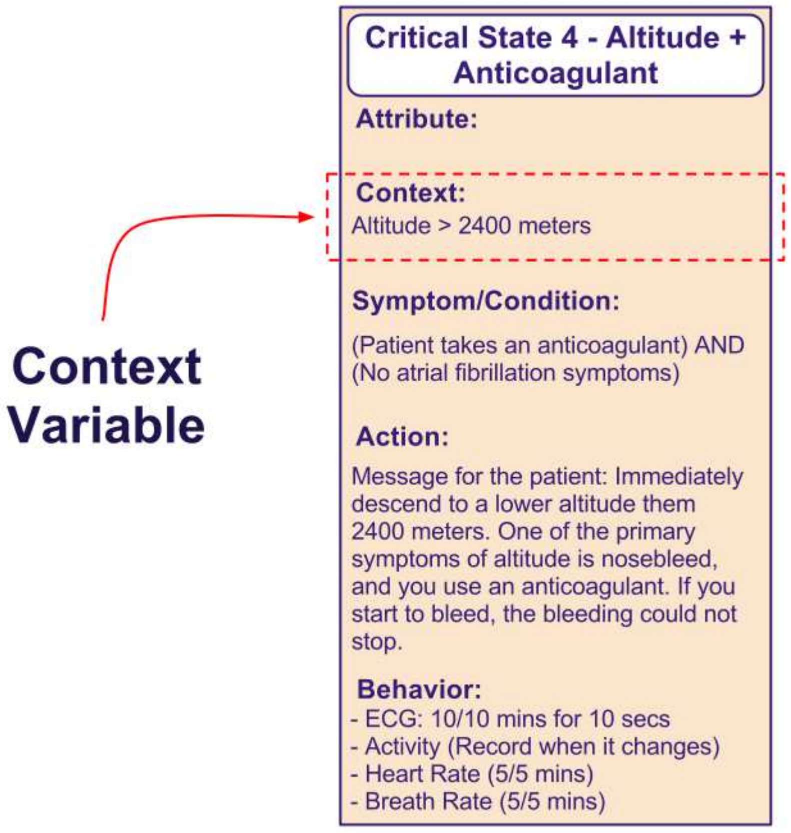

Critical State 4 - Altitude + Anticoagulant
Attribute:
Context:
Altitude > 2400 meters
Symptom/Condition:
(Patient takes an anticoagulant) AND (No atrial fibrillation symptoms)
Action:
Message for the patient: Immediately descend to a lower altitude them 2400 meters. One of the primary symptoms of altitude is nosebleed, and you use an anticoagulant. If you start to bleed, the bleeding could not stop.
Behavior:
- ECG: 10/10 mins for 10 secs
- Activity (Record when it changes)
- Heart Rate (5/5 mins)
- Breath Rate (5/5 mins)
Context Variable

Figure 4.10: *Critical State 4* - Altitude + Anticoagulant from the set of states defined in the atrial fibrillation example application developed for this thesis.

In this example, the context variable defines a minor situation that has to be monitored, the situation where the patient is located in a region that has an altitude greater then 2400 meters.

3. Condition variable

It is by the class **ConditionVariable**and it addresses the conditions that come from within the patient (i.e.: his body). In the example of the figure 4.10 there are two conditions variables. One regarding the fact that the patient uses anticoagulant, so he/she has an uncoagulated blood. The other one is about his atrial fibrillation symptoms (e.g.: Unusual tiredness, heart palpitation and lack of air) they are all conditions that come from the within the body of the patient and not from the environment.

4. Complex variable

It is represented by the class **ComplexVariable** and it have two subclasses, the *Or_ComplexVariables* and *And_ComplexVariables* classes and were designed to make more elaborated expressions by composing all the other variables presented so far (Attribute, Context, Condition and even other Complex variables). So far, by using the other types of variables, it is impossible to build expressions like "Only make this variable true if *expression A* is true or if *expression B* is true.".

To illustrate the usage of this class, we can pick the *Critical State 3* state showed in figure 4.11 to use as an example.

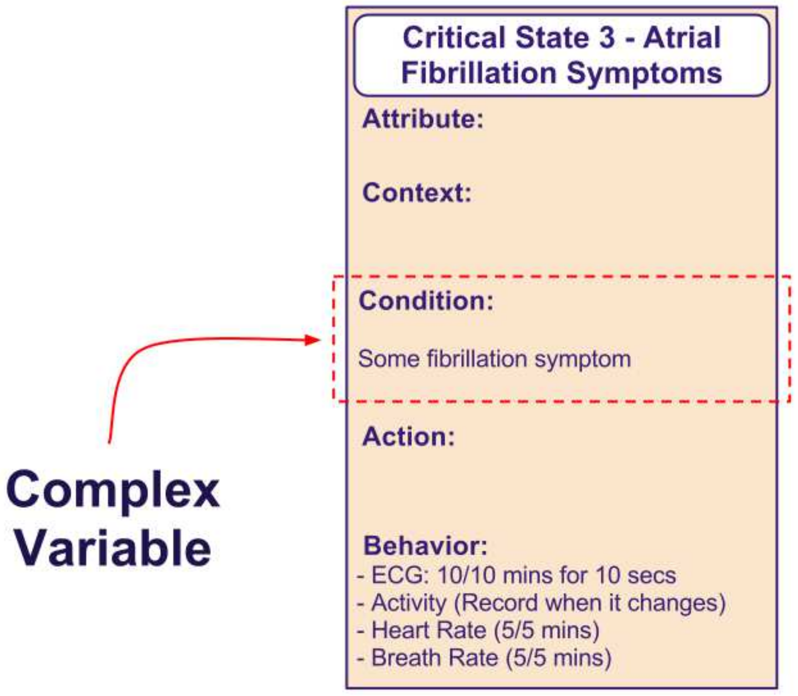

Critical State 3 - Atrial Fibrillation Symptoms
Attribute:
Context:
Condition:
Some fibrillation symptom
Action:
Behavior:
- ECG: 10/10 mins for 10 secs
- Activity (Record when it changes)
- Heart Rate (5/5 mins)
- Breath Rate (5/5 mins)
Complex Variable

Figure 4.11: *Critical State 3 - Atrial Fibrillation Symptoms* from the set of states defined in the atrial fibrillation example application developed for this thesis.

In the example, the doctor wants to consider a patient in a critical situation, if he or she feels any of the symptoms provoked by the atrial fibrillation disease. All the other variables classes presented so far are only capable to make an assertion about only one condition. Since each of the symptoms of atrial fibrillation are represented as a different condition, to be possible to the model express this situation, we have to give it some other capability, for example a way to express the OR operator.

Variables can be composed by using the *Or_ComplexVariable* class to assemble them using the OR operator. As in a similar way, the ontology engineer can assemble variables using the AND operator using the *And_ComplexVariable* class.

The OR and AND operators (represented by the classes *Or_ComplexVariable* and *And_ComplexVariable* respectively) can be seen as n-ary functions that all arguments can be either an attribute, context, condition variable or another complex variable.

In a formal way, any instance of the classes *Or_ComplexVariable* or *And_ComplexVariable* is a n-ary function where:

$$VAL = \{true, false\} \tag{4-1}$$

$$\begin{aligned} V = \{&ContextVariable\ instance, AttributeVariable\ instance,\\ &ConditionVariable\ instance, Or\_ComplexVariable\ instance,\\ &And\_ComplexVariable\ instance\}. \end{aligned} \tag{4-2}$$

OR is a n-ary function that we call OR operator, and is represented in the monitoring ontology by the *Or_ComplexVariable* class.

$$\mathbf{OR} : (x_1, ..., x_n) \rightarrow VAL, where\, x_i \in V. \tag{4-3}$$

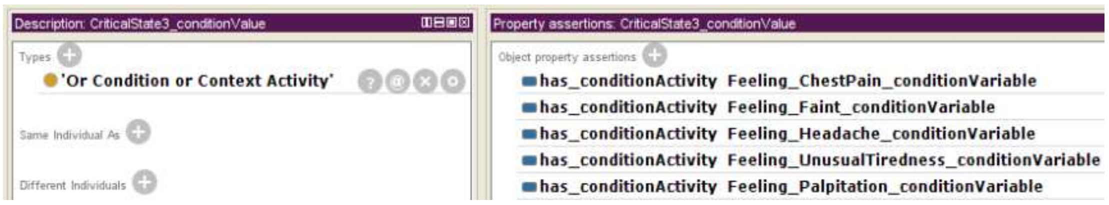
It returns true if any of the x_i returns true, otherwise it returns false.

AND is a n-ary function that we call AND operator, and is represented in the monitoring ontology by the *And_ComplexVariable* class.

$$\mathbf{AND} : (x_1, ..., x_n) \rightarrow VAL, where\, x_i \in V. \tag{4-4}$$

It returns false if any x_i returns false, otherwise it returns true.

Figure 4.12 shows how the ontology engineer can represent a situation that the doctor wants to consider a patient in a critical state, only if he or she feels any of the symptoms provoked by the atrial fibrillation disease. In order to express this situation, the ontology engineer must use an instance of the class *Or_ComplexVariable*.



Description: CriticalState3_conditionValue

Types
'Or Condition or Context Activity'

Same Individual As

Different Individuals

Property assertions: CriticalState3_conditionValue

Object property assertions
has_conditionActivity Feeling_ChestPain_conditionVariable
has_conditionActivity Feeling_Faint_conditionVariable
has_conditionActivity Feeling_Headache_conditionVariable
has_conditionActivity Feeling_UnusualTiredness_conditionVariable
has_conditionActivity Feeling_Palpitation_conditionVariable

Figure 4.12: Complex variable example.

The instance **CriticalState3_conditionValue** of the class *Or_ComplexVariable* represented in figure 4.12 uses the object property **has_conditionActivity** to associate other variables to it. The **has_conditionActivity** property can associate a complex variable to an attribute, context, condition variable or even to another complex variable.

Besides the variables as showed in figure 4.9, states can have *actions* and *behaviours*.

5. Actions and Behaviours

Whenever a patient is considered to be in a state, all the linked actions and behaviours are executed and applied. The concepts of action and behaviours can be differenced by its disturbance level. Actions are interferences that the system will apply, which will cause some disturbance to the patient or to the doctor (e.g.: Send an alert to the patient making his smart phone vibrates.). On the other hand behaviours are not disturbance, they are interferences that can be done without the patient even knowing that it happens (e.g.: As a result of the situation that the patient is in an altitude higher than 2400 meters, the system could configure itself to monitor more frequently the patient's breath rate.). Other examples of behaviours can be viewed in figure (Critical State 3), where the system changes its frequency of monitoring the activity, heart rate, breath rate and electrocardiogram signals when it detects some atrial fibrillation symptom.

The Actions can be divided in:

A Send an alert to the patient.

B Send a specific question to the patient.

C Send a SMS message to any phone number preconfigured within this Action, and to any number of recipients.

In (A), the doctor simply defines a message to be sent to the patient whenever the patient is considered to be in a specific state/situation. To give an example of it, we can take a look to the figure 4.10 where the doctor wants to alert his patient about the risks of the combination of altitude and being bearer of an anticoagulated blood.

The Action type (B) can be used if during any of the states, the doctor wants to make some question to the patient regarding some information that he wants to know. For example, if the doctor wants to ask the patient whatever he was doing when his heart rate passed 145 beats per minute.

And finally, the action type (C) can be used for example whenever the doctor wants to be warned about some situation that his patient is in. The action described in Figure 4.9 which the application sends a SMS to the doctor warning him about the fibrillating condition of his patient is one example of (C).

The behaviours are a way that the doctor can configure the frequency that the information is sent to be persisted in the web server. The possible configurations are:

A The period that some condition will be registered and sent to the server.

B The time for recording one measure of the condition (e.g.: whenever record electrocardiogram signals information, record during 10 seconds).

C Record the information whenever it changes. (e.g.: Whenever the information of activity of the patient is changed.) .

To have more detailed information about each of this elements that compose the concept of state, you can address the appendices where we show all the information contained in the *Monitoring Ontology*.

5
The Patient-Buddy-Build Framework

Before explaining how the PBB framework was developed, it is important to highlight that the focus of this thesis is to purpose a modeling approach for representing context information, and the PBB framework described in this chapter was intended only to serve as a prove of concept of our purposed modeling approach for representing context information. As so, it is not a complete and readily usable mobile application ready to be used in a real-life scenario.

The PBB framework aims at the generation of context-aware mobile client-server applications, where each mobile application is customized and than used for monitoring a patient. In order to give more details about how the PBB framework is implemented, figure 5.1 shows it‘s architecture.

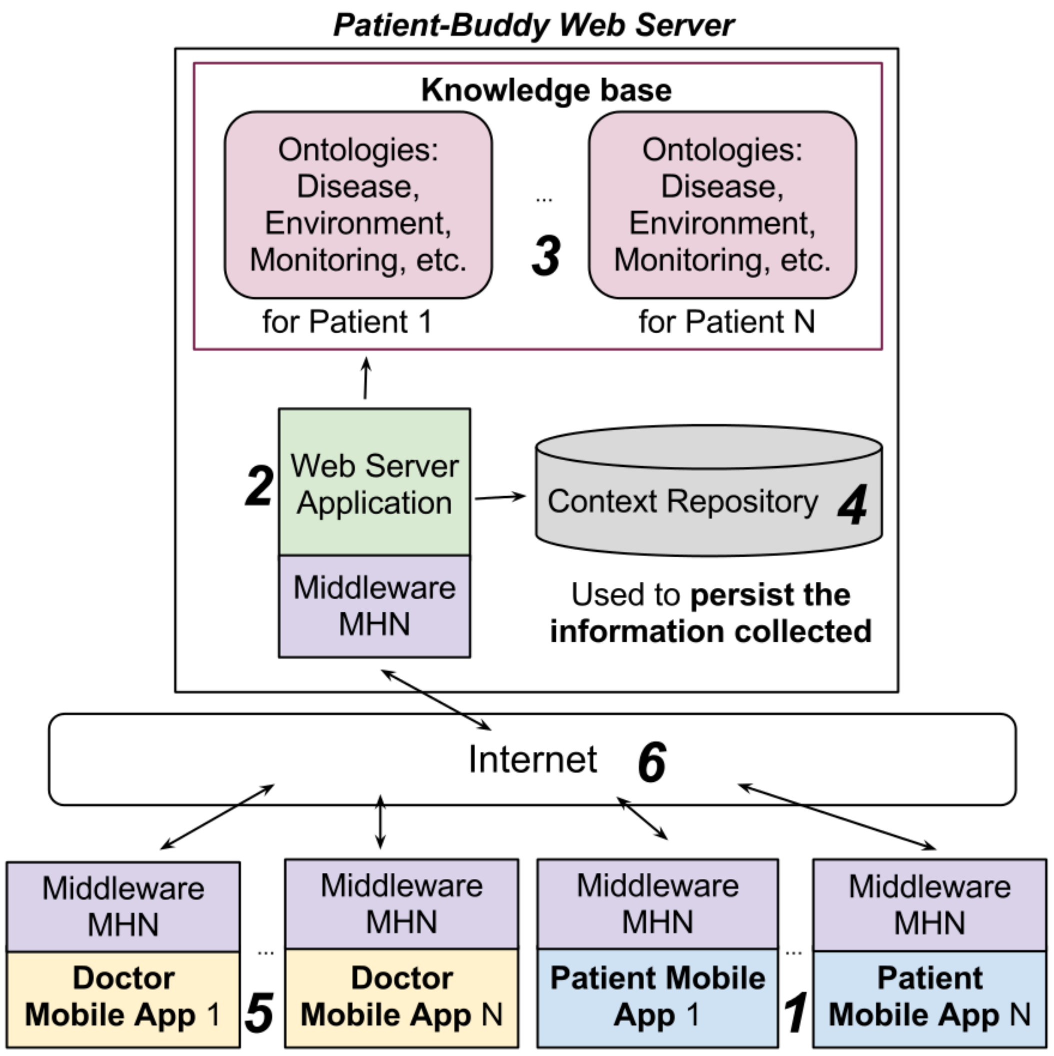


Figure 5.1: General overview of the PBB framework architecture.

The architecture has the following components:

1. **Patient mobile application (Represented by the *number 1*):** The mobile application that will be executing in the patient's smart phone. It will be further explained in the section 5.5.

2. **Web server (Represented by the *number 2*):** The web server is responsible of persisting all the monitored information, and generating the monitoring configurations for each of the patient's mobile applications. When we say "send monitoring configuration", we meant to send a specific part of the monitoring process that only related to the context that the patient is in. For example, if the patient location is not considered to have a high altitude it is not necessary to send to the mobile phone the part of the monitoring process related to altitude. It will be further explained in the section 5.1.

3. **Knowledge base (Represented by the *number 3*):** It centers all the domain information about the patients. It is where all the contextualized ontologies of the patients are persisted. For example, this knowledge base contains ontologies that represent the disease, the monitoring process and personal information about the patient.

4. **The Context Repository (Represented by the *number 4*):** It is used to persist all the monitored information collected. The patient's mobile applications send the monitored conditions such as the heart rate or the breath rate to be persisted. The doctor mobile application send messages to the patient and this messages are persisted using this context repository. It will be further explained in the subsection 5.1.1.

5. **Doctor mobile application(Represented by the *number 5*):** The mobile application that will be executing in the smart phone of the doctor. With this application, the doctor can send messages to his/her patients. The doctor can also request for text reports about the monitoring process of any of his patient's. The text will describe all the information collected from a patient and why the application made the decisions of collecting it. For example, if the application says that a patient was doing a physical activity during some time, it will also explain why the application considered that the patient was doing a physical activity (based in what parameters).

6. **Mobile Health Net Middleware (Represented by the *number 6*):** The middleware used for communication between mobile applications and the server. This middleware implements a publish/subscribe communication that the mobile applications can use to send and receive messages to the web server. The development of this middleware was not within the scope of this work, therefore it will not be explained in this thesis.

It is important to say that all the applications (**Patient Mobile App**, **Doctor Mobile App** and the **Web Server Application**) were only tested separately because of some issues that we had with the license of the middleware used to connect them all. However, all the functionalities of each applications were tested separately and they all were designed to work together. We tested the exchange of messages among applications by generating a copy of the messages from the senders applications inside each of the receiver applications.

5.1
Patient-Buddy Web Server

The objective of the Patient-Buddy Web Server (PBWS) is:

– Persist all the context data monitored from the patient's mobile devices as well as all the ontologies that contains the information related to the patients (e.g., his/her diseases, monitoring process, personal data and their medical data).

– Send the monitoring configuration files to each of the patient's mobile devices every time it is requested by them.

– A bridge that enables the communication between the doctor's mobile device and the patient's mobile device.

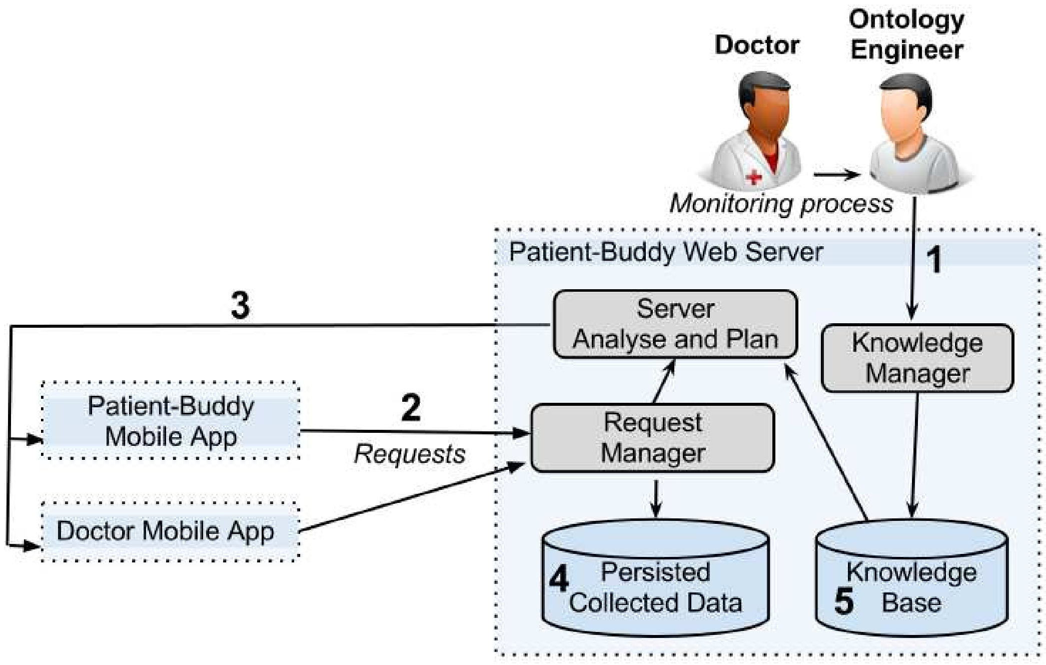


Figure 5.2: Patient-Buddy Web Server.

The figure 5.2 shows more details about how the Patient-Buddy Web Server was implemented and structured.

The knowledge about the monitoring process of a patient is entered at the server by an ontology engineer using ontologies. In the prototype application created to illustrate the usage of this framework, Protégé was the tool used to customize the ontologies. Protégé[1] is a open-source platform with a suite of functionalities to construct domain models and knowledge-based applications

[1]Protégé: `http://protege.stanford.edu/`

based on ontologies. The customization of the ontologies is done based in what was explained in chapter 4.

Using the ontologies, the ontology engineer's main task is to customize them according to the preferences and the monitoring process defined by the doctor (see *number 1*). For example, by customizing the Disease ontology represented in figure 4.1, the ontology engineer can specify the conditions that can be monitored from the patient by the PBB framework and to which diseases they are related.

Once the customization of the ontologies is completed, the ontology engineer starts the Patient-Buddy Web Server application, and using a browser of his choice he imports into the server Knowledge Base (*number 5*) the group of ontologies (customized from the ontologies listed in section 3.4) and the smart phone number that will be used to monitor the patient.

The customized ontologies will only be imported to the Knowledge Base if they passed through a consistency check. As explained in chapter 4, all the ontologies have some assertions made using description logic with the objective to help the ontology engineer on the task of customizing them.

These ontologies are persisted in the Knowledge Base using OWL-API (20). We chose the OWL-API because it is a JAVA library mainly designed to be used with and to manipulate OWL ontologies. The OWL-API is also used by several other works that involves the manipulation and the use of reasoners with OWL ontologies (including the Protégé tool) (5, 21).

When the patient turns on his phone for the first time (just when the PBMA application was installed), the application will send a request to the PBWS for the initial configuration of the monitoring process (*number 2*). The initial configuration corresponds to the configuration related to the initial state. The **Request Manager** interprets the request and begins the preparation of the monitoring process to be sent to the mobile application (PBMA). The preparation consists on send to the mobile device only the partition of the monitoring process related to the current state of the patient.

Regarding the set of ontologies that describe the entire monitoring process, there is a lot of information that PBMA does not need to know depending on the context in which it is inserted. For example, let's consider the state machine that defines the monitoring process of the example application developed for this work. The states that concern the effects of the high altitude do not need to be sent to PBMA, until it's user (the patient) is located in such context.

When PBMA requests for the patient's monitoring process, the component **"Server, Analyse and Plan"**, processes this request. The mobile application (PBMA) sends to the server (PBWS) the new situation that PBMA considers the patient to be in (the new situation corresponds to the new state of the monitoring process that the patient is in. If the PBMA was executing for the first time, then it send a request to PBWS for the initial configuration of the patient. If the mobile application (PBMA) requests for the initial configuration, the monitoring configuration is sent to the mobile device concerning the patient is in the normal state context. After receiving the state that the patient is inserted, the server (PBWS) sends to the PBMA the appropriate part of information related to the monitoring process.

In order to illustrate how the server selects a part of the monitoring process depending on the patient's state, figure 5.3 shows a conceptual representation of the possible transitions from the *Normal State* (the initial state) that the example application developed for this thesis contains.

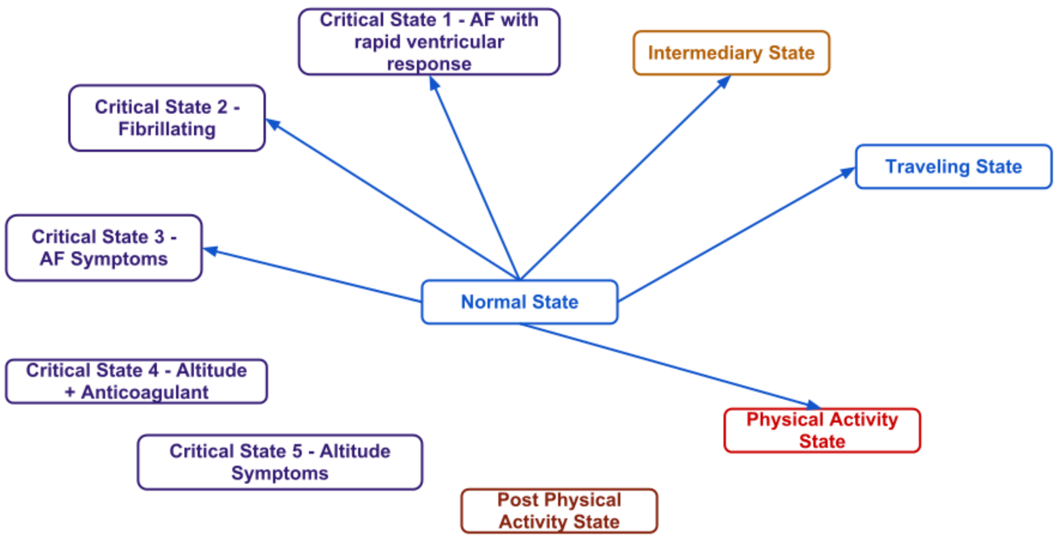

Critical State 1 - AF with rapid ventricular response
Intermediary State
Critical State 2 - Fibrillating
Traveling State
Critical State 3 - AF Symptoms
Normal State
Critical State 4 - Altitude + Anticoagulant
Physical Activity State
Critical State 5 - Altitude Symptoms
Post Physical Activity State

Figure 5.3: States reachable by the *Normal State*. All the states was created as part of the the monitoring process for the prototype application developed in this work. The prototype application will be further explained in chapter 6.

In the monitoring process, the states simply represents a possible situation that the patient can be inserted. By using their transitions, the server application (PBWS) can define the exact part of the monitoring process to be sent to the mobile application.

When the mobile application requests for its initial configuration, it will be updated with the monitoring process in the context of the *Normal State*. Thus, the part of the monitoring process sent to the mobile device will be only the information related to the states reachable by the current state of the patient (in the example of the figure 5.3, when sending the initial configuration, the current state is the *Normal State*).

To send the set of states to the mobile application, first the Server application (PBWS) reads this information in OWL and converts it to JAVA classes. These classes are created representing each of the OWL classes of the ontologies inserted in the knowledge base. The OWL format carries much metadata information about its entities and the OWL format is not suited to be processable by mobile devices, due to the huge amount of memory required. Therefore the part of the ontology sent to the mobile client (PBMA) had to be reduced. Thus we convert the information contained in the OWL ontologies to the JAVA classes, that were created with the propose of containing only the essential data to be sent to the mobile device. Finally, this set of information is converted from JAVA classes to the JSON format, which is then sent to the mobile device.

This procedure was created to facilitate the handling and manipulation of the set of ontologies that defines the remote monitoring by the patient.

Also, every state has an associate questionnaire that is generated automatically depending on the state‘s possible transitions. This questionnaire is generated in the Server (PBWS) and is sent to the patient mobile application (PBMA), the questionnaire is only generated for the current state of the patient. A questionnaire is generated by the PBWS using the **Server Analyse and Plan**, by the following steps:

1. Identify all the states that can be reached by the current state. Let‘s put these states together with the current state in a set called **TargetStates**.

2. For all states of the **TargetStates** set, the **Server Analyse and Plan** will identify all the state variables that are related to a condition that must be acquired by using a questionnaire. Let‘s put these state variables in a set called **TargetStateVariables**.

3. For all conditions related to each of the state variables contained in the **TargetStateVariables** set, the **Server Analyse and Plan** will use the environment ontology to see to what questions each of these conditions are related. This set of questions forms the questionnaire and it is associated to the current state of the patient.

The **Server Analyse and Plan** also generates reports about the patients and send to the Doctor Mobile App whenever the doctor requests for it. At this stage of the development, the doctor can only request only for day reports of his/her patients.

As a consequence that the monitoring process is represented in the PBB framework in the form of an automata and represented in a semantic way (4.4), when a report is requested from a patient by his doctor, the PBB framework has the capability of explaining why and how each information was collected from the patient. The PBB framework always explains why it made the decision to change the state of a patient, for example, from *Normal State* to *Physical Activity State*.

In the health sector, an explanation about how the machine makes each of it‘s decisions is very important to help in the diagnosis of the doctor. That was one of the reasons which made us to choose the implementation using automata and not choosing, for example, genetic algorithms or neural networks to be part of the decision system. Because when using them, when a decision is made, it is not possible to know the exactly steps that made that decision appears.

These are some examples of sentences that this capability of self explanation can produce to the doctor about the monitoring of his patient: (a) why the program was monitoring certain information; (b) why the patient was considered to be in a critical state; (c) why the program decides that the patient was doing physical activity, based on what information and what was the quality of that information (e.g.: the precision of the sensor that measured it).

In PBB framework, four types of explanations were implemented:

1. About data originated from the behaviour of a state.
2. About whenever a state became the new current state.
3. About data originated from the attribute, context and condition variables.
4. About data originated from complex variables.

The first explanation type is about the data originated from the behaviour of a state. It addresses the conditions that are acquired because of what was defined in the behaviour of a state.

Here is an example of this type of explanation that was generated by PBB, in this explanation the state **Normal State** was configured to record the heart rate data every 6 hours. This state is part of the set of the states created to the monitoring process of atrial fibrillation, which is the disease that our example application was developed to monitor.

“The heart rate value was 98,0 beats per minute (bpm). This value was acquired in: Sep 17, 2013 12:27:44 AM. This information

> was registered because of the behaviour that was configured for the Normal State. Every 360.0 minutes the heart rate value is registered. The value 98.0 beats per minute (bpm) was acquired automatically from the sensor Zephyr and it has a precision error of 5.0 beats per minute (bpm)."

The second explanation type is about whenever a state became the new current state. It explains why the application made that decision to consider the patient in a new current state. This is an example of explanation generated by PBB that concerns why the system considered a patient in the **Normal State**.

> "The patient was considered in the Normal State from the moment Sep 25, 2013 1:23:37 AM. The patient was considered to be in the Normal State because he was not feeling any of the symptoms defined by the doctor. This information was registered to record the moment when the Normal State was considered the new current state of the patient."

The third explanation type is about the data originated from every attribute, context and condition variables. Every time that a new value of a condition is acquired, the PBB checks if the current state of the patient will change or not. To do this, PBB evaluates every variable of every state that can be reachable by the current state of the patient. So every time that a attribute, context or condition variable is evaluated as true, the condition associated to it is persisted and a explanation is generated.

For example, we simulate PBB generating an explanation about when a context variable was evaluated as true. This context variable is evaluated as true whenever the patient‘s location altitude is higher than 2400 meters. This context variable belongs to a state called **Intermediary State** and the current state of the patient is the **Normal State**. The **Normal State** can reach the **Intermediary State** and because of that the PBB evaluates the **Intermediary State**‘s context variable. The explanation generated by PBB was:

> "This information was registered because it is one of the requirements to the patient be considered in the Intermediary State. The Intermediary State is reachable from the current state of the patient (Normal State). The measured value of Altitude was 2459.0 meters. This requirement was configured to be considered true in

> the Intermediary State every time that the measured value of Altitude is greater or equal to 2400.0 meters. This measure was taken in Sep 25, 2013 1:23:36 AM. The value of 2459.0 meters was acquired automatically from the internal GPS sensor of the mobile device and it has a precision error of 100.0 meters.".

The fourth explanation type is about the data originated from every complex variable. It is similar to the third explanation type, the difference is that the fourth explanation is related when a complex variable is evaluated as true. An example of this type of explanation is bellow. The complex variable of the example is configured to be evaluated as true if the patient‘s heart rate is higher or equal than 100.0 beats per minute and if the patient‘s is not doing any physical activity.

> "This requirement is configured to be evaluated as true in the Critical State 1 whenever the following informations are considered as true: 1) The measured value of the patient‘s Activity was Not Moving. This requirement is configured to be registered and considered true in the Critical State 1 whenever the measured value of the patient‘s activity is equal to [Not Moving, Flat]. || 2) The measured value of the patient‘s heart rate was 121.0 beats per minute (bpm). This requirement is configured to be registered and considered true in the Critical State 1 whenever the measured value of the patient‘s heart rate is greater or equal than 100.0. ".

5.1.1 Context repository

Another functionality of the Patient-Buddy Web Server application is the persistence of the data monitored from the patient's mobile applications (PBMA). The main purpose of the context repository is to persist the history of all monitored data from the patients. This way the PBWS can generate reports based on all data monitored from the patients, and deliver it to the doctor.

In the PBB framework, all the context information are persisted using ontologies. We choose ontologies to persist the collected context information because all the domain information related to the patient, the disease and the monitoring process are described through ontologies.

The chosen database was the Virtuoso triple store [2] and all the information are persisted in the RDF (Resource Description Framework) format. The

[2]Virtuoso RDF Graph Model Engine: `http://virtuoso.openlinksw.com/rdf-quad-store/`

Virtuoso triple store is widely adopted by other works and already proved to have a good evaluation on several benchmarks (8).

All the data monitored from the patient have the characteristic to be series data (multi-dimensional data). Each data monitored from the patient are related to at least a specific time that this information was acquired, a situation in which it was acquired, an error precision and a specific form by which it is acquired (e.g., by sensors).

To handle the representation of multi-dimensional data and to link this data to the other concepts of the ontologies of PBB, we choose to use the Data Cube[3] vocabulary, which is a RDF vocabulary supported by W3C[4].

The data cube model also helps to facilitate the query for the stored data because it can be seen through some measure of interest. When generating reports to the doctor, the PBB becomes a decision support system that must explain to the doctor each of the decisions made by the program, to support the doctor‘s decision on his/her evaluation about the patient.

In the Data Cube vocabulary, when representing series data, each of the recorded data in the sequence is called observation. An observation can be composed by three components: dimensions, attributes and measurements. The dimension component defines what the observation applies to (e.g., the time it was measured). The measurement component define the set of measured values (e.g., a heart rate value or a temperature of the patient's environment). And finally, the attribute component defines how the observation was expressed (e.g., units).

A data cube is composed by one or more data structures definitions[5]. There are four data structures defined in the Patient-Buddy-Build persistence model. The *Condition Measurement* data structure, the *Explication of State* data structure, the *Explication of State Variable* data structure and the *Explication of Complex State Variable* data structure.

Every time the patient's mobile application sends a context data to be persisted, it sends because of one of the four reasons listed bellow. Each of this reasons are represented using a different data structure.

Bellow we associate each created data structure with each of the reasons that the PBMA (patient mobile application) sends data to be persisted.

1. *Condition Measurement* data structure: Represents data that is associated with the frequency in which a monitored data is configurated to be persisted. The frequency in which a context data is persisted is dictated

[3]Data Cube vocabulary: `http://www.w3.org/TR/vocab-data-cube/`
[4]W3C (World Wide Web Consortium): `http://www.w3.org/`
[5]SMDX: `http://www.w3.org/TR/2013/CR-vocab-data-cube-20130625/#intro-sdmx`

by the states of the monitoring process (represented by an automata). For example, it is possible to define that if the patient is in the *normal state*, the heart rate only need to be persisted with intervals of 6 hours. In the monitoring state, the data related to the frequency that a context data will be persisted can be found in it's behavior. The behavior of a state dictates to the application what are the frequency that each of the context data will be persisted in the PBWS context repository. Figure 5.4 shows the structure of a state. The behavior is represented in the Conclusion part of the State because it is only applied when every State variable is evaluated as true.

2. *Explanation of State* data structure: Whenever the patient's monitoring state changes, the new current state is persisted in the PBWS context repository. This data structure purpose is to persist every state that became active during a monitoring of a patient. The explanation of a state will put together in a text why it was considered as a new current state. It will contain information related to all the context data associated with the state and the moment that it was activated.

3. *Explanation of State Variable* data structure: Every state have three variables, whenever a variable becomes true, PBMA sends the context data related to this variable to be persisted in the PBWS context repository. Figure 5.4 show the three variables of a State (illustrated in figure 5.4). It is important to highlight that a state is only persisted when all its variables are true. This form of persistence allows that every time that any variable is evaluated as true, it is send to the PBWS context repository to be persisted. For example, a monitoring process can create a monitoring state that represents a situation which the patient location is considered a high altitude. One variable of this state will check if the patient's location is higher then a certain limit, if the variable is true, the patient's location altitude is persisted.

4. *Explanation of Complex State Variable* data structure: Some variables of a state can be represented as Complex Variables. The Complex Variables contains one or more State Variables associated to it. Not all State Variables within a Complex Variable is evaluated as true. The state variables inserted by the doctor in the complex variable are only important if associated, and because of that they generate a different explanation to the doctor. Thus, this data structure is responsible to persist the Complex Variables.

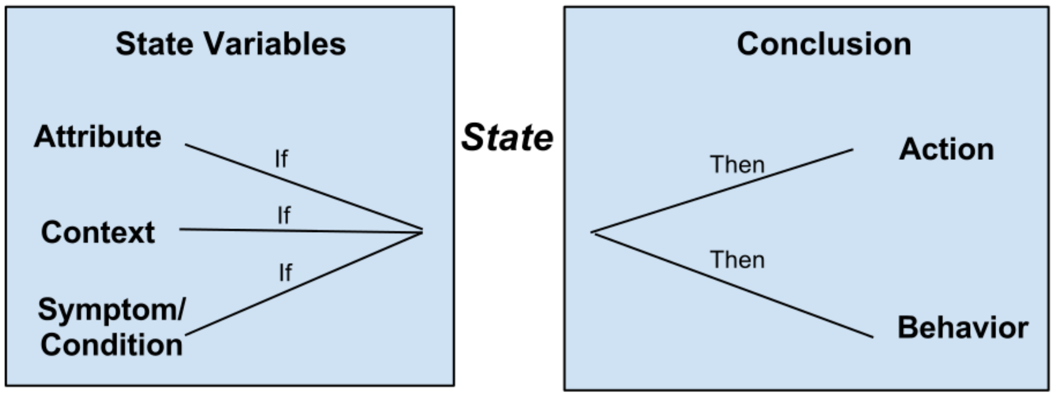


Figure 5.4: Structure of a State in the monitoring process.

All the details about the four data structures can be found in the appendices.

5.2 Patient-Buddy Mobile Application

The patient mobile application is responsible for collecting the data from the patient and to behavior according to the monitoring process defined by the doctor.

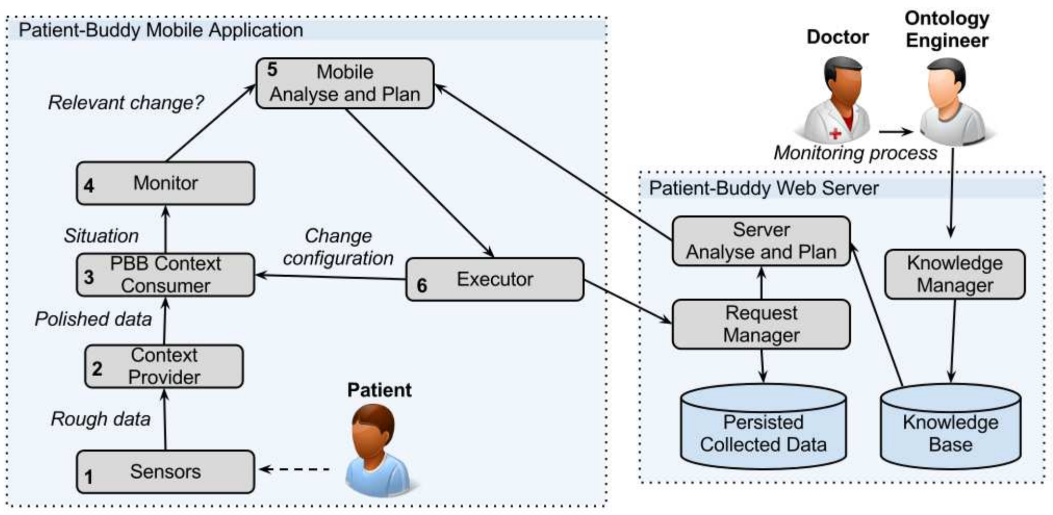


Figure 5.5: Patient Mobile Application.

Figure 5.5 represents how this application is organized. The block *number 1* (Sensors) is the block that retrieves context data from the patient and from the environment. It can be a sensor attached to the patient's body (e.g.: To measure the heart rate) or a web service that retrieves data from the internet (e.g.: Weather information about the patient's location).

Usually the data retrieved from sensors comes with a lot of information that the application will not use. Thus this data has to be processed in order to send to the application only the necessary data. For example, if the application only wants to know about the temperature of the patient's location. When using a web service to retrieve it, this information could come in a XML format and include other data like humidity and percentage of chances to rain. Therefore the context provider (*number 2* in figure 5.5) task is to send to the context consumer only the information that the application will use.

The PBB context consumer (*number 3* in figure 5.5) is responsible for relating the context information acquired from the patient with a condition defined in the disease ontology. By doing so, the user will say to the system what are each of the information produced by the sensors. The user relates the information with the condition defined in the disease ontology by using it's URI.

After the context information is related to a condition, it is sent to the monitor (represented by the *number 4* in figure 5.5). The **Monitor** is responsible for detecting when the new information is relevant to the system. For example, in order to change the monitoring state of a patient from **Normal State** to one of the critical states, the patient must be feeling at least one of the symptoms associated to the atrial fibrillation disease. Therefore whenever any of the symptoms related to atrial fibrillation is acquired as true, the **Monitor** must identify it as a relevant change.

The **Monitor** knows when a new context information of a patient is relevant according to what was defined within his/her monitoring ontology. All the states in the monitoring ontology represent situations that a patient can be detected in. As we said before, the monitoring states must have a set of assertions that define if the patient is considered to be in the state. This assertions are made by using the state variables, and are defined by expressions that can relate to one or more conditions (e.g.: heart rate, breath rate, patient's activity). The **Monitor** considers a new context information to be relevant whenever this new information causes a state variable to became true.

For example, considering a state (named *Fibrillating State*) that has two state variables:

1. The patient is not doing a physical activity.
2. Patient's heart rate is higher then 100 beats per minute.

This state variables define when an intermittent patient is fibrillating or not. Both variables are related to a condition, the first state variable is related to the patient‘s activity and the second is related to the his/her heart rate. Whenever the heart rate of a patient is higher then 100 beats per minute or the or the patient is doing a physical activity, the **Monitor** will consider it a relevant change.

After the relevant changes are computed by the *Monitor*, it is sent to the *Mobile Analyse and Plan* (represented by the *number 5* in figure 5.5). The *Mobile Analyse and Plan* records all the last relevant changes of each condition and it is responsible for detecting when there is a change in the monitoring state of the patient. The *Mobile Analyse and Plan* is also responsible to send relevant changes of context data to be persisted in the server (PBWS). For example, if the heart rate is higher then 100 beats per minute the *Mobile Analyse and Plan* will send it to be persisted in the server. If also the patient is not doing any physical activity, the *Mobile Analyse and Plan* will send a request to the *Executor* module to change the monitoring configuration to the new state (*Fibrillating State*).

The *Executor* module is represented by the *number 6* in figure 5.5 and is responsible for executing the orders sent by the *Mobile Analyse an Plan*. The orders can be:

– **Send a context data to be persisted in the server or in the mobile device.** (e.g.: Persist a measure of the patient's heart rate.)

– **Change the monitoring configuration of the mobile device.** (e.g.: Change the monitoring state from the *Normal state* to a *Patient is Fibrillating state*). Each state contains a set of actions and different behaviours, this will be further explained in section 4.4. The fact to persist the heart rate every 5 minutes is an example of a state behaviour. Changing a state can change the frequency which the heart rate is persisted or even do not monitor it any more.

– **Send an alert message to the patient's mobile device.** The atrial fibrillation example application developed within this thesis, sends an alert message to the patient whenever he/she is located in a place where the altitude is higher then 2400 meters and he/she has ancoagulated blood. The message warns the patient that if his/her nose starts to bleed (nose bleeding is a common symptom of the altitude), because of the fact that he/she uses an anticoagulant, the bleeding could not stop.

6 Patient-Buddy Context-aware Application for patients with Atrial Fibrillation

Using the PBB framework we have developed a context-aware application for remote monitoring patients with atrial fibrillation to prove the feasibility of the proposed context modeling approach,.

Atrial fibrillation (AF) is the most common type of arrhythmia. It is diagnosed when the atria and ventricles no longer beat in a coordinated way. The ventricles may beat 100 to a 175 times a minute, in contrast to the normal rate of 60 to 100 beats a minute. People who have atrial fibrillation may not feel symptoms. However, even when atrial fibrillation is not noticed, it can increase the risk of stroke. In some people, atrial fibrillation can cause chest pain or heart failure, especially if the heart rhythm is very rapid.

Remote monitoring for atrial fibrillation provides invaluable tools for diagnosis and management. There are two types of atrial fibrillation patients when it comes to the frequency the fibrillation occur: Paroxysmal and Permanent AF patients. Permanent patients are fibrillating all the time and the paroxysmal patients only fibrillate during random periods of the day. Early identification and treatment of paroxysmal AF episodes has the potential to reduce the progression to persistent or permanent AF. The earlier treatment using, for example, anticoagulant therapy can reduce the rate of strokes.

In order to remote monitor patients with AF, we used a sensor that is attached to the patient's chest. Figure 6.1 bellow is a photo of the sensor[1].

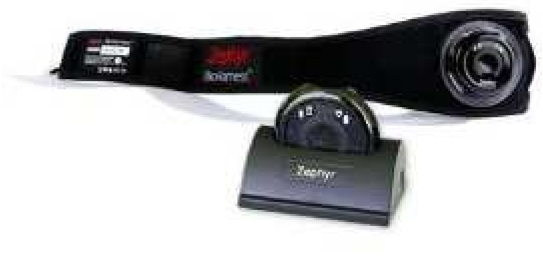


Figure 6.1: Sensor used for the example application.

[1]Zephyr sensor: http://www.zephyranywhere.com/products/bioharness-3/

This sensor have the capability of monitoring the following data:

– Electrocardiogram (ECG)

– Body Temperature

– Posture

– Heart Rate

– Breath Rate

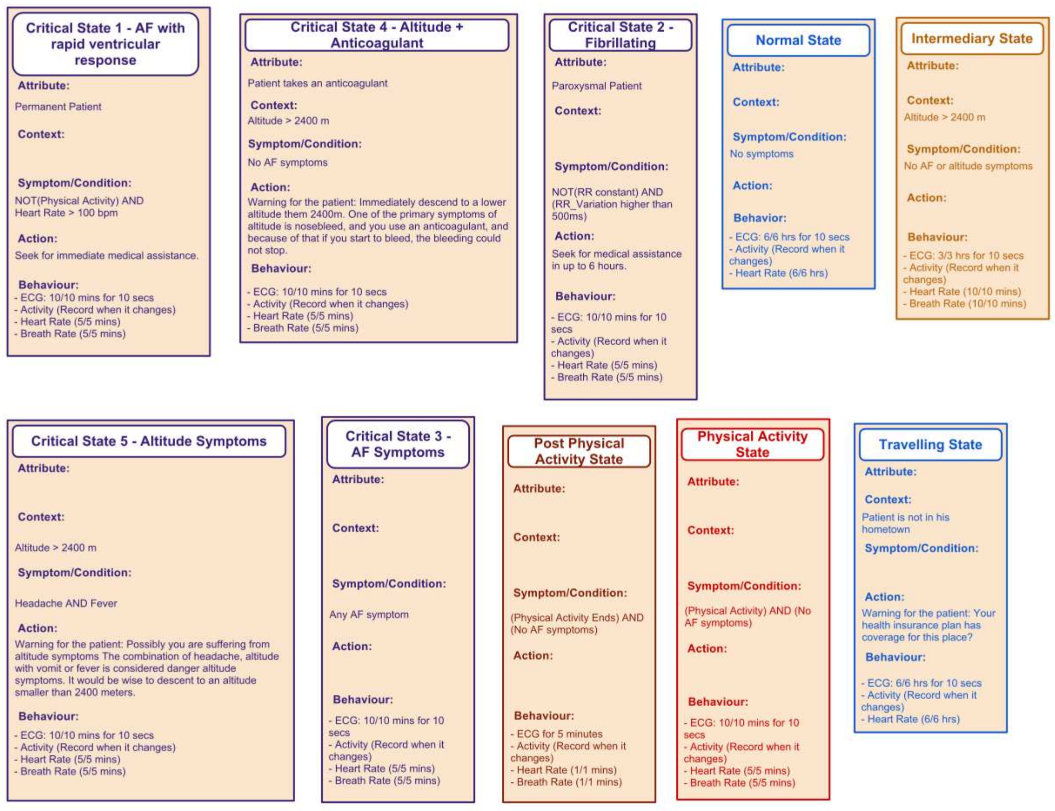
– Activity (using accelerometer)

The data of this sensor is sent to the patient's Android smart phone using a Bluetooth connection.

6.1 The Monitoring Process for Atrial Fibrillation patients

In order to define how a patient with AF would be remotely monitored (i.e. the monitoring process), we arrange several meetings with a specialist. The developed monitoring process is a joint work with doctor Bruno Azevedo, who is a doctoral candidate in Cardiology by Federal University of Rio de Janeiro (UFRJ) and Mechanical Engineer by Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio).

The monitoring process was developed having in mind the set of data that this sensor can retrieve from the patient, the possibility of using web services and the capability of making questions to the patient. Also, all the possible customization allowed for the monitoring process was generated by the meetings with Bruno Azevedo. Figure 6.2 shows all the states created for the monitoring process of a patient with atrial fibrillation.



Critical State 1 - AF with rapid ventricular response
Attribute:
Permanent Patient
Context:
Symptom/Condition:
NOT(Physical Activity) AND Heart Rate > 100 bpm
Action:
Seek for immediate medical assistance.
Behaviour:
- ECG: 10/10 mins for 10 secs
- Activity (Record when it changes)
- Heart Rate (5/5 mins)
- Breath Rate (5/5 mins)

Critical State 4 - Altitude + Anticoagulant
Attribute:
Patient takes an anticoagulant
Context:
Altitude > 2400 m
Symptom/Condition:
No AF symptoms
Action:
Warning for the patient: Immediately descend to a lower altitude them 2400m. One of the primary symptoms of altitude is nosebleed, and you use an anticoagulant, and because of that if you start to bleed, the bleeding could not stop.
Behaviour:
- ECG: 10/10 mins for 10 secs
- Activity (Record when it changes)
- Heart Rate (5/5 mins)
- Breath Rate (5/5 mins)

Critical State 2 - Fibrillating
Attribute:
Paroxysmal Patient
Context:
Symptom/Condition:
NOT(RR constant) AND (RR_Variation higher than 500ms)
Action:
Seek for medical assistance in up to 6 hours.
Behaviour:
- ECG: 10/10 mins for 10 secs
- Activity (Record when it changes)
- Heart Rate (5/5 mins)
- Breath Rate (5/5 mins)

Normal State
Attribute:
Context:
Symptom/Condition:
No symptoms
Action:
Behavior:
- ECG: 6/6 hrs for 10 secs
- Activity (Record when it changes)
- Heart Rate (6/6 hrs)

Intermediary State
Attribute:
Context:
Altitude > 2400 m
Symptom/Condition:
No AF or altitude symptoms
Action:
Behaviour:
- ECG: 3/3 hrs for 10 secs
- Activity (Record when it changes)
- Heart Rate (10/10 mins)
- Breath Rate (10/10 mins)

Critical State 5 - Altitude Symptoms
Attribute:
Context:
Altitude > 2400 m
Symptom/Condition:
Headache AND Fever
Action:
Warning for the patient: Possibly you are suffering from altitude symptoms The combination of headache, altitude with vomit or fever is considered danger altitude symptoms. It would be wise to descent to an altitude smaller than 2400 meters.
Behaviour:
- ECG: 10/10 mins for 10 secs
- Activity (Record when it changes)
- Heart Rate (5/5 mins)
- Breath Rate (5/5 mins)

Critical State 3 - AF Symptoms
Attribute:
Context:
Symptom/Condition:
Any AF symptom
Action:
Behaviour:
- ECG: 10/10 mins for 10 secs
- Activity (Record when it changes)
- Heart Rate (5/5 mins)
- Breath Rate (5/5 mins)

Post Physical Activity State
Attribute:
Context:
Symptom/Condition:
(Physical Activity Ends) AND (No AF symptoms)
Action:
Behaviour:
- ECG for 5 minutes
- Activity (Record when it changes)
- Heart Rate (1/1 mins)
- Breath Rate (1/1 mins)

Physical Activity State
Attribute:
Context:
Symptom/Condition:
(Physical Activity) AND (No AF symptoms)
Action:
Behaviour:
- ECG: 10/10 mins for 10 secs
- Activity (Record when it changes)
- Heart Rate (5/5 mins)
- Breath Rate (5/5 mins)

Travelling State
Attribute:
Context:
Patient is not in his hometown
Symptom/Condition:
Action:
Warning for the patient: Your health insurance plan has coverage for this place?
Behaviour:
- ECG: 6/6 hrs for 10 secs
- Activity (Record when it changes)
- Heart Rate (6/6 hrs)

Figure 6.2: Monitoring process implemented for patients with atrial fibrillation.

The states listed in figure 6.2 are:

- **Normal State**: is the initial state of the monitoring process. The initial state is set when the system is turning on to start the monitoring and it has no current information about the patient. States reachable by this state: **Physical Activity State**, **Traveling State**, **Intermediary State**, **Critical State 1**, **Critical State 2** and **Critical State 3**.
- **Physical Activity State**: Represents the situation where the patient is doing a physical activity. The difference here is the frequency in which some monitored information will be sent to the server to be persisted. States reachable by this state: **Post Physical Activity State**, **Critical State 1**, **Critical State 2** and **Critical State 3**.
- **Post Physical Activity State**: Represents the moment when the patient stop doing a physical activity. This state is only active for five minutes. States reachable by this state: **Physical Activity State**, **Normal State**, **Traveling State**, **Intermediary State**, **Critical State 1**, **Critical State 2** and **Critical State 3**.
- **Traveling State**: Represents a situation where the patient location is outside the city that he/she lives. The purpose of this state is only to

send a message to the patient with a warning about the coverage of his/her health insurance plan. States reachable by this state: **Physical Activity State**, **Normal State**, **Intermediary State**, **Critical State 1**, **Critical State 2** and **Critical State 3**.

- **Intermediary State**: Represents the situation where the patient's location altitude is higher than 2400 meters. States reachable by this state: **Physical Activity State**, **Normal State**, **Traveling State**, **Critical State 1**, **Critical State 2**, **Critical State 3**, **Critical State 4** and **Critical State 5**.
- **Critical State 1**: It is only reachable by permanent AF patients. Represents the situation where the patient if fibrillating with rapid ventricular response. This state becomes active when the patient is not doing a physical activity and his/her heart rate is greater than 100 beats per minute. The action related to this state is to send a message to the patient's doctor and to all the emergency contacts registered seeking for immediate medical assistance. States reachable by this state: **Physical Activity State**, **Intermediary State**, **Normal State**, **Traveling State** and**Critical State 3**.
- **Critical State 2**: It is only reachable by paroxysmal AF patients. Represents the situation where the patient is fibrillating. It is characterized when the electrocardiogram signal of the patient has $\boldsymbol{R}$ interval variation higher than 500 ms after a period of 56 ms. The 56 milliseconds period was chosen because it is the rate that the sensor sends ECG information to the mobile device. The figure 6.3 bellow illustrates the algorithm used for detecting fibrillation on paroxysmal AF patients. The action related to it is to send a message to the doctor and the patient's emergency contacts seeking for medical assistance in up to 6 hours. States reachable by this state: **Physical Activity State**, **Intermediary State**, **Normal State**, **Traveling State** and**Critical State 3**.

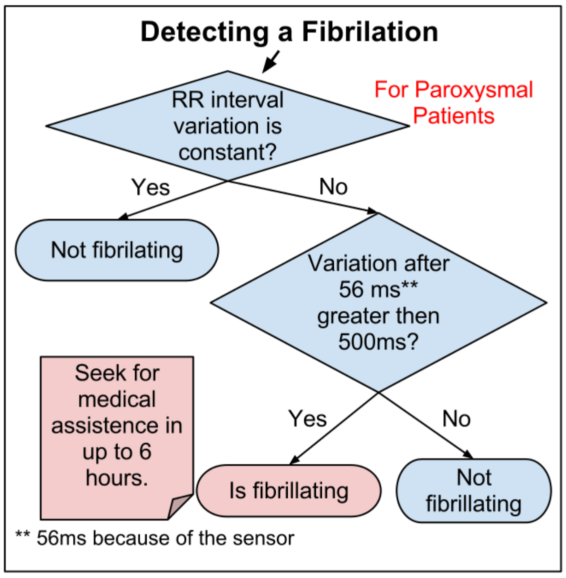

Detecting a Fibrilation
RR interval variation is constant?
For Paroxysmal Patients
Yes
No
Not fibrilating
Variation after 56 ms** greater then 500ms?
Seek for medical assistence in up to 6 hours.
Yes
No
Is fibrillating
Not fibrillating
** 56ms because of the sensor

Figure 6.3: Detecting fibrillation on paroxysmal patients.

– **Critical State 3**: It represents the situation where the patient is feeling any AF symptoms. For this implemented monitoring process, all the questions that can be made to ask the patient about AF symptoms are accessible by pressing a button. The objective of this button is to give the possibility to the patient for reporting a symptom. States reachable by this state: **Physical Activity State**, **Intermediary State**, **Normal State**, **Traveling State**, **Critical State 1** and **Critical State 2**.

– **Critical State 4**: This state is only reachable by the intermediary state. Represents the situation where the patient uses an anticoagulant medication and is located in a place with altitude higher than 2400 meters. The action related to it is to send a warning to the patient telling him/her about the dangerous of the combination of high altitude and not coagulated blood. States reachable by this state: **Physical Activity State**, **Intermediary State**, **Normal State**, **Traveling State**, **Critical State 1**, **Critical State 2**, **Critical State 3** and **Critical State 5**.

– **Critical State 5**: It is only reachable by the intermediary state. Represents the situation where the patient's altitude is higher than 2400 meters, does not have not coagulated blood and is feeling fever. States reachable by this state: **Physical Activity State**, **Intermediary State**, **Normal State**, **Traveling State**, **Critical State 1**, **Critical State 2**, **Critical State 3** and **Critical State 4**.

6.2 Implemented Context Providers

To be able to retrieve all the context information necessary for the monitoring of patients with atrial fibrillation, the following context providers were implemented:

– **Activity Context Provider**: The determination of a person's activity is a very complicated issue. Since the focus of this work is not the detection a person's activity, the algorithm implemented in the PBB framework is not to be used for real cases. The activity detection here is only to make possible the implementation of a prototype to test our context information representation model. The conditions related to this context consumer (defined on the disease ontology) is each of the activities that the patient is doing and can be determined using the accelerometer and the patient's body inclination. The activity context consumer determines if the patient is in a flat position, standing up, walking, walking hard, running or running hard. Because of this, the activity context consumer must have some intelligence to generate the activity information based on the accelerometer and inclination data acquired from the sensor. It also determines if the patient is doing a physical activity, which is important to better understanding, for example, the heart rate variation during short periods of time. Concerning the atrial fibrillation disease, the heart rate parameter is important. It can be used to determine if a permanent patient is fibrillating with rapid ventricular response. To make this diagnosis, it is important to know if the patient's heart rate is high because he is doing a physical activity. The flat position is determined when the patient inclination is near 90% (between 70% - 100%) and there is speed registered in any of the three dimensions of the accelerometer. The standing up position is when the inclination of the patient is near 0% and he is also not moving. The walking activity is determined when the accelerometer speed is between 0.5km/h and 7 km/h. If the speed is higher than 7 km/h, the person is running.

– **Body Temperature Context Provider**: The information related to this context consumer is the body temperature. Based on the temperature collected by this provider, the system is able to determine if the patient has a fever or not.

– **Electrocardiogram Context Provider**: This context provider is responsible for acquiring information from the electrocardiogram (ECG) signal. It process the electrocardiogram signal using simple algorithms, since the focus of this thesis is not in ECG signal processing. The sensor used for this example application provides information about the RR interval, the ST segment and the PQ segment (showed in the figure 6.4 bellow).

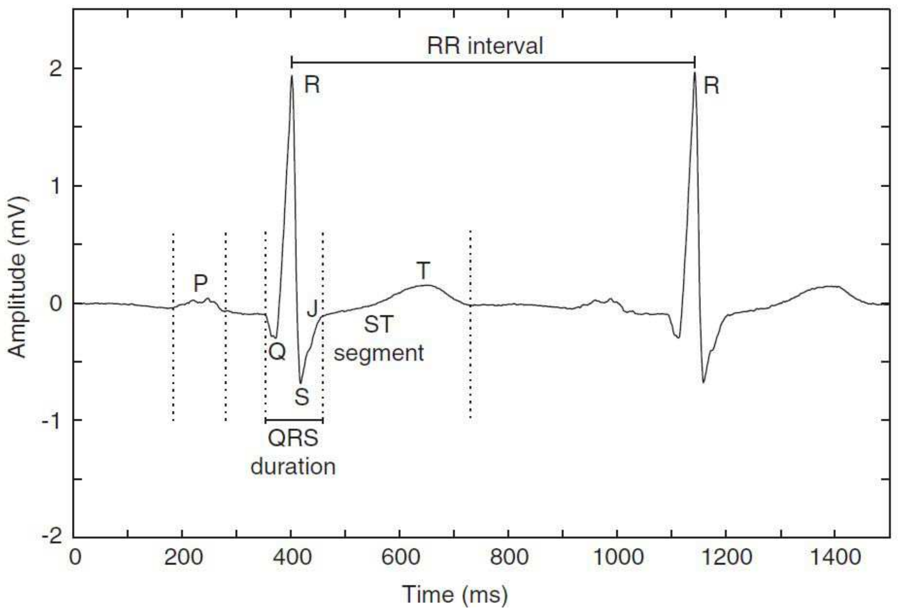


Figure 6.4: Data monitored from the ECG signal.

The RR interval is used to determine if a patient is fibrillating or not. The ST and PQ segment are only sent to the server to be persisted whenever the ECG information is recorded.

- **Breath Rate Context Providers**: It is a simple context provider that knows how to receive the breath rate information from the sensor attached to the patient.
- **Heart Rate Context Provider**: Is a simple context provider that knows how to connect to the sensor attached to the patient and to retrieve the heart rate information.
- **Location Context Provider**: It is a context provider that use the GPS from the smart phone to retrieve latitude, longitude and altitude information about the patient's location. It also provide information about which city and country the patient is current located.

7
Evaluating the proposed model

7.1
The Hybrid Patient-Buddy context information model

In the literature, besides the models listed in the chapter 2.3, there is another approach that can be used for representation of context information. It is called hybrid models, and was developed as a response to the fact that all context modeling approaches described in chapter 2.3 have advantages and disadvantages when it comes to covering requirements for context representation models. Several authors (6, 27, 44, 34, 7) have identified that, even though each approach may provide an effective solution for a particular domain, none of them seems to provide a good solution to all the requirements(chapter 2.3). Hybrid models try to combine different approaches in order to reduce their disadvantages in comparison to when they are used separately.

Another reason to create hybrid context models is the fact that applying context information models in real world problems/scenarios often lead designers to prefer covering a certain list of requirements (section 2.2) rather than others.

Ontology-based models provide excellent capabilities of logic inference and provides good support for reasoning. Strang & Linhoff-Popien (42) considered the ontology-based modeling as the most promising approach, as they enable formal analysis of the domain knowledge, promising contextual knowledge sharing and reuse in a computing system and context reasoning based on semantic web technologies (13). However, most frameworks supporting ontology-based context models have a centralized component for context reasoning and ontology management (27). Within the applicability to existing environments point of view, providing reasoning tools that are sufficiently optimized for low resources devices is yet an open problem.

Despite the advantage of rich expressiveness for representing complex context information, the **ontology-based model** brings with it a high cost in performance, thus the reasoning systems used to deduce complex context information have high complexity and performance issues.

The use of reasoners by mobile devices is an even bigger challenge, since mobile devices have limited resources like battery, memory and processing power. Depending on how complex the deduction of a new context information is, it can be impractical to use OWL-DL reasoner systems (38).

Concerning the scenario for which this proposed model of representing context information was developed, performance is an essential factor. Deduce all the complex context information in server-side is not a good choice. In our scenario of remote monitoring of chronic patients, quick answers and actions are essential to better help and support the patients' health. The remote monitoring procedure may be dealing with critical information that may need immediate attention.

Regarding this scenario, it is important to provide the mobile device with the capability of deducing new context information. This will make the mobile system capable of having faster reactions towards possible critical context information collected. To give an example, we can look to the application developed for this thesis: The monitoring system must change its behaviour to record five minutes of heart rate signal just when the patient has stopped doing some physical activity. Usually, after a physical activity, knowing how fast or slow the heart rate of the patient decreases, can give the doctor very useful information about other potential problems that the patient may have. The quick reaction of the mobile device can be compromised if the device rely entirely on the server-side to deduce that the patient just stopped doing a physical activity. The communication delay between the mobile application and the server-side might exist due to several reasons, such as low internet signal, no internet connection or a low quality of service.

In order to accommodate this requirement and design a way that makes it possible for resource-limited devices deduce new context information, we decide to address the advantages of the distribution composition and applicability of both object-oriented and markup-based context models approach.

Our proposal is to combine the advantages of ontology-based models (level of formality, reasoning and consistency check capabilities, knowledge sharing and reuse) with the advantages of the object-oriented models (distributed composition, heterogeneity of environments, more applicable to resource-limited devices) and the applicability of markup-based models that provides light weight data representation.

One example of combining ontology-based and object-oriented models is explained in 5.1, where the system, in order to send information about the patient's monitoring process to the mobile device, translates the information from the OWL ontologies into object-oriented classes. This set of object-oriented classes contains a subset of all information represented in the OWL ontologies, and is translated to JSON format in order to provide more applicable capabilities to the system. We send to the mobile device only the information that it needs to make decisions regarding the context that it is in. Once the information in JSON format reaches the mobile device, it is validated with a schema generated from the object-oriented classes.

In order to give modularity capabilities to the ontology-based model, we aggregate the capabilities of the spatial model approach, so that it would be possible to only send the necessary set of information that the mobile device needs and ensure that it will have all the information required to make decisions about the context that it is in.

In the spatial model approach, even if location is not a primary context of the context-aware application, a spatial organization of the context information can be useful. When the amount of managed context information is large, spatial partitioning can be used to cope with the complexity.

By combining the ontology-based and the spatial models, we can achieve efficient procedures for the execution of typical spatial queries and provides solutions for knowledge partitioning based on spatial divisions. The interoperability among different spatial models can be achieved by using ontologies (6).

In order to give an example of this modularity advantage of the spatial models, in the case of the implemented system in this work, the monitoring process is partitioned within "areas". Only part of the monitoring process information is sent to the mobile device. For example, there are parts of the monitoring process within the knowledge base that will be only necessary if the patient is located in a place that has an altitude greater than 2400 meters 4.10 or if the patient is doing some sort of physical activity. This examples show us that a spatial pre-selection of relevant context information could be reasonable to speed up the reasoning process by reducing the size of the knowledge base (7).

The model proposed by (11), explained in 3.2, uses the notion of spatial partitioning into context ontologies. Each of the possible contexts that an entity (e.g., a person) could be in is represented as an ontology in a way that, if the system wants to deduce (reason) information about a person within the context of a monitoring process (e.g., about what are the next possible states that the monitoring process can have for a specific person), it is only necessary to align the person and monitoring process ontologies. A reasoning tool, or any other process used to infer new context information, does not need other ontologies.

This suggests that a hybrid context model can be considered as an interesting approach for combining:

- The advantages of ontologies (e.g., interoperability potentiality);
- The modularity capability that the notion of spatial division can provide;
- And the applicability to existing environment and distribution composition that the object-oriented and markup-based approaches have.

Another advantage we take from the ontology-based model is the possibility of explaining the process behind the deduction of new information and all the decisions made by the application related to the remote monitoring of a patient.

In the ontology-based models, explaining the deduction process executed by the system is possible, but not trivial. It is indeed a complex task if the new context information is deduced using a reasoner tool. To do so, we must generate an explanation for the proof tree(41) that concluded the deduced fact.

Regarding the applicability requirement of our scenario, in the proposed model, we choose a way that this explanation can be acquired with a less complex form. The proposed model represents all the remote monitoring process using a state machine declared using the OWL language. Each of the states represents a possible situation that the patient can be considered in. Also, each of these states have few conditions that, when true, characterize the patient in the states that they belong. These conditions can be also expressions written using primary context information.

An expression is something you want to assert about a measure context data acquired from the patient. These expressions are defined using the OWL vocabulary and interpreted by the application. Expressions can be built using the following operators:

- The math operators: $(>,\geq,<,\leq,\neq)$
- The operator *EqualTo* that returns *true* whenever two objects (number or text) are equal. *EqualTo*(A,B) is *true* when A = B and *false* when A $\neq$ B.
- The NOT operator equivalent to the NOT operator of description logic.

An example of expression could be *Heart rate greater than 95 beats per minute.* Where *Heart rate* is a context information acquired from the patient (in number form) and *greater than* is the math operator “>”.

In order to create complex expressions it is possible to use the operators AND and OR that are equivalent to the AND and OR from description logic.

If there is a AND operator applied to two expressions, the AND operator will only return true if both expressions are true. Otherwise it will return false. In the case of the OR operator, it will return false if all the expressions are false. Otherwise it will return true.

With the expressions and information from the state machine that represents the monitoring process, when the system identifies a situation, it is capable to explain to the doctor why it was considered true. When a situation is considered true, it means that the patient is currently in that situation.

This solution does not bring all the advantages of the level of formality and reasoning that the ontology-based model has, but covers the applicability to existing environment requirement (to deduce complex context information from within the mobile device).

Despite not using the reasoning tools for deducing complex information on the mobile device, the reasoning tool is used for consistency check. To check the consistency of the metamodel created to represent the monitoring process and the model that the ontology engineer customizes for monitoring a patient, it is good to minimize human errors at the time of the definition of the monitoring process. This provides the user a validation tool that can increase the chances of the doctor actually inserting and instantiating his or her model correctly.

Other advantage of the ontology-based model is the fact that this model provides formal semantics to context data, making it possible to share and/or integrate context among different sources. The model proposed by this work proposed represents all the persisted information acquired from the sensors and deduced by the application with an ontology.

With the objective of providing a better model for remote monitoring of patients, this work developed a hybrid model that is a union of the spatial, ontology-based, markup-based and the object-oriented models. These models were unified with the intention of combining their strengths in order to meet our applicability requirement (e.g., deduce new context information from the mobile device) while trying not to carry the specific weaknesses into the resulting architecture.

7.2 Weak spots of the Patient-Buddy context information model

In order to breakdown Patient-Buddy-Build into a feasible subject for this dissertation, some simplifying decisions have been made. Remaining issues have been postponed for future work.

1. In the current implementation, the doctor needs knowledge about the

domain ontologies, its vocabulary and logical restrictions to be able to insert and correctly instantiate a remote monitoring for his/her patients. Therefore, we need the presence of an ontology specialist to aid the doctor in the task of instantiating his/her monitoring process. This weak spot is addressed as future work in Chapter 8.

2. When dealing with large amounts of data, the data cube model used to persist the collected context information can become very complex. The update rate of the context information can be defined from the model, and there is no restriction to the number of context information that can be monitored from the patient. Therefore, this model may suffer scalability problems.

7.3 Evaluating

7.3.1 Table of comparison

We have elaborated a table that compares each of the modelling approaches discussed in chapter 2.3 with how well they provide mechanisms for implementing the functionalities of the Patient-Buddy-Build model. We analyse each of the context models based on criteria for developing context information models listed in section 2.2. In the next subsections, we will associate each of the implemented functionality with a requirement of context information models and justify why this association should be made. It is worth mentioning that this table represents a very general comparison based on how well and naturally each of the implemented functionalities in the PBB could be implemented using others context models.

Approach / Criteria	DC	PVaR	RQI	IaA	LF	AEE
Key-value	-	-	-	-	-	+
Markup-based	-	+	+	-	-	+
Object-role	-	-	+	-	-	+
Object-oriented	+	+	+	+	-	+
Ontology-based	+	+	+	+	+	-
Patient-buddy	+	+	+	+	+	+

In the table above, the symbol "+" means that would be more likely to implement the functionalities of PBB related to a specific criteria using a specific modelling approach, and the symbol '-" means that would be more difficult.

7.3.2 Criterion for evaluation

In order to evaluate the proposed context information model, we list all the features that our system of remote monitoring has. After justifying why each of the chosen features were considered requirements that the underling context information model should provide, we will discuss how they could be implemented with the context information modelling approaches we discussed previously in chapter 2.3).

To each of the developed features within the system, we will associate one or more requirements (section 2.2) that have to be handled when creating models for representation of context information. Here is the criteria listed in section 2.2 and used for evaluating each of the context information models:

1. Distributed Composition (DC)
2. Partial Validation and Reasoning (PVaR)
3. Richness and Quality of Information (RQI)
4. Incompleteness and Ambiguity (IaA)
5. Level of Formality (LF)
6. Applicability to existing environments (AEE)

For each of the listed criterion above, we will associate with a implemented functionality of our proposed context information model. The implemented functionalities were focusing on the scenario of remote monitoring.

Distributed Composition (DC)

1.1 The possibility of adding or excluding new context data to be monitored from the patient: This feature comes from the distributed composition requirement, and shows that the system must know how to acquire new context data from different sources (sensors or web services). PBB uses the CMS (Context Management Service) to encapsulate the complexity of knowing how to communicate with the sensors. CMS is implemented using the object oriented paradigm. The ontology model only specifies how the system will interact with the context management service in order to retrieve a specific context information (e.g., with which context provider to connect for acquiring a specific context information).

1.2 The possibility of customizing the questions that are made to patients: Regarding the heterogeneity problem of context information, the system must be adaptable to changing environments, not only to adapt to different devices hardware specification, but to different users necessities too. This implemented aspect addresses how a question should be presented to the user. For example, from within the remote monitoring scenario, patients could have writing difficulties or vision difficulties regarding the tasks of answering and reading the question. The system must be adaptable to this context by, for example, increasing the size of the letters (for those who have vision problems) or providing multiple answer choices (for those who have writing difficulties).

1.3 The possibility of adding new contexts that the patient could be in, not only new monitored context information: During the remote monitoring, the patient could change his job and, for example, the new job could be located in an oil platform. The doctor might want to change the remote monitoring procedure of the patient to consider this new context, giving the system the ability of seeing the patient when it is at work (e.g., the oil platform could provide new doctors to assist him, so the monitoring process can send messages to these doctors too). The PBB represents each context in a separate ontology (the person is represented in an ontology and the environment of the oil platform in another). By using ontology alignment, the PBB can generate the context "person in the oil platform" and by integrating it to the monitoring process, it generates an ontology that will encapsulate information that is relevant to how the system will monitor a specific person in an oil platform environment. Customizable options about the environment are described in section 4.3.

1.4 The possibility of defining new sensors that could provide new context data: A mobile device can provide different sensors that could be useful to the task of collecting information. The capacity of adding new sensors to the system and integrating them to the information model is desirable for the remote monitoring scenario.

Partial Validation and Reasoning (PVaR)

2.1 Consistency check of the instantiated model for monitoring patients and the domain model: Consistency check is an important functionality for patient monitoring systems where a human can define his/her own monitoring process. It is very common that errors appear in the moment a user is entering in the system, the monitoring procedure the doctor described for his/her patient. The task to trying to reduce the quantity of errors that a user can insert when describing a monitoring process to the machine is difficult. PBB uses OWL-DL expressions to create rules that assert a set of characteristics that a monitoring procedure model should have (e.g., The PBB monitoring procedure does not permit that a patient is monitored without a responsible doctor, or that a disease is instantiated without any conditions related to it). After defining the monitoring procedure, the reasoning tool is important to validate it and check for any inconsistencies.

2.2 Schematic validation of the information produced by the mobile device. The information is exchanged among the server and the mobile devices by using JSON format, which is a markup language. Every time that a JSON object is generated within the PBB system, it passes through a schema validation that verifies if the types and classes of each acquired information are correct with respect to the model.

2.3 Reasoning capabilities for querying the persisted context information collected form the patient. All the context information collected from the patient is persisted in RDF format. A rule model is defined for this persisted information. This rule model defines classes, subclasses and properties characteristics. When querying for instances for a specific class using SPAQRL, using inference capabilities it is possible to retrieve the instances of the subclasses to. For example, when querying for all instances of the class "Condition" using the reasoner, the system will also return the instances of the subclasses "External Condition" and "Internal Condition".

Richness and Quality of Information (RQI)

3.1 The possibility of adding (or excluding) new context data to be monitored from the patient and to define new sensors: When adding new context data, we might be dealing with different sources and different precisions provided by sources for sensing it. In the PBB system this criteria is addressed by adding quality labels. For example, each sensor must have a percentage of how trustworthy the data provided is.

3.2 The system capability of explaining itself (executed on the web server): For all context information collected and deduced, it is important that the system knows how to explain to the doctor how each of the information was acquired and, for each sensor evolved, what their warranties of preciseness are. There are some specific information that are only acquired when a patient is in some particular situation, for instance, doing some physical activity. The system must know how to explain whether a information was acquired because it was considering that the patient was doing some physical activity, or, if it was acquired from a question, what the question was and why the system judged to make that question.

Incompleteness and Ambiguity (IaA)

4.1 The possibility of adding (or excluding) new context data to be monitored from the patient and to define new sensors: The PBB system handles the incompleteness and ambiguity by using ranges specifications. For example, a sensor that measures the body temperature of a person can have a precision of 2 degrees celsius. By providing range functionality that addresses information such as within which range a value of body temperature can vary and still be considered constant.

Level of Formality (LF)

5.1 Consistency check of the instantiated model for monitoring patients (executed on the web server): The capability of the consistency check will be limited by the formality level that a context representation model can provide.

5.2 The system capability of explaining itself: The formality level of the system can contribute to its capacity of explanation. The better a system can describe its procedures for itself, the better capacity it has for understanding and producing explanations. Almost all the context information acquired from the patient are series data and the persistence model should be able to represent this data adequately. Another requirement is to represent as well for each context information, how it was acquired and from what procedure (explain the provenance of a context information concerning why the system decided to acquire it). For example, this helps the doctor in the diagnosis since it enriches the set of information about the patient and helps understand the association that the system made in order to monitor a specific data.

5.3 The possibility of defining a variety of chronic diseases: That feature regards how well the model can handle the task of defining new diseases that a patient can suffer from. It is very common that during a monitoring process the doctor decides to monitor another disease too. The model must provide a way in which a new definition can be added and integrated with the current model. The model must have the capability of defining general properties about the diseases, for example, that all disease must have at least one condition related to it.

5.4 The possibility of adding new context data to be monitored from the patient or to define new sensors to be used by the system: This functionality concerns how well the user can express how each of the context data has to be interpreted (e.g., if it is a text or a number information and if it has a unit of measure). It is needed a minimum level of formality to be possible to say what data a sensor can provide and how they should be interpreted by the system. Each new sensor must be represented within the model in a way that the system can explain to the doctor what the sensor is, the precision that it has for measuring data, and what is each of the context data acquired by it.

5.5 The possibility of defining the form that a specific context data will be acquired depending on the patient preferences and available sensors: Sometimes the patient can prefer not using a specific sensor and instead providing the information through questions. When instantiating the model, patients' or doctors' preferences regarding the form that each data will be acquired from can be defined. The model should be able to define general characteristics about the acquisition process of the monitored conditions. For example, each of them must have one and only one form to be acquired.

Applicability to existing environments (AEE)

6.1 Partition the knowledge base in order to send only the necessary information to the mobile device: The mobile device must be able to know the monitoring process defined by the doctor to the patient. Depending on the situation of the patient (e.g., if the patient is doing physical activity, or if he is located in a place with high altitude) the mobile device should have only the necessary information about the monitoring process. For example, if the patient is not on a place with a high altitude, all the monitoring procedures related to the high altitude should not be persisted in the mobile device. The monitoring process is represented by using an automata. This automata is written using OWL language, if the patient is in the physical activity state of this automata, the set of states sent to the mobile device will be the states that are reachable by the physical activity state.

6.2 Deduce new context information based on primary context from within the mobile device: This feature of the implemented system is related to this requirement because it concerns the adaptation of the system to the applicable infrastructure.

6.3 Persist ontologies on the mobile device: The OWL ontology format is not designed for mobile devices. Thus, we use a light weight format to persist the ontologies on the mobile device (JSON format).

6.4 Schematic validation of the information produced by the mobile device: The schematic validation is done in the mobile device. The mobile device has low resources and limit hardware capacity, therefore the validation procedure must be realizable within the mobile devices.

In the following subsections we will compare how the other context modeling approaches surveyed in chapter 2.3 could implement each of the functionalities listed. Each of the functionalities implemented within the Patient-Buddy context model is associated with a criteria for evaluate context information models. The objective here is to compare how better or worse each of the models could implement the listed necessities.

7.3.3 Patient-Buddy-Build proposed model

The functionalities listed within the *distributed composition* criteria were implemented using advantages from the object-oriented, markup-based and ontology-based approaches. In order to implement item 1.1, the model provides an OWL ontology for defining new context information and new sources from which this context could come. The ontology brings the advantage of the consistency check from a variety of reasoners tools that can be used with OWL. Once an OWL ontology describes the scenario of the remote monitoring of the patient, a partition of the knowledge base concerning the current context of the patient (e.g., a state of his monitoring process) is picked up and translated to a set of object-oriented classes. This is done in order to prepare this part of the knowledge base to be transported into devices with low level resources (e.g., smart phones). Once the objects are instantiated, they are converted to JSON format, that has the characteristic of being a light weight format that is more suitable to be processed by devices with low resources. The object-oriented model is used in order to bring its *distribution composition* advantage together with the markup-based models.

The functionalities listed within the *Richness and quality* criteria are addressed by adding quality labels, for example, each sensor must have a precision percentage of how trustworthy the data provided by it is. *Incompleteness and ambiguity* are addressed using consistency checks towards the ontologies that define the remote monitoring process and its characteristics and by specifying ranges to values. For example, in order to acknowledge that a value is constant, a range can be used to show to the system that, if the values of a specific context are never out of the range, so they must be considered constant. Another form of providing support to ambiguity is by giving priority values. Within a monitoring process, there is a number of states that a patient can be in. States are specific parts of the monitoring process that must be addressed by the system differently. They specify a situation in which the patient can be, and the system can identify that a patient is in two states at once. For example, if one state is defining a monitoring behaviour for addressing a situation where the patient is doing a physical activity, another state could address the situation where he is located within a region of a high altitude. By using priority labels, the system can choose which behaviour to execute.

The use of the object-oriented model also brings up the item 1.4. We use the advantages of the object-oriented paradigm to make it possible to add new sensors to the system. The Patient-Buddy model extended classes from a context management service (CMS) with an abstract class that encapsulates the functionalities that every sensor should have with the rest of the framework. When defining new sensors, the user must specialize this abstract class and implement how the information should be interpreted and processed in order to be sent and used by the framework.

7.3.4 Key-value models

Key-value models are strong in the criteria of *applicability to existent environment*. Thus, the applicability requirement is a strong point of this approach when implementing the items *6.1*, *6.2* and *6.3*, since all the context information is described using key-value pairs. However, the lack of *level of formality* poses a difficult issue to overcome.

Concerning the representation of the monitoring process defined for a patient in the Patient-Buddy model, the key-value model would be able to implement it too, because the monitoring process is only described using the OWL language and interpreted only in the application level. The characteristic that the key-value model is a flat model (all the things are represented as key or values, there is no concept of hierarchy defined within the model) will complicate the implementation of the monitoring process defined in PBB.

Adding new sensors would be difficult because of the lack of formality, since it requires a certain level of abstraction capacity.

The *partial validation* requirement is a complex task because there is no scheme or at least a range of definitions to check against, making items *2.1*, *2.2* and *2.3* impossible to implement from within the model. This also makes the *level of formality* very limited. The fact that no scheme definition is possible, makes any developed algorithm for helping on the validation task, susceptible to errors at run time. Validating new context information inferred from the primary context and implementing a consistency check would be hard.

Considering the items related to the *richness and quality of information* requirement, it is possible to implement it from within the model. The PBB system only creates a vocabulary to express precision data of sensors and context information. The semantic of the properties created to express this precision is implemented in the application level.

7.3.5
Markup-based models

Regarding the *partial validation and reasoning* criteria, markup-based models have a set of scheme definitions that can be useful to implement the item *2.2*. These scheme definitions, which can be for example the XML schema, usually come with a set of validation tools for structural and syntactical checking. The schematic validation offered by, for example XML schemas, have the advantage to have liner time complexity (4), this enchant the applicability of the validation procedure (fulfilling item *6.4*).

Regarding the items *2.1* and *2.3*, it would not be possible to implement it from within the model, because it uses inferences capabilities from reasoners that work with description logic expressions.

Range checking is also possible to implement by using schema definitions. Functionalities that evolve the *richness and quality of information* and *Incompleteness and ambiguity* criterias (items from *3.1* to *3.2* and item *4.1*), are able to be implemented.

JSON is a light weigh format (a markup language) that PBB uses to persist the ontologies in the mobile device, thus the item *6.3* would be implemented by the markup-based models using the similar technique and complexity.

Some limitations of the markup-based models, would be a minor drawback regarding the *distribution composition* requirement. The non existence of complex overriding and merging mechanisms, for example, the Document Type Definitions[1] (DTD) when used on the markup structure level it does not provide overriding and merging (45). The implementation of the *1.3* functionality would be difficult, since it merges the existent model ontologies with a new context ontology. For instance, within the remote monitoring scenario, when trying to merge a new XML file containing information about the new job that a patient just got, to find similar concepts could be a more difficult task compared to if it was implemented using ontologies. Using ontologies, the reasoner can infer similar concepts automatically. In the markup-based models, the functionality provided by the reasoner would have to be implemented in the application level.

All the functionalities listed in the formality level requirement could be implemented too. But, again, implemented in a more difficult way, because some of the functionalities would concern the use of description logic and reasoner capabilities. The semantic provided by it and the inference process would have to be implemented in the application level.

[1]Document Type Definition: `http://www.w3.org/QA/2002/04/valid-dtd-list.html`

7.3.6 Graphic/Object-role based models and Spatial Models

The Object-role and Spatial model will be evaluated together because from the point of view of the implemented functionalities they would have the similar solutions to provide implementations. Most of the spatial based models can be seen as object-role models that give more importance to the spatial context (7), and, because of that, they inherit the advantages of the object-role models.

The functionalities implemented in PBB related to the *richness and quality of information* and the *incompleteness and ambiguity* requirements would be possible for the object-role approach to implement by using quality labels. One example of this implementation is the extension of the ORM model proposed in (16). This extension of ORM model provides solutions for items *3.1*, and *4.1*. In PBB the model only indicates how the system should handle the problem related to this two requirements, the semantic of how to interpret these indications written in the model is implemented in the application level.

The object-role approach would be useful to implement the functionalities related to the *applicability to existing environments* requirement. Persist structure data from the monitoring process in the mobile device would use similar device resources that the implementation used by PBB uses (item *6.3*). PBB uses the JSON format, but when the OWL ontologies are converted to this format they only maintain their structure and schema. Thus representing it with the object-role paradigm would be a possible way too. Even the categorization of the context information made by the PBB model in order to know if a context information was acquired from a sensor or through a question is possible to be implemented within the object-role approach. The extension proposed by Henricksen in (18) is an example that it is possible to divide and categorize context information. This fact could help the implementation of functionalities *3.1*, *3.2*, *5.2*, *5.3*, *5.4*, *6.1* and *6.4*.

The functionalities related to the *partial validation* requirement are possible regarding the possibility of using modeling technologies like UML. There are several tools that can be used to validate schematically a UML model. But implementing the item *2.3*, since the object-role model only carries syntactical information, it would be required to implement a query executor mechanism that have the capability of reason using the model. For example, to infer that when a query asks for all instances of a “Condition” class the response should include all instances of the subclasses of the “Condition” class too.

The functionality *1.3* related to the *distributed composition* requirement would be hard to implemented due to the fact that the merging of models is not a trivial task. Again, the object-role models carry only syntactical information, a better formality level would be necessary in order to provide knowledge sharing and merging.

7.3.7 Object-oriented models

Regarding the implementations related to the *richness and quality of information* and the *incompleteness and ambiguity* requirement, the object-oriented models can implement it using similar solution of the object-role and markup-based models. By using the concept of classes, it could be possible to define that all sensors should have a set of quality parameters. Using range definitions would cover all the functionalities related to these two requirements implemented in the PBB system.

In order to deal with the implemented functionalities *2.1*, *2.2*, *2.3* and *5.1*, constrains could be used to validate the created monitoring process and also to validate the new context information generated in the mobile device. The only drawback here is related to the consistency check and the reasoning related functionalities (items *2.1*, *2.3* and *5.1*). Because all constrains have to be defined explicitly within the model. When defining the rules using OWL-DL language, it uses description logic expressiveness and reasoner capability, and implicit relations/constrains are revealed and applied by the reasoner. The constrains of the object-oriented model are written in a syntactical level.

But it is still possible to implement functionalities *2.1* and *2.3* in the application level. For example, by implementing a program that simulates reasoner capabilities on top of the language used to formulate the object constrains. It would be more difficult to implement this functionalities compared to the ontology-based models, because the reasoner and the language to describe the constrains/rules already exist on the ontology-based models (OWL-DL language and reasoners).

The functionalities related to the *distributed composition* requirement would be possible to be implemented using the object-oriented approach. The possibility of adding new sensors or context information could be provided by abstract model of classes that could be extended in order to add new sensors or context information. PBB uses the CMS (Context Management Service) that is implemented using an object-oriented approach, and this model encapsulates the complexity of knowing how to communicate with each sensor.

The functionalities related to the *applicability to existing environment* are possible and could be implemented using similar resources from the mobile device, compared to which hardware resources the PBB uses.

7.3.8 Ontology-based models

Compared to how the PBB system implements all the functionalities, the only difference is regarding how the ontology-based model would implement the functionalities related to the *applicability to existing environment* requirement. It would be difficult to implement functionality *6.2* by using a reasoner for OWL-DL in a mobile device. Mobile devices have more limited resources, and, for example, in order to know when an instance is of a **Condition** class or a **State** class, it categorizes a realization problem. The realization problem is when the reasoner wants to find the most specific class the individual belongs to. The realization problem has *NExpTime* complexity (7). Thus the use of reasoners by a mobile device is yet a problem.

8
Conclusion

In this thesis we describe our proposed model for representation of context information that was designed for the scenario of remote monitoring of patients with chronic diseases.

The creation of the ontology vocabulary that can describe the monitoring process, the diseases and the form of acquire each of the monitored condition was mainly designed for the requirements for monitoring patients with atrial fibrillation. Because of this, we can say that our system can monitor patients with atrial fibrillation (considering the requirements of the atrial fibrillation). In previous chapters, we derived some requirements on a context model which would be optimal for usage in an context-aware application environment. Based on the scenario of remote monitoring of patients with chronic disease, we defined a set of functionalities that an application should implement/provide. The determination of what functionalities an application for remote monitoring should provide was mainly derived from the necessities of remote monitoring patients with the atrial fibrillation disease. This necessities were originated from several meeting with a medical specialist in atrial fibrillation. Based on that necessities we implemented all the possible customizations to make the context model capable of monitoring any chronic disease that can have theirs monitoring process expressed by using the features and customizations available in PBB.

We related this set of required functionalities (that solves a problem for a specific scenario) with the modeling context requirements listed in chapter 2.3. Based on a survey of the most prominent context modeling approaches, we discussed the fulfilment of the different approaches with respect to the implemented functionalities.

Comparing only the context models listed in chapter 2.3, the most promising assets for context modeling for the remote monitor of patients with chronic disease scenario with respect to the requirements and functionalities listed in chapter 7.3 can be found in the ontology and object oriented category. The representatives of this category met the requirements for implementing the listed functionalities best. However, this does not mean that the other approaches are unsuitable.

Due to our analysis we arrived at the conclusion that hybrid context models (i.e. The Patient-Buddy-Build model) are an interesting way for realizing systems and formulate context models (focus on specific scenarios) for context-aware applications. When focusing on a specific scenario, hybrid context models have the possibility of combining advantages from different models approaches. This combination is interesting because depending on the applicable scenario/environment, the context modeling requirements can have different importances. Thus, an hybrid model that implements the requirements/functionalities that have more importance to the target environment would certainly be more suitable.

To illustrate the functionalities of the proposed hybrid model, we developed the Patient-Buddy-Build that is a framework that can be used to instantiate applications for monitoring a variety of chronic diseases. The Patient-Buddy-Build framework have two possible ways to be extended/customized. One of the ways of customizing this system is through ontologies. The ontologies must be customized before use the application. All the possible customizations are listed in section 4. In general lines, by customizing the ontologies it is possible to:

- Define the chronic disease that will be monitored.
- Define all the monitored conditions that the doctor want to collect from the patient and from the patient's environment. It is also possible to define how each of the monitored conditions will be acquired (from sensors, from a web service or from a question).
- Define the monitoring process that the doctor wants for each of his/her patient.
- If a context information would be acquired by a question, it is possible to customize the answer form of that question. If it have to be answered with multiply choice answers that the patient can only choose one answer, by multiple choice answers that the patient can chose more than one answer or by only using free a text.

The PBB framework can be also extended in order to include new sensors. By using an object-oriented paradigm, the user can extend classes and implement his/her own logic of communicating with new sensors. By extending this classes and adding the sensors information in the customized ontologies, the PBB framework can understand how to communicate to the new sensors and how to integrate them into the monitoring process of the patient.

Other contribution of this thesis is the set of ontologies written in OWL-DL that provides a vocabulary to describe a variety of details of a remote monitoring of patients with chronic diseases. All the ontologies provides logical restrictions (written in OWL-DL) that aids the user who wants to write new monitoring procedures to avoid customizing the ontologies in a inconsistent way.

The Monitoring ontology, compared to the other ontologies, brings one different contribution for this thesis that is it‘s form of representing context information.

The set of states model of this ontology was created with the main purpose of representing the monitoring process of a patient with atrial fibrillation. However, it was generalized to represent a monitoring process in general, because the domain of the monitored information is represented in the monitoring ontology. Thus, we can see this model as a form of representing context information, that focus on the monitoring of some entity, that can be a person, an object or even group of them.

When we say that the model of states focus on the monitoring, we say that because during a monitoring is necessary that the application knows how to behave towards the acquired context information and how to take different actions according to the context that the application is inserted, and both are characteristics that are represented in the model.

Another contribution of this thesis is that it gives a practical implementation and a possible application to the theoretical framework purposed by (11). This practical implementation proves that with only this set of logical operators and level of formality it is possible to implement an application that can remote monitor patient with chronic diseases that have the set of functionalities that we offer in PBB. The practical implementation of PBB does not contain the existential and the for all operators of the first order logic available for the user to customize and express his/her form of remote monitoring of patients.

8.1 Future Work

Due to the experience gained during the development of a model that could represent and proper manage context information within the scenario of remote monitoring, a variety of possible lines of future works emerged.

1- The task of remote monitoring of patients with chronic diseases can potentially have a large volume of context information gathered from numerous distributed sensors and may need to support any client requesting context information. To achieve a scalable context management system it is necessary to provide scalable solutions for the system's components. The approach taken this far in this proposed context management model is to persist all the data in RDF format. A possibility of future work is to map the RDF model directly to one or more relational database tables (39). The chosen repository for persist our RDF data was the OpenLink Virtuoso Universal Server. Virtuoso provides a solution to map relational data to RDF. It can dynamically convert relational data into RDF and expose it from a Virtuoso-hosted SPARQL endpoint [1]. We could use it to improve the scalability and performance of the proposed context management model.

2- Concerning the interface requirements that have to be addressed in order to adapt the system to the patient's preferences and physical needs there are some emerging promising solutions. PRISSMA (Presentation of Resources for Interoperable Semantic and Shareable Mobile Adaptability) is a domain-independent vocabulary for displaying Web of Data resources in mobile environments. It consists in a lightweight ontology that describes the context conditions in which a given set of Fresnel declarations must be activated. This vocabulary was a initiative that came after the Fresnel conceptualization(30), which is solution for capturing the general needs for presenting RDF content to users and wanting to promote the exchange of presentation knowledge. The Fresnel vocabulary is described by an OWL ontology that relies on two concepts: *lenses* and *formats*. Lenses specify which properties of RDF resources are shown and how these properties are ordered while formats indicate how to format a content selected by lenses. PRISSMA models the context of usage of each Fresnel *lens* or *format* by modeling different dimensions: the user who requested the resource, the device features on which the resource will be displayed and the surrounding physical environment[2].

The usage of the PRISSMA model could be a promising future work. Within the remote monitoring scenario, the elderly people could benefit a lot giving the vision natural deterioration that they suffer because of their ages.

3- Data mining techniques to enchant the capability of the system of better using the collected context data. This line of future work seeks to better utilize all the context data collected and deduced from the patient or a group of patients. Some epidemics could be detected with the evaluation of large quantities of data that, for example, could indicate a large group living in a small town that is suffering from Dengue fever (26).

[1]Mapping relational data to RDF using OpenLink Virtuoso: `http://virtuoso.openlinksw.com/whitepapers/relational%20rdf%20views%20mapping.html`

[2]PRISSMA: `http://ns.inria.fr/prissma/v2/prissma_v2.html`

4- Autonomic computing and ubiquitous systems walk side by side. Ubiquitous systems are designed to disappear within the environment in a way that the user do not have to know that the system is proving him with help for his daily activities. In order to assist the user in his everyday tasks, the context management system must take a number of decisions to act with the environment that must be configurable according to the user's process (e.g., if the system will be allowed to send an emergency message to the hospital service without consulting the doctor before). This aspect of the context management system concerns to the level of autonomy that the user wants it to have. This autonomy level should be configurable in an easy way. Within the remote monitoring scenario and with the advance of the information technology in general real life areas, autonomic computing could be a promising line for future work regarding context management models. To give an example of it within the remote monitoring scenario, it is desirable that a context management system should adapt it's privacy settings dictated by the patient depending on the context that he is in. For instance, if the patient has a monitoring system in his smart phone in a moment that he just suffers a car accident, it is desirable that the emergency team that first arrives in the scene could have permission to access the patient's data. In order to do this, the system must be aware of the situation and give the emergency team the permission.

5- At this stage of the context management model implementation, the way that the monitoring process knowledge and medical knowledge is inserted is far from being at least an acceptable form for the doctor communicate with the application. However we could try to address this problem from within the remote monitoring scenario and try the usage of, for example, medical guidelines. Medical guidelines (also called a clinical guideline, clinical protocol or clinical practice guideline) is a document with the aim of guiding decisions and criteria regarding diagnosis management and treatment in specific areas of healthcare. Such documents have been used for many years during the history of medicine. This documents are not protocols to be restrictedly followed, their objective is only to guide the doctor on how to manage and treat each disease. Providing a set of ontologies that implements a specific guideline can be a solution for minimizing the effort of the doctor to insert the remote monitoring knowledge.

6 - There exists some researches towards porting OWL reasoners to specific embedded devices (38). This researches show that choosing a low expressive logic can help to decrees the complexity inferring new context information using reasoners. This work shows that a reasoner for OWL-RL is possible to be portable to mobile devices. Other possibility is to use the EL

logic that is a subset of the DL and thus less complex to be used within the mobile devices scenario.

Bibliography

[1] ABOWD, G. D.; DEY, A. K.; BROWN, P. J.; DAVIES, N.; SMITH, M. ; STEGGLES, P. **Towards a better understanding of context and context-awareness.** In: PROCEEDINGS OF THE 1ST INTERNATIONAL SYMPOSIUM ON HANDHELD AND UBIQUITOUS COMPUTING, HUC '99, p. 304–307, London, UK, UK, 1999. Springer-Verlag.

[2] ASMI A, RAGAVAN L, C. J. **Pervasive asthma monitoring system. pams. a health systems approach to remote monitoring of asthma.** 2007.

[3] Baader, F.; Calvanese, D.; McGuinness, D. L.; Nardi, D. ; Patel-Schneider, P. F., editors. **The description logic handbook: theory, implementation, and applications.** New York, NY, USA: Cambridge University Press, 2003.

[4] BARBOSA, D.; LEIGHTON, G. ; SMITH, A. **Efficient incremental validation of xml documents after composite updates.** In: PROCEEDINGS OF THE 4TH INTERNATIONAL CONFERENCE ON DATABASE AND XML TECHNOLOGIES, XSym'06, p. 107–121, Berlin, Heidelberg, 2006. Springer-Verlag.

[5] BECHHOFER, S.; VOLZ, R. ; LORD, P. **Cooking the semantic web with the owl api.** In: PROC. OF THE FIRST INTERNATIONAL SEMANTIC WEB CONFERENCE 2003 (ISWC 2003), OCTOBER 21-23, 2003, SANIBEL ISLAND, FLORIDA, p. 659–675, 2003.

[6] BECKER, C.; NICKLAS, D. **Where do spatial context-models end and where do ontologies start? a proposal of a combined approach.** In: PROCEEDINGS OF FIRST INTERNATIONAL WORKSHOP ON ADVANCED CONTEXT MODELLING, REASONING AND MANAGEMENT, 2004.

[7] BETTINI, C.; BRDICZKA, O.; HENRICKSEN, K.; INDULSKA, J.; NICKLAS, D.; RANGANATHAN, A. ; RIBONI, D. **Pervasive Mob. Comput.** A survey of context modelling and reasoning techniques, journal, v.6, n.2, p. 161–180, Apr. 2010.

[8] BIZER, C.; SCHULTZ, A. **International Journal On Semantic Web and Information Systems**. The berlin sparql benchmark, journal, 2009.

[9] BOUZY, B.; CAZENAVE, T. **Using the object oriented paradigm to model context in computer go**, 1997.

[10] **Brazil has more than 96 million broadband connections.** ARCTEL, Website, June 2013. Available in `http://www.arctel-cplp.org/noticias/detalhe/418/en` [Accessed in March 2014].

[11] CAFEZEIRO, I.; VITERBO, J.; RADEMAKER, A.; HAEUSLER, E. ; ENDLER, M. **A formal framework for understanding context-aware behavior in ubiquitous computing.** In: COMMUNICATIONS IN COMPUTER AND INFORMATION SCIENCE, volume 17. Springer Verlag, Porto Sani (Greece), 2008.

[12] DEY, A. K. **Personal Ubiquitous Comput.** Understanding and using context, journal, v.5, n.1, p. 4–7, Jan. 2001.

[13] GLASSEY, R.; STEVENSON, G.; RICHMOND, M.; NIXON, P.; TERZIS, S.; WANG, F. ; FERGUSON, I. **Towards a middleware for generalised context management**. In: IN 1ST INTERNATIONAL WORKSHOP ON MIDDLEWARE FOR PERVASIVE AND AD-HOC COMPUTING (MPAC03, p. 45–52, 2003.

[14] GRUBER, T. R. **Int. J. Hum.-Comput. Stud.** Toward principles for the design of ontologies used for knowledge sharing, journal, v.43, n.5-6, p. 907–928, Dec. 1995.

[15] HALPIN, T. **Information Modeling and Relational Databases: From Conceptual Analysis to Logical Design (The Morgan Kaufmann Series in Data Management Systems)**. Morgan Kaufmann, april 2001.

[16] HENRICKSEN, K.; INDULSKA, J. **Modelling and using imperfect context information**. In: PROCEEDINGS OF THE SECOND IEEE ANNUAL CONFERENCE ON PERVASIVE COMPUTING AND COMMUNICATIONS WORKSHOPS, PERCOMW '04, p. 33–, Washington, DC, USA, 2004. IEEE Computer Society.

[17] HENRICKSEN, K.; INDULSKA, J. **Pervasive Mob. Comput.** Developing context-aware pervasive computing applications: Models and approach, journal, v.2, n.1, p. 37–64, Feb. 2006.

[18] HENRICKSEN, K.; INDULSKA, J. ; RAKOTONIRAINY, A. **Modeling context information in pervasive computing systems.** In: PROCEEDINGS OF THE FIRST INTERNATIONAL CONFERENCE ON PERVASIVE COMPUTING, Pervasive '02, p. 167–180, London, UK, UK, 2002. Springer-Verlag.

[19] HENRICKSEN, K.; INDULSKA, J. ; RAKOTONIRAINY, A. **Generating context management infrastructure from high-level context models.** In: IN 4TH INTERNATIONAL CONFERENCE ON MOBILE DATA MANAGEMENT (MDM) - INDUSTRIAL TRACK, p. 1–6, 2003.

[20] HORRIDGE, M.; BECHHOFER, S. **Semant. web.** The owl api: A java api for owl ontologies, journal, v.2, n.1, p. 11–21, Jan. 2011.

[21] HORRIDGE, M.; BECHHOFER, S. **Semant. web.** The owl api: A java api for owl ontologies, journal, v.2, n.1, p. 11–21, Jan. 2011.

[22] INDULSKA, J.; ROBINSON, R.; RAKOTONIRAINY, A. ; HENRICKSEN, K. **Experiences in using cc/pp in context-aware systems.** In: IN PROC. OF THE INTL. CONF. ON MOBILE DATA MANAGEMENT (MDM, p. 247–261. Springer, 2003.

[23] **Cleveland clinic unveils top 10 medical innovations for 2012.** Website, Outubro de 2011. Available in `http://tinyurl.com/4y5tlgj` [Accessed in August 2013].

[24] **Laboratory for advanced collaboration - puc-rio.** `http://lac-rio.com/`. [Accessed in August 2013].

[25] **Laboratóio de sistemas distribuíos - ufma.** `http://www.lsd.ufma.br`. [Accessed in August 2013].

[26] MATTHEWS, J.; KULKARNI, R.; GERLA, M. ; MASSEY, T. **Mob. Netw. Appl.** Rapid dengue and outbreak detection with mobile systems and social networks, journal, v.17, n.2, p. 178–191, Apr. 2012.

[27] PAGANELLI, F.; BIANCHI, G. ; GIULI, D. **A context model for context-aware system design towards the ambient intelligence vision: experiences in the etourism domain.** In: PROCEEDINGS OF THE 9TH CONFERENCE ON USER INTERFACES FOR ALL, ERCIM'06, p. 173–191, Berlin, Heidelberg, 2007. Springer-Verlag.

[28] PARNAS, D. **On the criteria to be used in decomposing systems into modules.** Communications of the ACM, 1972.

[29] PASCOE, J. **The stick-e note architecture: extending the interface beyond the user**. In: PROCEEDINGS OF THE 2ND INTERNATIONAL CONFERENCE ON INTELLIGENT USER INTERFACES, IUI '97, p. 261–264, New York, NY, USA, 1997. ACM.

[30] PIETRIGA, E.; BIZER, C.; KARGER, D. ; LEE, R. **Fresnel: a browser-independent presentation vocabulary for rdf**. In: PROCEEDINGS OF THE 5TH INTERNATIONAL CONFERENCE ON THE SEMANTIC WEB, ISWC'06, p. 158–171, Berlin, Heidelberg, 2006. Springer-Verlag.

[31] PINHEIRO, V.; HAEUSLER, E. ; ENDLER, M. **Patient-buddy-build: Acompanhamento remoto móvel customizável de pacientes com doenças crônicas**. In: ANAIS DO XIII CONGRESSO BRASILEIRO DE INFORMáTICA EM SAúDE ISSN: 2178-2857, XIII Congresso Brasileiro de Informática em Saúde, Curitiba, Brasil, November 2012.

[32] PINHEIRO, V.; HAEUSLER, E. ; ENDLER, M. **Patient-buddy-build: Customized mobile monitoring for patients with chronic diseases**. In: THE 5TH INTERNATIONAL CONFERENCE ON EHEALTH, TELEMEDICINE, AND SOCIAL MEDICINE, ISBN: 978-1-61208-252-3, PP. 121-124, eTELEMED 2013, The Fifth International Conference on eHealth, Telemedicine, and Social Medicine, Nice, France, February 2013.

[33] R H ISTEPANIAN, S LAXMINARAYAN, C. S. P. **M-Health: Emerging Mobile Health Systems**. New York, NY, USA: Springer, 2006.

[34] ROUSSAKI, I.; STRIMPAKOU, M.; PILS, C.; KALATZIS, N. ; ANAGNOSTOU, M. **Hybrid context modelling: A location-based scheme using ontologies**. In: PROCEEDINGS OF THE FOURTH ANNUAL IEEE INTERNATIONAL CONFERENCE ON PERVASIVE COMPUTING AND COMMUNICATIONS WORKSHOPS (PERCOMW'06), 2006.

[35] SACRAMENTO, V.; ENDLER, M.; RUBINSZTEJN, H. K.; DOS S. LIMA, L.; GONÇALVES, K. M.; NASCIMENTO, F. N. ; BUENO, G. A. **IEEE Distributed Systems Online**. Moca: A middleware for developing collaborative applications for mobile users., journal, v.5, n.10, 2004.

[36] SCHILIT, B.; ADAMS, N. ; WANT, R. **Context-aware computing applications**. In: PROCEEDINGS OF THE 1994 FIRST WORKSHOP ON MOBILE COMPUTING SYSTEMS AND APPLICATIONS, WMCSA '94, p. 85–90, Washington, DC, USA, 1994. IEEE Computer Society.

[37] SCHILIT, B.; ADAMS, N. ; WANT, R. **Context-aware computing applications.** In: PROCEEDINGS OF THE 1994 FIRST WORKSHOP ON MOBILE COMPUTING SYSTEMS AND APPLICATIONS, WMCSA '94, p. 85–90, Washington, DC, USA, 1994. IEEE Computer Society.

[38] SEITZ, C.; SCHÖNFELDER, R. **Rule-based owl reasoning for specific embedded devices.** In: PROCEEDINGS OF THE 10TH INTERNATIONAL CONFERENCE ON THE SEMANTIC WEB - VOLUME PART II, ISWC'11, p. 237–252, Berlin, Heidelberg, 2011. Springer-Verlag.

[39] SEQUEDA, J. F.; ARENAS, M. ; MIRANKER, D. P. **On directly mapping relational databases to rdf and owl.** In: PROCEEDINGS OF THE 21ST INTERNATIONAL CONFERENCE ON WORLD WIDE WEB, WWW '12, p. 649–658, New York, NY, USA, 2012. ACM.

[40] **Em 2012 smartphones venderão 73% mais.** Gazeta do Povo, Website, March 2012. Available in `http://tinyurl.com/9zjhbxf` [Accessed in March 2014].

[41] SPECHT, G. **Generating explanation trees even for negations in deductive database systems.** In: LPE, p. 8–13, 1993.

[42] STRANG, T.; LINNHOFF-POPIEN, C. **A context modeling survey.** In: IN: WORKSHOP ON ADVANCED CONTEXT MODELLING, REASONING AND MANAGEMENT, UBICOMP 2004 - THE SIXTH INTERNATIONAL CONFERENCE ON UBIQUITOUS COMPUTING, NOTTINGHAM/ENGLAND, 2004.

[43] TELES, A.; PINHEIRO, D.; GONÇALVEZ, J.; BATISTA, R.; SILVA, F.; PINHEIRO, V.; HAEUSLER, E. ; ENDLER, M. **Mobilehealthnet: A middleware for mobile social networks in m-health.** In: LECTURE NOTES OF THE INSTITUTE FOR COMPUTER SCIENCES, SOCIAL INFORMATICS AND TELECOMMUNICATIONS ENGINEERING, 3rd International Conference on Wireless Mobile Communication and Healthcare (MOBIHEALTH 2012), Paris, November 2012. Springer.

[44] TOPCU, F. **Context modeling and reasoning techniques.** SNET Seminar in the ST, 2011.

[45] W3C. **Composite capabilities / preferences profile (cc/pp).** `http://www.w3.org/TR/CCPP-struct-vocab/`. [Accessed in August 2013].

[46] WAGNER, E. H.; AUSTIN, B. T.; DAVIS, C.; HINDMARSH, M.; SCHAEFER, J. ; BONOMI, A. **Health Affairs**. Improving chronic illness care: translating evidence into action., journal, v.20, n.6, p. 64–78, 2001.

[47] WANG, X. H.; ZHANG, D. Q.; GU, T. ; PUNG, H. K. **Ontology based context modeling and reasoning using owl**. p. 18–22, 2004.

[48] WILLIAM J. CLANCEY, E. H. S. **Readings in medical artificial intelligence.** Boston, MA, USA: Addison-Wesley Longman Publishing Company, 1984.

9 Appendix - PBB Data Structures

These Data Structures were created using the Data Cube RDF vocabulary.

The *Condition Measurement* data structure has the follow dimensions, attributes and measurements:

I **Dimensions:**

- *Date and Time*: The date and the time that the condition was measured/acquired from the patient. If it was deduced from another conditions, the date and time refers to when the information was generated.
- *Condition Identifier*: The URI of the condition measured that identifies it (e.g.: if it was a heart rate, a breath rate or an activity of the patient).
- *The Patient*: An URI that identifies who is the patient that the measured condition is related to.
- *Acquisition form by question*: If the measured condition was acquired from a question, this dimension identifies what was that question.
- *Automatic acquisition form*: If the measured condition was acquired in a automatic way, this dimension identifies what was the sensor or what was the web service responsible for its acquisition.
- *Persisted by state behavior*: If the measurement of the condition was sent to be persisted in the PBWS context repository by the behavior of the state.
- *Persisted by state variable*: If the measurement of the condition was sent to be persisted in the PBWS context repository by the state variable. The variable contains a test related to a measurement of a condition, if it is true, the measurement is persisted. For example, the altitude variable (patient's location altitude greater then 2400 meters) if it is true, the patient's location altitude is persisted.

II **Measurements:**

- *Number value*: This is used if the condition is measured using a numerical representation. For example, heart rate is a condition that is measured using a numerical representation (e.g.: 90 beats per minute).
- *Text value*: This is used if the condition is measured using a textual representation. For example, a headache condition is measured using text values that represent if the patient is feeling headache or not. This is described by the patient using descriptions like "feeling headache" or "not feeling headache".
- *Explanation*: This is a textual representation that contains a explanation about what has been measured, how the condition was measured and why it was measured. It summarizes all the dimensions information in a text representation and add the precision of the sensor that measured the condition (if it was acquired by a sensor). This text is intended to be read by the doctor.

III **Attributes:**

There is only one attribute used, which is the unit of the measured condition. It will only exit when the measured condition is a numerical representation (e.g.: Beats per minute, meters, breath per minute).

The *Explanation of State* data structure has the follow dimensions, attributes and measurements:

I **Dimensions:**

- *Date and Time*: The date and the time that the patient was considered to be in the state.
- *State variable*: Will contain all the variables used to conclude that the patient was in this state. All the state variables in the Patient-Buddy-Build system are identified using an URI. This dimension will be related to one ore more URI's depending on how many variables the related state has.
- *State Action*: If the state has an action associate to it, what is this action. Whenever a patient is considered to be in a state, if it has an action the action is executed. The action can be to send a SMS message to the doctor or to people that was previously registered in the system, or to send an alert to the patient.

II **Measurements:**

1. There is only one measurement, that is the explanation about why and how the patient was considered to be in a specific state of the monitoring process. This explanation is intended to be read by the doctor and will resume all the dimensions related to the state.

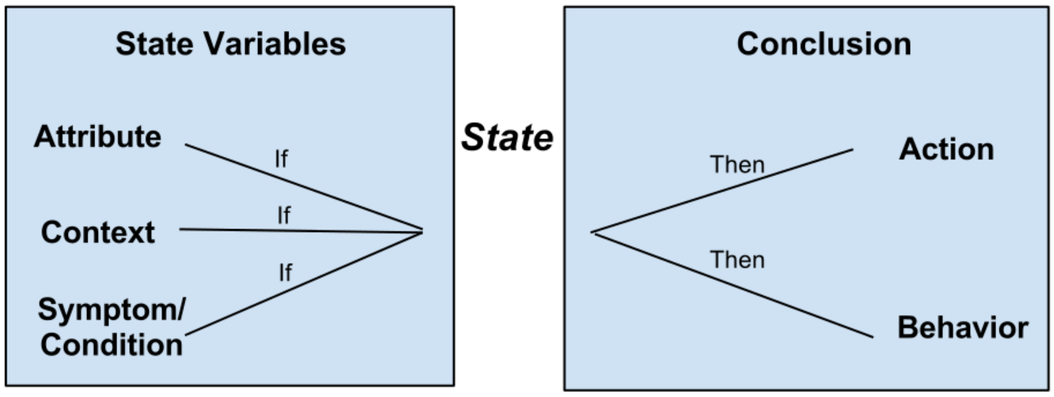


Figure 9.1: Structure of a State in the monitoring process.

The other two data structures represents how a state variable is persisted. Figure 9.1 shows what are the state variables. To represent the tree types of state variables described in the figure, we created two ontology classes: The **State Variable Class** and the **Complex State Variable Class**. The **State Variable Class** is persisted using the *explication of state variable* data structure and the **Complex State Variable Class** is persisted using the *explication of complex state variable.*

Both *Explanation of State Variable* and *Explanation of Complex State Variable* data structures have the same components:

I **Dimensions:**

– *Date and Time*: The date and the time that the state variable was verified as true.

II **Measurements:**

– *Explanation*: Every state variable carries information about which state it is inserted, how it was considered true and specific information about the conditions that made it true.

The different between this two data structures is that the complex state variable have two or more state variables connected using the OR operator or an AND operator. The AND operator indicates that the complex variable is only true when both state variables are true. The OR operator indicates that the complex variable is true if at least one of the state variables are true.

10 Appendix - Algorithms Used

In this appendix we will list the algorithms used for compose and decompose the contextualizes ontologies. This algorithms are created and defined by Cazefeiro in (11).

The first algorithm have the goal of verifying if a composition of contextualized ontologies were executed respecting the logical constraints and if the structure of the concepts and the relations (both taxonomic and non-taxonomic) are preserved.

Considering two ontologies O = (C,R,H^C, rel,A), the resulting ontology after the composition $O^{'} = (C^{'}$,$R^{'}$,$H^{C^{'}}$, $rel^{'}$,$A^{'}$), and a link between O and $O^{'}$ f : C $\rightarrow$ $C^{'}$, g : R $\rightarrow$ $R^{'}$).

The algorithm to be executed is defined by Cafezeiro as follows:

1. "For all (c1, c2) $\in H^C$ if (f(c1), f(c2)) $\notin H^{C^{'}}$ returns false;"
2. "For all r $\in$ (c1, c2) $\in$ rel if (f(c1), f(c2)) $\notin$ g(r) returns false;"
3. "For all a $\in$ A if $trans_{fg}(a) \notin$ Th($A^{'}$) returns false"

Explaining the steps of the algorithm:

- Step 1: Test if the taxonomic relations between the concepts of the ontology structure O are preserved with respect of the ontology structure $O^{'}$. If not, return false.
- Step 2: Test if the non-taxonomic relations between the concepts of the ontology structure O are preserved with respect of the ontology structure $O^{'}$. If not, return false.
- Step 3: Test if the set of axioms of the ontology structure O is still a valid sentence in the Theory of the ontology structure $O^{'}$. If checks if the map between the theories of the ontologies are consistent.

The second algorithm concerns the operations of alignment (Context Integration and Collapsed Union) and coalignment (Entity Integration and Relative Intersection). It tests if all the concepts and relations of the mediator ontology structure are consistently mapped with respect of the square: C $\leftarrow$

C1 ← EMed → C2 → C. Consistently in a way that EMed is mapped in the same way in C following EMed → C1 → C or EMed → C2 → C.

The square is represented by four links: $O' \xleftarrow{l_1'} O_1 \xleftarrow{l_1} O \xrightarrow{l_2} O_2 \xrightarrow{l_2'} O'$. The second algorithm is defined by Cafezeiro as follows:

1. “For all c $\in H^C$ if $(f_1'(f_1(c))) \neq (f_2'(f_2(c)))$ returns false;”

2. “For all r ∈ rel if $(g_1'(g_1(c))) \neq (g_2'(g_2(c)))$ returns false;”

Explaining the steps of the algorithm:

– Step 1: Verifies if every concept of the ontology structure EMed is mapped in the same way following EMed → C1 → C or EMed → C2 → C.
– Step 2: Verifies if all the non-taxonomical relations of the ontology structure EMed are mapped in the same way following EMed → C1 → C or EMed → C2 → C.

The third algorithm is used to compute the Collapsed Union operation but can be also modified to execute the Context Integration operation. The only difference between this operations if that when we modify the Collapsed Union algorithm and contextualize each entity or context in it self it becomes the Context Integration algorithm.

The third algorithm (executes the Collapsed Union) is defined by Cafezeiro as follows:

```
1. For all component x of the alignment mediator:
2.     add a new component y to the coalignment mediator
3.     create a link from the image of x by the left link to y
4.     create a link from the image of x by the right link to y
5. For all component x that are not in the image of the left or right links
   from the alignment mediator:
6.     add a new component y to the coalignment mediator
7.     create a link from x to y
8. Perform the union of axioms, with the proper translations.
```

The forth algorithm is used to compute the Relative Intersection operation but can be also modified to execute the Entity Integration operation. By doing contextualizing each entity or context in it self (By using an identity link, making both ontologies of the pair of contextualized ontologies the same ontologies).

```
1. For all component x of the coalignment mediator that is in the image of
   the left and right links:
2.      add a new component y to the alignment mediator
3.      create a link from y to the domain of x by the left link
4.      create a link from y to the domain of x by the right link
5. Perform the intersection of the disjunction of axioms, with the proper
   translations.
```

11 Appendix - Monitoring Ontology

Classes

Action

Action $\sqsubseteq \neg$ AtributeChecker

Action $\sqsubseteq \neg$ ComplexConditionOrContextActivity

Action $\sqsubseteq \neg$ Monitoring

Action $\sqsubseteq \neg$ PersonToMonitor

Action $\sqsubseteq \neg$ ResponsablePerson

Action $\sqsubseteq \neg$ State

Action $\sqsubseteq \neg$ StateActivity

Action $\sqsubseteq \neg$ AtributeChecker

ComplexConditionOrContextActivity $\sqsubseteq \neg$ AtributeChecker

Monitoring $\sqsubseteq \neg$ AtributeChecker

PersonToMonitor $\sqsubseteq \neg$ AtributeChecker

ResponsablePerson $\sqsubseteq \neg$ AtributeChecker

State $\sqsubseteq \neg$ AtributeChecker

StateActivity $\sqsubseteq \neg$ AtributeChecker

Action $\sqsubseteq \neg$ ComplexConditionOrContextActivity

AtributeChecker $\sqsubseteq \neg$ ComplexConditionOrContextActivity

Monitoring $\sqsubseteq \neg$ ComplexConditionOrContextActivity

PersonToMonitor $\sqsubseteq \neg$ ComplexConditionOrContextActivity

ResponsablePerson $\sqsubseteq \neg$ ComplexConditionOrContextActivity

State $\sqsubseteq \neg$ ComplexConditionOrContextActivity

StateActivity $\sqsubseteq \neg$ ComplexConditionOrContextActivity

Action $\sqsubseteq \neg$ Monitoring

AtributeChecker $\sqsubseteq \neg$ Monitoring

ComplexConditionOrContextActivity $\sqsubseteq \neg$ Monitoring

PersonToMonitor $\sqsubseteq \neg$ Monitoring

ResponsablePerson $\sqsubseteq \neg$ Monitoring

State $\sqsubseteq \neg$ Monitoring

StateActivity $\sqsubseteq \neg$ Monitoring
Action $\sqsubseteq \neg$ PersonToMonitor
AtributeChecker $\sqsubseteq \neg$ PersonToMonitor
ComplexConditionOrContextActivity $\sqsubseteq \neg$ PersonToMonitor
Monitoring $\sqsubseteq \neg$ PersonToMonitor
ResponsablePerson $\sqsubseteq \neg$ PersonToMonitor
State $\sqsubseteq \neg$ PersonToMonitor
StateActivity $\sqsubseteq \neg$ PersonToMonitor
Action $\sqsubseteq \neg$ ResponsablePerson
AtributeChecker $\sqsubseteq \neg$ ResponsablePerson
ComplexConditionOrContextActivity $\sqsubseteq \neg$ ResponsablePerson
Monitoring $\sqsubseteq \neg$ ResponsablePerson
PersonToMonitor $\sqsubseteq \neg$ ResponsablePerson
State $\sqsubseteq \neg$ ResponsablePerson
StateActivity $\sqsubseteq \neg$ ResponsablePerson
Action $\sqsubseteq \neg$ State
AtributeChecker $\sqsubseteq \neg$ State
ComplexConditionOrContextActivity $\sqsubseteq \neg$ State
Monitoring $\sqsubseteq \neg$ State
PersonToMonitor $\sqsubseteq \neg$ State
ResponsablePerson $\sqsubseteq \neg$ State
StateActivity $\sqsubseteq \neg$ State
Action $\sqsubseteq \neg$ StateActivity
AtributeChecker $\sqsubseteq \neg$ StateActivity
ComplexConditionOrContextActivity $\sqsubseteq \neg$ StateActivity
Monitoring $\sqsubseteq \neg$ StateActivity
PersonToMonitor $\sqsubseteq \neg$ StateActivity
ResponsablePerson $\sqsubseteq \neg$ StateActivity
State $\sqsubseteq \neg$ StateActivity

Action_AlertToPatient

Action_AlertToPatient $\sqsubseteq \exists$ has_message Datatype: `http://www.w3.org/2001/XMLSchema#string`

Action_AlertToPatient $\sqsubseteq$ Action

Action_AlertToPatient $\sqsubseteq \neg$ Action_Question

Action_AlertToPatient $\sqsubseteq \neg$ Action_SMS
Action_AlertToPatient $\sqsubseteq \neg$ Action_Question
Action_SMS $\sqsubseteq \neg$ Action_Question

Action_AlertToPatient $\sqsubseteq \neg$ Action_SMS
Action_Question $\sqsubseteq \neg$ Action_SMS

Action_Question

Action_Question $\sqsubseteq$ Action
Action_Question $\sqsubseteq \neg$ Action_AlertToPatient
Action_SMS $\sqsubseteq \neg$ Action_AlertToPatient
Action_Question $\sqsubseteq \neg$ Action_AlertToPatient
Action_Question $\sqsubseteq \neg$ Action_SMS
Action_AlertToPatient $\sqsubseteq \neg$ Action_SMS
Action_Question $\sqsubseteq \neg$ Action_SMS

Action_SMS

Action_SMS $\sqsubseteq \exists$ has_personToContact ResponsablePerson
Action_SMS $\sqsubseteq$ Action
Action_Question $\sqsubseteq \neg$ Action_AlertToPatient
Action_SMS $\sqsubseteq \neg$ Action_AlertToPatient
Action_AlertToPatient $\sqsubseteq \neg$ Action_Question
Action_SMS $\sqsubseteq \neg$ Action_Question
Action_SMS $\sqsubseteq \neg$ Action_AlertToPatient
Action_SMS $\sqsubseteq \neg$ Action_Question

And_ConditionOrContextActivity

And_ConditionOrContextActivity $\sqsubseteq$ ComplexConditionOrContextActivity
And_ConditionOrContextActivity $\sqsubseteq \neg$ Or_ConditionOrContextActivity

Answer_Monitoring

AtributeChecker

AtributeChecker $\sqsubseteq =$ has_propertyUri
AtributeChecker $\sqsubseteq =$ has_targetConcept Thing
AtributeChecker $\sqsubseteq =$ has_targetCharacteristic Characteristic
AtributeChecker $\sqsubseteq \neg$ Action
ComplexConditionOrContextActivity $\sqsubseteq \neg$ Action

Monitoring $\sqsubseteq \neg$ Action

PersonToMonitor $\sqsubseteq \neg$ Action

ResponsablePerson $\sqsubseteq \neg$ Action

State $\sqsubseteq \neg$ Action

StateActivity $\sqsubseteq \neg$ Action

AtributeChecker $\sqsubseteq \neg$ Action

AtributeChecker $\sqsubseteq \neg$ ComplexConditionOrContextActivity

AtributeChecker $\sqsubseteq \neg$ Monitoring

AtributeChecker $\sqsubseteq \neg$ PersonToMonitor

AtributeChecker $\sqsubseteq \neg$ ResponsablePerson

AtributeChecker $\sqsubseteq \neg$ State

AtributeChecker $\sqsubseteq \neg$ StateActivity

Action $\sqsubseteq \neg$ ComplexConditionOrContextActivity

AtributeChecker $\sqsubseteq \neg$ ComplexConditionOrContextActivity

Monitoring $\sqsubseteq \neg$ ComplexConditionOrContextActivity

PersonToMonitor $\sqsubseteq \neg$ ComplexConditionOrContextActivity

ResponsablePerson $\sqsubseteq \neg$ ComplexConditionOrContextActivity

State $\sqsubseteq \neg$ ComplexConditionOrContextActivity

StateActivity $\sqsubseteq \neg$ ComplexConditionOrContextActivity

Action $\sqsubseteq \neg$ Monitoring

AtributeChecker $\sqsubseteq \neg$ Monitoring

ComplexConditionOrContextActivity $\sqsubseteq \neg$ Monitoring

PersonToMonitor $\sqsubseteq \neg$ Monitoring

ResponsablePerson $\sqsubseteq \neg$ Monitoring

State $\sqsubseteq \neg$ Monitoring

StateActivity $\sqsubseteq \neg$ Monitoring

Action $\sqsubseteq \neg$ PersonToMonitor

AtributeChecker $\sqsubseteq \neg$ PersonToMonitor

ComplexConditionOrContextActivity $\sqsubseteq \neg$ PersonToMonitor

Monitoring $\sqsubseteq \neg$ PersonToMonitor

ResponsablePerson $\sqsubseteq \neg$ PersonToMonitor

State $\sqsubseteq \neg$ PersonToMonitor

StateActivity $\sqsubseteq \neg$ PersonToMonitor

Action $\sqsubseteq \neg$ ResponsablePerson

AtributeChecker $\sqsubseteq \neg$ ResponsablePerson

ComplexConditionOrContextActivity $\sqsubseteq \neg$ ResponsablePerson

Monitoring $\sqsubseteq \neg$ ResponsablePerson

PersonToMonitor $\sqsubseteq \neg$ ResponsablePerson

State $\sqsubseteq \neg$ ResponsablePerson

StateActivity $\sqsubseteq \neg$ ResponsablePerson
Action $\sqsubseteq \neg$ State
AtributeChecker $\sqsubseteq \neg$ State
ComplexConditionOrContextActivity $\sqsubseteq \neg$ State
Monitoring $\sqsubseteq \neg$ State
PersonToMonitor $\sqsubseteq \neg$ State
ResponsablePerson $\sqsubseteq \neg$ State
StateActivity $\sqsubseteq \neg$ State
Action $\sqsubseteq \neg$ StateActivity
AtributeChecker $\sqsubseteq \neg$ StateActivity
ComplexConditionOrContextActivity $\sqsubseteq \neg$ StateActivity
Monitoring $\sqsubseteq \neg$ StateActivity
PersonToMonitor $\sqsubseteq \neg$ StateActivity
ResponsablePerson $\sqsubseteq \neg$ StateActivity
State $\sqsubseteq \neg$ StateActivity

BehaviorActivity

BehaviorActivity $\sqsubseteq =$ is_responsableFor Condition
BehaviorActivity $\sqsubseteq$ StateActivity
BehaviorActivity $\sqsubseteq \neg$ ConditionOrContextActivity

Characteristic

ComplexConditionOrContextActivity

ComplexConditionOrContextActivity $\sqsubseteq \neg$ ConditionOrContextActivity
AtributeChecker $\sqsubseteq \neg$ Action
ComplexConditionOrContextActivity $\sqsubseteq \neg$ Action
Monitoring $\sqsubseteq \neg$ Action
PersonToMonitor $\sqsubseteq \neg$ Action
ResponsablePerson $\sqsubseteq \neg$ Action
State $\sqsubseteq \neg$ Action
StateActivity $\sqsubseteq \neg$ Action
Action $\sqsubseteq \neg$ AtributeChecker
ComplexConditionOrContextActivity $\sqsubseteq \neg$ AtributeChecker
Monitoring $\sqsubseteq \neg$ AtributeChecker
PersonToMonitor $\sqsubseteq \neg$ AtributeChecker
ResponsablePerson $\sqsubseteq \neg$ AtributeChecker
State $\sqsubseteq \neg$ AtributeChecker

$\text{StateActivity} \sqsubseteq \neg\ \text{AtributeChecker}$

$\text{ComplexConditionOrContextActivity} \sqsubseteq \neg\ \text{Action}$

$\text{ComplexConditionOrContextActivity} \sqsubseteq \neg\ \text{AtributeChecker}$

$\text{ComplexConditionOrContextActivity} \sqsubseteq \neg\ \text{Monitoring}$

$\text{ComplexConditionOrContextActivity} \sqsubseteq \neg\ \text{PersonToMonitor}$

$\text{ComplexConditionOrContextActivity} \sqsubseteq \neg\ \text{ResponsablePerson}$

$\text{ComplexConditionOrContextActivity} \sqsubseteq \neg\ \text{State}$

$\text{ComplexConditionOrContextActivity} \sqsubseteq \neg\ \text{StateActivity}$

$\text{Action} \sqsubseteq \neg\ \text{Monitoring}$

$\text{AtributeChecker} \sqsubseteq \neg\ \text{Monitoring}$

$\text{ComplexConditionOrContextActivity} \sqsubseteq \neg\ \text{Monitoring}$

$\text{PersonToMonitor} \sqsubseteq \neg\ \text{Monitoring}$

$\text{ResponsablePerson} \sqsubseteq \neg\ \text{Monitoring}$

$\text{State} \sqsubseteq \neg\ \text{Monitoring}$

$\text{StateActivity} \sqsubseteq \neg\ \text{Monitoring}$

$\text{Action} \sqsubseteq \neg\ \text{PersonToMonitor}$

$\text{AtributeChecker} \sqsubseteq \neg\ \text{PersonToMonitor}$

$\text{ComplexConditionOrContextActivity} \sqsubseteq \neg\ \text{PersonToMonitor}$

$\text{Monitoring} \sqsubseteq \neg\ \text{PersonToMonitor}$

$\text{ResponsablePerson} \sqsubseteq \neg\ \text{PersonToMonitor}$

$\text{State} \sqsubseteq \neg\ \text{PersonToMonitor}$

$\text{StateActivity} \sqsubseteq \neg\ \text{PersonToMonitor}$

$\text{Action} \sqsubseteq \neg\ \text{ResponsablePerson}$

$\text{AtributeChecker} \sqsubseteq \neg\ \text{ResponsablePerson}$

$\text{ComplexConditionOrContextActivity} \sqsubseteq \neg\ \text{ResponsablePerson}$

$\text{Monitoring} \sqsubseteq \neg\ \text{ResponsablePerson}$

$\text{PersonToMonitor} \sqsubseteq \neg\ \text{ResponsablePerson}$

$\text{State} \sqsubseteq \neg\ \text{ResponsablePerson}$

$\text{StateActivity} \sqsubseteq \neg\ \text{ResponsablePerson}$

$\text{Action} \sqsubseteq \neg\ \text{State}$

$\text{AtributeChecker} \sqsubseteq \neg\ \text{State}$

$\text{ComplexConditionOrContextActivity} \sqsubseteq \neg\ \text{State}$

$\text{Monitoring} \sqsubseteq \neg\ \text{State}$

$\text{PersonToMonitor} \sqsubseteq \neg\ \text{State}$

$\text{ResponsablePerson} \sqsubseteq \neg\ \text{State}$

$\text{StateActivity} \sqsubseteq \neg\ \text{State}$

$\text{Action} \sqsubseteq \neg\ \text{StateActivity}$

$\text{AtributeChecker} \sqsubseteq \neg\ \text{StateActivity}$

$\text{ComplexConditionOrContextActivity} \sqsubseteq \neg\ \text{StateActivity}$

Monitoring $\sqsubseteq \neg$ StateActivity
PersonToMonitor $\sqsubseteq \neg$ StateActivity
ResponsablePerson $\sqsubseteq \neg$ StateActivity
State $\sqsubseteq \neg$ StateActivity

Condition

ConditionOrContextActivity

ConditionOrContextActivity $\sqsubseteq$ StateActivity
ConditionOrContextActivity $\sqsubseteq \neg$ ComplexConditionOrContextActivity
ConditionOrContextActivity $\sqsubseteq \neg$ BehaviorActivity

ExternalCondition

ExternalCondition $\sqsubseteq$ Condition

InternalCondition

InternalCondition $\sqsubseteq$ Condition

Monitoring

Monitoring $\sqsubseteq =$ has_currentState State
Monitoring $\sqsubseteq \geq 1$ has_state State
Monitoring $\sqsubseteq =$ has_patientToMonitor PersonToMonitor
AtributeChecker $\sqsubseteq \neg$ Action
ComplexConditionOrContextActivity $\sqsubseteq \neg$ Action
Monitoring $\sqsubseteq \neg$ Action
PersonToMonitor $\sqsubseteq \neg$ Action
ResponsablePerson $\sqsubseteq \neg$ Action
State $\sqsubseteq \neg$ Action
StateActivity $\sqsubseteq \neg$ Action
Action $\sqsubseteq \neg$ AtributeChecker
ComplexConditionOrContextActivity $\sqsubseteq \neg$ AtributeChecker
Monitoring $\sqsubseteq \neg$ AtributeChecker
PersonToMonitor $\sqsubseteq \neg$ AtributeChecker
ResponsablePerson $\sqsubseteq \neg$ AtributeChecker
State $\sqsubseteq \neg$ AtributeChecker
StateActivity $\sqsubseteq \neg$ AtributeChecker
Action $\sqsubseteq \neg$ ComplexConditionOrContextActivity
AtributeChecker $\sqsubseteq \neg$ ComplexConditionOrContextActivity

Monitoring $\sqsubseteq \neg$ ComplexConditionOrContextActivity
PersonToMonitor $\sqsubseteq \neg$ ComplexConditionOrContextActivity
ResponsablePerson $\sqsubseteq \neg$ ComplexConditionOrContextActivity
State $\sqsubseteq \neg$ ComplexConditionOrContextActivity
StateActivity $\sqsubseteq \neg$ ComplexConditionOrContextActivity
Monitoring $\sqsubseteq \neg$ Action
Monitoring $\sqsubseteq \neg$ AtributeChecker
Monitoring $\sqsubseteq \neg$ ComplexConditionOrContextActivity
Monitoring $\sqsubseteq \neg$ PersonToMonitor
Monitoring $\sqsubseteq \neg$ ResponsablePerson
Monitoring $\sqsubseteq \neg$ State
Monitoring $\sqsubseteq \neg$ StateActivity
Action $\sqsubseteq \neg$ PersonToMonitor
AtributeChecker $\sqsubseteq \neg$ PersonToMonitor
ComplexConditionOrContextActivity $\sqsubseteq \neg$ PersonToMonitor
Monitoring $\sqsubseteq \neg$ PersonToMonitor
ResponsablePerson $\sqsubseteq \neg$ PersonToMonitor
State $\sqsubseteq \neg$ PersonToMonitor
StateActivity $\sqsubseteq \neg$ PersonToMonitor
Action $\sqsubseteq \neg$ ResponsablePerson
AtributeChecker $\sqsubseteq \neg$ ResponsablePerson
ComplexConditionOrContextActivity $\sqsubseteq \neg$ ResponsablePerson
Monitoring $\sqsubseteq \neg$ ResponsablePerson
PersonToMonitor $\sqsubseteq \neg$ ResponsablePerson
State $\sqsubseteq \neg$ ResponsablePerson
StateActivity $\sqsubseteq \neg$ ResponsablePerson
Action $\sqsubseteq \neg$ State
AtributeChecker $\sqsubseteq \neg$ State
ComplexConditionOrContextActivity $\sqsubseteq \neg$ State
Monitoring $\sqsubseteq \neg$ State
PersonToMonitor $\sqsubseteq \neg$ State
ResponsablePerson $\sqsubseteq \neg$ State
StateActivity $\sqsubseteq \neg$ State
Action $\sqsubseteq \neg$ StateActivity
AtributeChecker $\sqsubseteq \neg$ StateActivity
ComplexConditionOrContextActivity $\sqsubseteq \neg$ StateActivity
Monitoring $\sqsubseteq \neg$ StateActivity
PersonToMonitor $\sqsubseteq \neg$ StateActivity
ResponsablePerson $\sqsubseteq \neg$ StateActivity

State $\sqsubseteq \neg$ StateActivity

NumberValueConditionActivity

NumberValueConditionActivity $\sqsubseteq$ ConditionOrContextActivity

NumberValueConditionActivity $\sqsubseteq =$ is_responsableFor Condition

NumberValueConditionActivity $\sqsubseteq \neg$ SerieValueConditionActivity

NumberValueConditionActivity $\sqsubseteq \neg$ TextValueConditionActivity

NumberValueConditionActivity $\sqsubseteq \neg$ SerieValueConditionActivity

TextValueConditionActivity $\sqsubseteq \neg$ SerieValueConditionActivity

NumberValueConditionActivity $\sqsubseteq \neg$ TextValueConditionActivity

SerieValueConditionActivity $\sqsubseteq \neg$ TextValueConditionActivity

Or_ConditionOrContextActivity

Or_ConditionOrContextActivity $\sqsubseteq$ ComplexConditionOrContextActivity

Or_ConditionOrContextActivity $\sqsubseteq \neg$ And_ConditionOrContextActivity

PersonToMonitor

PersonToMonitor $\sqsubseteq \neg$ ResponsablePerson

AtributeChecker $\sqsubseteq \neg$ Action

ComplexConditionOrContextActivity $\sqsubseteq \neg$ Action

Monitoring $\sqsubseteq \neg$ Action

PersonToMonitor $\sqsubseteq \neg$ Action

ResponsablePerson $\sqsubseteq \neg$ Action

State $\sqsubseteq \neg$ Action

StateActivity $\sqsubseteq \neg$ Action

Action $\sqsubseteq \neg$ AtributeChecker

ComplexConditionOrContextActivity $\sqsubseteq \neg$ AtributeChecker

Monitoring $\sqsubseteq \neg$ AtributeChecker

PersonToMonitor $\sqsubseteq \neg$ AtributeChecker

ResponsablePerson $\sqsubseteq \neg$ AtributeChecker

State $\sqsubseteq \neg$ AtributeChecker

StateActivity $\sqsubseteq \neg$ AtributeChecker

Action $\sqsubseteq \neg$ ComplexConditionOrContextActivity

AtributeChecker $\sqsubseteq \neg$ ComplexConditionOrContextActivity

Monitoring $\sqsubseteq \neg$ ComplexConditionOrContextActivity

PersonToMonitor $\sqsubseteq \neg$ ComplexConditionOrContextActivity

ResponsablePerson $\sqsubseteq \neg$ ComplexConditionOrContextActivity

State $\sqsubseteq \neg$ ComplexConditionOrContextActivity

StateActivity $\sqsubseteq \neg$ ComplexConditionOrContextActivity

Action $\sqsubseteq \neg$ Monitoring

AtributeChecker $\sqsubseteq \neg$ Monitoring

ComplexConditionOrContextActivity $\sqsubseteq \neg$ Monitoring

PersonToMonitor $\sqsubseteq \neg$ Monitoring

ResponsablePerson $\sqsubseteq \neg$ Monitoring

State $\sqsubseteq \neg$ Monitoring

StateActivity $\sqsubseteq \neg$ Monitoring

PersonToMonitor $\sqsubseteq \neg$ Action

PersonToMonitor $\sqsubseteq \neg$ AtributeChecker

PersonToMonitor $\sqsubseteq \neg$ ComplexConditionOrContextActivity

PersonToMonitor $\sqsubseteq \neg$ Monitoring

PersonToMonitor $\sqsubseteq \neg$ ResponsablePerson

PersonToMonitor $\sqsubseteq \neg$ State

PersonToMonitor $\sqsubseteq \neg$ StateActivity

Action $\sqsubseteq \neg$ ResponsablePerson

AtributeChecker $\sqsubseteq \neg$ ResponsablePerson

ComplexConditionOrContextActivity $\sqsubseteq \neg$ ResponsablePerson

Monitoring $\sqsubseteq \neg$ ResponsablePerson

PersonToMonitor $\sqsubseteq \neg$ ResponsablePerson

State $\sqsubseteq \neg$ ResponsablePerson

StateActivity $\sqsubseteq \neg$ ResponsablePerson

Action $\sqsubseteq \neg$ State

AtributeChecker $\sqsubseteq \neg$ State

ComplexConditionOrContextActivity $\sqsubseteq \neg$ State

Monitoring $\sqsubseteq \neg$ State

PersonToMonitor $\sqsubseteq \neg$ State

ResponsablePerson $\sqsubseteq \neg$ State

StateActivity $\sqsubseteq \neg$ State

Action $\sqsubseteq \neg$ StateActivity

AtributeChecker $\sqsubseteq \neg$ StateActivity

ComplexConditionOrContextActivity $\sqsubseteq \neg$ StateActivity

Monitoring $\sqsubseteq \neg$ StateActivity

PersonToMonitor $\sqsubseteq \neg$ StateActivity

ResponsablePerson $\sqsubseteq \neg$ StateActivity

State $\sqsubseteq \neg$ StateActivity

Question

ResponsablePerson

ResponsablePerson $\sqsubseteq \neg$ PersonToMonitor
AtributeChecker $\sqsubseteq \neg$ Action
ComplexConditionOrContextActivity $\sqsubseteq \neg$ Action
Monitoring $\sqsubseteq \neg$ Action
PersonToMonitor $\sqsubseteq \neg$ Action
ResponsablePerson $\sqsubseteq \neg$ Action
State $\sqsubseteq \neg$ Action
StateActivity $\sqsubseteq \neg$ Action
Action $\sqsubseteq \neg$ AtributeChecker
ComplexConditionOrContextActivity $\sqsubseteq \neg$ AtributeChecker
Monitoring $\sqsubseteq \neg$ AtributeChecker
PersonToMonitor $\sqsubseteq \neg$ AtributeChecker
ResponsablePerson $\sqsubseteq \neg$ AtributeChecker
State $\sqsubseteq \neg$ AtributeChecker
StateActivity $\sqsubseteq \neg$ AtributeChecker
Action $\sqsubseteq \neg$ ComplexConditionOrContextActivity
AtributeChecker $\sqsubseteq \neg$ ComplexConditionOrContextActivity
Monitoring $\sqsubseteq \neg$ ComplexConditionOrContextActivity
PersonToMonitor $\sqsubseteq \neg$ ComplexConditionOrContextActivity
ResponsablePerson $\sqsubseteq \neg$ ComplexConditionOrContextActivity
State $\sqsubseteq \neg$ ComplexConditionOrContextActivity
StateActivity $\sqsubseteq \neg$ ComplexConditionOrContextActivity
Action $\sqsubseteq \neg$ Monitoring
AtributeChecker $\sqsubseteq \neg$ Monitoring
ComplexConditionOrContextActivity $\sqsubseteq \neg$ Monitoring
PersonToMonitor $\sqsubseteq \neg$ Monitoring
ResponsablePerson $\sqsubseteq \neg$ Monitoring
State $\sqsubseteq \neg$ Monitoring
StateActivity $\sqsubseteq \neg$ Monitoring
Action $\sqsubseteq \neg$ PersonToMonitor
AtributeChecker $\sqsubseteq \neg$ PersonToMonitor
ComplexConditionOrContextActivity $\sqsubseteq \neg$ PersonToMonitor
Monitoring $\sqsubseteq \neg$ PersonToMonitor
ResponsablePerson $\sqsubseteq \neg$ PersonToMonitor
State $\sqsubseteq \neg$ PersonToMonitor
StateActivity $\sqsubseteq \neg$ PersonToMonitor

ResponsablePerson $\sqsubseteq \neg$ Action
ResponsablePerson $\sqsubseteq \neg$ AtributeChecker
ResponsablePerson $\sqsubseteq \neg$ ComplexConditionOrContextActivity
ResponsablePerson $\sqsubseteq \neg$ Monitoring
ResponsablePerson $\sqsubseteq \neg$ PersonToMonitor
ResponsablePerson $\sqsubseteq \neg$ State
ResponsablePerson $\sqsubseteq \neg$ StateActivity
Action $\sqsubseteq \neg$ State
AtributeChecker $\sqsubseteq \neg$ State
ComplexConditionOrContextActivity $\sqsubseteq \neg$ State
Monitoring $\sqsubseteq \neg$ State
PersonToMonitor $\sqsubseteq \neg$ State
ResponsablePerson $\sqsubseteq \neg$ State
StateActivity $\sqsubseteq \neg$ State
Action $\sqsubseteq \neg$ StateActivity
AtributeChecker $\sqsubseteq \neg$ StateActivity
ComplexConditionOrContextActivity $\sqsubseteq \neg$ StateActivity
Monitoring $\sqsubseteq \neg$ StateActivity
PersonToMonitor $\sqsubseteq \neg$ StateActivity
ResponsablePerson $\sqsubseteq \neg$ StateActivity
State $\sqsubseteq \neg$ StateActivity

SerieValueConditionActivity

SerieValueConditionActivity $\sqsubseteq =$ is_responsableFor Condition
SerieValueConditionActivity $\sqsubseteq$ ConditionOrContextActivity
SerieValueConditionActivity $\sqsubseteq \neg$ NumberValueConditionActivity
TextValueConditionActivity $\sqsubseteq \neg$ NumberValueConditionActivity
SerieValueConditionActivity $\sqsubseteq \neg$ NumberValueConditionActivity
SerieValueConditionActivity $\sqsubseteq \neg$ TextValueConditionActivity
NumberValueConditionActivity $\sqsubseteq \neg$ TextValueConditionActivity
SerieValueConditionActivity $\sqsubseteq \neg$ TextValueConditionActivity

State

State $\sqsubseteq \exists$ has_complexVariable (And_ConditionOrContextActivity $\sqcup$ Or_ConditionOrContextA
berValueConditionActivity $\sqcup$ SerieValueConditionActivity $\sqcup$ TextValueConditionActivity) $\sqcup$ $\exists$ has_contextVariable (NumberValueConditionActivity $\sqcup$ SerieValueConditionActivity $\sqcup$ TextValueConditionActivity)

$\text{State} \sqsubseteq \exists\ \text{has\_possibleTransition State}$
$\text{State} \sqsubseteq =\ \text{has\_priority}$
$\text{AtributeChecker} \sqsubseteq \neg\ \text{Action}$
$\text{ComplexConditionOrContextActivity} \sqsubseteq \neg\ \text{Action}$
$\text{Monitoring} \sqsubseteq \neg\ \text{Action}$
$\text{PersonToMonitor} \sqsubseteq \neg\ \text{Action}$
$\text{ResponsablePerson} \sqsubseteq \neg\ \text{Action}$
$\text{State} \sqsubseteq \neg\ \text{Action}$
$\text{StateActivity} \sqsubseteq \neg\ \text{Action}$
$\text{Action} \sqsubseteq \neg\ \text{AtributeChecker}$
$\text{ComplexConditionOrContextActivity} \sqsubseteq \neg\ \text{AtributeChecker}$
$\text{Monitoring} \sqsubseteq \neg\ \text{AtributeChecker}$
$\text{PersonToMonitor} \sqsubseteq \neg\ \text{AtributeChecker}$
$\text{ResponsablePerson} \sqsubseteq \neg\ \text{AtributeChecker}$
$\text{State} \sqsubseteq \neg\ \text{AtributeChecker}$
$\text{StateActivity} \sqsubseteq \neg\ \text{AtributeChecker}$
$\text{Action} \sqsubseteq \neg\ \text{ComplexConditionOrContextActivity}$
$\text{AtributeChecker} \sqsubseteq \neg\ \text{ComplexConditionOrContextActivity}$
$\text{Monitoring} \sqsubseteq \neg\ \text{ComplexConditionOrContextActivity}$
$\text{PersonToMonitor} \sqsubseteq \neg\ \text{ComplexConditionOrContextActivity}$
$\text{ResponsablePerson} \sqsubseteq \neg\ \text{ComplexConditionOrContextActivity}$
$\text{State} \sqsubseteq \neg\ \text{ComplexConditionOrContextActivity}$
$\text{StateActivity} \sqsubseteq \neg\ \text{ComplexConditionOrContextActivity}$
$\text{Action} \sqsubseteq \neg\ \text{Monitoring}$
$\text{AtributeChecker} \sqsubseteq \neg\ \text{Monitoring}$
$\text{ComplexConditionOrContextActivity} \sqsubseteq \neg\ \text{Monitoring}$
$\text{PersonToMonitor} \sqsubseteq \neg\ \text{Monitoring}$
$\text{ResponsablePerson} \sqsubseteq \neg\ \text{Monitoring}$
$\text{State} \sqsubseteq \neg\ \text{Monitoring}$
$\text{StateActivity} \sqsubseteq \neg\ \text{Monitoring}$
$\text{Action} \sqsubseteq \neg\ \text{PersonToMonitor}$
$\text{AtributeChecker} \sqsubseteq \neg\ \text{PersonToMonitor}$
$\text{ComplexConditionOrContextActivity} \sqsubseteq \neg\ \text{PersonToMonitor}$
$\text{Monitoring} \sqsubseteq \neg\ \text{PersonToMonitor}$
$\text{ResponsablePerson} \sqsubseteq \neg\ \text{PersonToMonitor}$
$\text{State} \sqsubseteq \neg\ \text{PersonToMonitor}$
$\text{StateActivity} \sqsubseteq \neg\ \text{PersonToMonitor}$
$\text{Action} \sqsubseteq \neg\ \text{ResponsablePerson}$
$\text{AtributeChecker} \sqsubseteq \neg\ \text{ResponsablePerson}$

ComplexConditionOrContextActivity $\sqsubseteq \neg$ ResponsablePerson

Monitoring $\sqsubseteq \neg$ ResponsablePerson

PersonToMonitor $\sqsubseteq \neg$ ResponsablePerson

State $\sqsubseteq \neg$ ResponsablePerson

StateActivity $\sqsubseteq \neg$ ResponsablePerson

State $\sqsubseteq \neg$ Action

State $\sqsubseteq \neg$ AtributeChecker

State $\sqsubseteq \neg$ ComplexConditionOrContextActivity

State $\sqsubseteq \neg$ Monitoring

State $\sqsubseteq \neg$ PersonToMonitor

State $\sqsubseteq \neg$ ResponsablePerson

State $\sqsubseteq \neg$ StateActivity

Action $\sqsubseteq \neg$ StateActivity

AtributeChecker $\sqsubseteq \neg$ StateActivity

ComplexConditionOrContextActivity $\sqsubseteq \neg$ StateActivity

Monitoring $\sqsubseteq \neg$ StateActivity

PersonToMonitor $\sqsubseteq \neg$ StateActivity

ResponsablePerson $\sqsubseteq \neg$ StateActivity

State $\sqsubseteq \neg$ StateActivity

StateActivity

AtributeChecker $\sqsubseteq \neg$ Action

ComplexConditionOrContextActivity $\sqsubseteq \neg$ Action

Monitoring $\sqsubseteq \neg$ Action

PersonToMonitor $\sqsubseteq \neg$ Action

ResponsablePerson $\sqsubseteq \neg$ Action

State $\sqsubseteq \neg$ Action

StateActivity $\sqsubseteq \neg$ Action

Action $\sqsubseteq \neg$ AtributeChecker

ComplexConditionOrContextActivity $\sqsubseteq \neg$ AtributeChecker

Monitoring $\sqsubseteq \neg$ AtributeChecker

PersonToMonitor $\sqsubseteq \neg$ AtributeChecker

ResponsablePerson $\sqsubseteq \neg$ AtributeChecker

State $\sqsubseteq \neg$ AtributeChecker

StateActivity $\sqsubseteq \neg$ AtributeChecker

Action $\sqsubseteq \neg$ ComplexConditionOrContextActivity

AtributeChecker $\sqsubseteq \neg$ ComplexConditionOrContextActivity

Monitoring $\sqsubseteq \neg$ ComplexConditionOrContextActivity

$\text{PersonToMonitor} \sqsubseteq \neg\ \text{ComplexConditionOrContextActivity}$

$\text{ResponsablePerson} \sqsubseteq \neg\ \text{ComplexConditionOrContextActivity}$

$\text{State} \sqsubseteq \neg\ \text{ComplexConditionOrContextActivity}$

$\text{StateActivity} \sqsubseteq \neg\ \text{ComplexConditionOrContextActivity}$

$\text{Action} \sqsubseteq \neg\ \text{Monitoring}$

$\text{AtributeChecker} \sqsubseteq \neg\ \text{Monitoring}$

$\text{ComplexConditionOrContextActivity} \sqsubseteq \neg\ \text{Monitoring}$

$\text{PersonToMonitor} \sqsubseteq \neg\ \text{Monitoring}$

$\text{ResponsablePerson} \sqsubseteq \neg\ \text{Monitoring}$

$\text{State} \sqsubseteq \neg\ \text{Monitoring}$

$\text{StateActivity} \sqsubseteq \neg\ \text{Monitoring}$

$\text{Action} \sqsubseteq \neg\ \text{PersonToMonitor}$

$\text{AtributeChecker} \sqsubseteq \neg\ \text{PersonToMonitor}$

$\text{ComplexConditionOrContextActivity} \sqsubseteq \neg\ \text{PersonToMonitor}$

$\text{Monitoring} \sqsubseteq \neg\ \text{PersonToMonitor}$

$\text{ResponsablePerson} \sqsubseteq \neg\ \text{PersonToMonitor}$

$\text{State} \sqsubseteq \neg\ \text{PersonToMonitor}$

$\text{StateActivity} \sqsubseteq \neg\ \text{PersonToMonitor}$

$\text{Action} \sqsubseteq \neg\ \text{ResponsablePerson}$

$\text{AtributeChecker} \sqsubseteq \neg\ \text{ResponsablePerson}$

$\text{ComplexConditionOrContextActivity} \sqsubseteq \neg\ \text{ResponsablePerson}$

$\text{Monitoring} \sqsubseteq \neg\ \text{ResponsablePerson}$

$\text{PersonToMonitor} \sqsubseteq \neg\ \text{ResponsablePerson}$

$\text{State} \sqsubseteq \neg\ \text{ResponsablePerson}$

$\text{StateActivity} \sqsubseteq \neg\ \text{ResponsablePerson}$

$\text{Action} \sqsubseteq \neg\ \text{State}$

$\text{AtributeChecker} \sqsubseteq \neg\ \text{State}$

$\text{ComplexConditionOrContextActivity} \sqsubseteq \neg\ \text{State}$

$\text{Monitoring} \sqsubseteq \neg\ \text{State}$

$\text{PersonToMonitor} \sqsubseteq \neg\ \text{State}$

$\text{ResponsablePerson} \sqsubseteq \neg\ \text{State}$

$\text{StateActivity} \sqsubseteq \neg\ \text{State}$

$\text{StateActivity} \sqsubseteq \neg\ \text{Action}$

$\text{StateActivity} \sqsubseteq \neg\ \text{AtributeChecker}$

$\text{StateActivity} \sqsubseteq \neg\ \text{ComplexConditionOrContextActivity}$

$\text{StateActivity} \sqsubseteq \neg\ \text{Monitoring}$

$\text{StateActivity} \sqsubseteq \neg\ \text{PersonToMonitor}$

$\text{StateActivity} \sqsubseteq \neg\ \text{ResponsablePerson}$

$\text{StateActivity} \sqsubseteq \neg\ \text{State}$

TextValue

TextValueConditionActivity

$\text{TextValueConditionActivity} \sqsubseteq = \text{is\_responsableFor Condition}$
$\text{TextValueConditionActivity} \sqsubseteq \text{ConditionOrContextActivity}$
$\text{SerieValueConditionActivity} \sqsubseteq \neg\ \text{NumberValueConditionActivity}$
$\text{TextValueConditionActivity} \sqsubseteq \neg\ \text{NumberValueConditionActivity}$
$\text{NumberValueConditionActivity} \sqsubseteq \neg\ \text{SerieValueConditionActivity}$
$\text{TextValueConditionActivity} \sqsubseteq \neg\ \text{SerieValueConditionActivity}$
$\text{TextValueConditionActivity} \sqsubseteq \neg\ \text{NumberValueConditionActivity}$
$\text{TextValueConditionActivity} \sqsubseteq \neg\ \text{SerieValueConditionActivity}$

Thing

Object properties

has_action

$\exists\ \text{has\_action Thing} \sqsubseteq \text{State}$
$\top \sqsubseteq \forall\ \text{has\_action Action}$

has_attributeVariable

$\exists\ \text{has\_attributeVariable Thing} \sqsubseteq \text{State}$
$\top \sqsubseteq \forall\ \text{has\_attributeVariable AtributeChecker}$

has_behaviorActivity

$\exists\ \text{has\_behaviorActivity Thing} \sqsubseteq \text{State}$
$\top \sqsubseteq \forall\ \text{has\_behaviorActivity BehaviorActivity}$

has_complexActivity

$\exists\ \text{has\_complexActivity Thing} \sqsubseteq \text{ComplexConditionOrContextActivity}$
$\top \sqsubseteq \forall\ \text{has\_complexActivity ComplexConditionOrContextActivity}$

has_complexVariable

$\exists\ \text{has\_complexVariable Thing} \sqsubseteq \text{State}$
$\top \sqsubseteq \forall\ \text{has\_complexVariable ComplexConditionOrContextActivity}$

has_conditionActivity

$\exists$ has_conditionActivity Thing $\sqsubseteq$ ComplexConditionOrContextActivity

$\top \sqsubseteq \forall$ has_conditionActivity ConditionOrContextActivity

has_conditionVariable

$\exists$ has_conditionVariable Thing $\sqsubseteq$ State

$\top \sqsubseteq \forall$ has_conditionVariable ConditionOrContextActivity

has_contextVariable

$\exists$ has_contextVariable Thing $\sqsubseteq$ State

$\top \sqsubseteq \forall$ has_contextVariable ConditionOrContextActivity

has_currentState

$\top \sqsubseteq \leq 1$ has_currentState Thing

$\exists$ has_currentState Thing $\sqsubseteq$ Monitoring

$\top \sqsubseteq \forall$ has_currentState State

has_patientToMonitor

$\exists$ has_patientToMonitor Thing $\sqsubseteq$ Monitoring

$\top \sqsubseteq \forall$ has_patientToMonitor PersonToMonitor

has_personToContact

$\exists$ has_personToContact Thing $\sqsubseteq$ Action_SMS

$\top \sqsubseteq \forall$ has_personToContact ResponsablePerson

has_possibleCurrentsStates

$\exists$ has_possibleCurrentsStates Thing $\sqsubseteq$ Monitoring

$\top \sqsubseteq \forall$ has_possibleCurrentsStates State

has_possibleTransition

$\exists$ has_possibleTransition Thing $\sqsubseteq$ State

$\top \sqsubseteq \forall$ has_possibleTransition State

has_question

$\exists$ has_question Thing $\sqsubseteq$ Action_Question

$\top \sqsubseteq \forall$ has_question Question

has_state

$\exists$ has_state Thing $\sqsubseteq$ Monitoring

$\top \sqsubseteq \forall$ has_state State

has_targetCharacteristic

$\top \sqsubseteq \leq 1$ has_targetCharacteristic Thing

$\exists$ has_targetCharacteristic Thing $\sqsubseteq$ AtributeChecker

$\top \sqsubseteq \forall$ has_targetCharacteristic Characteristic

has_targetConcept

$\top \sqsubseteq \leq 1$ has_targetConcept Thing

$\exists$ has_targetConcept Thing $\sqsubseteq$ AtributeChecker

$\top \sqsubseteq \forall$ has_targetConcept PersonToMonitor

has_targetCondition

$\top \sqsubseteq \leq 1$ has_targetCondition Thing

$\top \sqsubseteq \forall$ has_targetCondition Condition

has_textEqualTo

$\exists$ has_textEqualTo Thing $\sqsubseteq$ TextValueConditionActivity

$\top \sqsubseteq \forall$ has_textEqualTo Characteristic

$\top \sqsubseteq \forall$ has_textEqualTo Answer_Monitoring

has_textNotEqualTo

$\exists$ has_textNotEqualTo Thing $\sqsubseteq$ TextValueConditionActivity

$\top \sqsubseteq \forall$ has_textNotEqualTo Characteristic

$\top \sqsubseteq \forall$ has_textNotEqualTo Answer_Monitoring

has_value

$\top \sqsubseteq \leq 1$ has_value Thing

$\exists$ has_value Thing $\sqsubseteq$ Condition

is_responsableFor

$\top \sqsubseteq \leq 1$ is_responsableFor Thing

$\exists$ is_responsableFor Thing $\sqsubseteq$ StateActivity

$\top \sqsubseteq \forall$ is_responsableFor Condition

Data properties

has_constantTolerancy

$\top \sqsubseteq \leq 1$ has_constantTolerancy

$\exists$ has_constantTolerancy Datatype: `http://www.w3.org/2000/01/rdf-schema#Literal`$\sqsubseteq$ SerieValueConditionActivity

$\top \sqsubseteq \forall$ has_constantTolerancy Datatype: `http://www.w3.org/2001/XMLSchema#float`

has_message

$\top \sqsubseteq \leq 1$ has_message

$\exists$ has_message Datatype: `http://www.w3.org/2000/01/rdf-schema#Literal`$\sqsubseteq$ Action

$\top \sqsubseteq \forall$ has_message Datatype: `http://www.w3.org/2001/XMLSchema#string`

has_numberType

$\top \sqsubseteq \leq 1$ has_numberType

$\exists$ has_numberType Datatype: `http://www.w3.org/2000/01/rdf-schema#Literal`$\sqsubseteq$ Condition

$\top \sqsubseteq \forall$ has_numberType Datatype: `http://www.w3.org/2001/XMLSchema#boolean`

has_priority

$\top \sqsubseteq \leq 1$ has_priority

$\exists$ has_priority Datatype: `http://www.w3.org/2000/01/rdf-schema#Literal`$\sqsubseteq$ State

$\top \sqsubseteq \forall$ has_priority Datatype: `http://www.w3.org/2001/XMLSchema#int`

has_propertyUri

$\top \sqsubseteq \leq 1$ has_propertyUri

$\exists$ has_propertyUri Datatype: `http://www.w3.org/2000/01/rdf-schema#Literal` $\sqsubseteq$ AtributeChecker

$\top \sqsubseteq \forall$ has_propertyUri Datatype: `http://www.w3.org/2001/XMLSchema#string`

has_recordFrequency

$\top \sqsubseteq \leq 1$ has_recordFrequency

$\exists$ has_recordFrequency Datatype: `http://www.w3.org/2000/01/rdf-schema#Literal` $\sqsubseteq$ BehaviorActivity

$\top \sqsubseteq \forall$ has_recordFrequency Datatype: `http://www.w3.org/2001/XMLSchema#float`

has_recordPeriod

$\top \sqsubseteq \leq 1$ has_recordPeriod

$\exists$ has_recordPeriod Datatype: `http://www.w3.org/2000/01/rdf-schema#Literal` $\sqsubseteq$ BehaviorActivity

$\top \sqsubseteq \forall$ has_recordPeriod Datatype: `http://www.w3.org/2001/XMLSchema#float`

has_serieType

$\top \sqsubseteq \leq 1$ has_serieType

$\exists$ has_serieType Datatype: `http://www.w3.org/2000/01/rdf-schema#Literal` $\sqsubseteq$ Condition

$\top \sqsubseteq \forall$ has_serieType Datatype: `http://www.w3.org/2001/XMLSchema#boolean`

has_stateName

$\exists$ has_stateName Datatype: `http://www.w3.org/2000/01/rdf-schema#Literal` $\sqsubseteq$ State

$\top \sqsubseteq \forall$ has_stateName Datatype: `http://www.w3.org/2001/XMLSchema#string`

has_textType

$\top \sqsubseteq \leq 1$ has_textType

$\exists$ has_textType Datatype: `http://www.w3.org/2000/01/rdf-schema#Literal` $\sqsubseteq$ Condition

$\top \sqsubseteq \forall$ has_textType Datatype: `http://www.w3.org/2001/XMLSchema#boolean`

has_variationEqualTo

$\top \sqsubseteq \leq 1$ has_variationEqualTo

$\exists$ has_variationEqualTo Datatype: `http://www.w3.org/2000/01/rdf-schema#Literal` $\sqsubseteq$ SerieValueConditionActivity

$\top \sqsubseteq \forall$ has_variationEqualTo Datatype: `http://www.w3.org/2001/XMLSchema#float`

has_variationGreaterOrEqualThen

$\top \sqsubseteq \leq 1$ has_variationGreaterOrEqualThen

$\exists$ has_variationGreaterOrEqualThen Datatype: `http://www.w3.org/2000/01/rdf-schema#Literal`$\sqsubseteq$ SerieValueConditionActivity

$\top \sqsubseteq \forall$ has_variationGreaterOrEqualThen Datatype: `http://www.w3.org/2001/XMLSchema#float`

has_variationGreaterThen

$\top \sqsubseteq \leq 1$ has_variationGreaterThen

$\exists$ has_variationGreaterThen Datatype: `http://www.w3.org/2000/01/rdf-schema#Literal`$\sqsubseteq$ SerieValueConditionActivity

$\top \sqsubseteq \forall$ has_variationGreaterThen Datatype: `http://www.w3.org/2001/XMLSchema#float`

has_variationInterval

$\top \sqsubseteq \leq 1$ has_variationInterval

$\exists$ has_variationInterval Datatype: `http://www.w3.org/2000/01/rdf-schema#Literal`$\sqsubseteq$ SerieValueConditionActivity

$\top \sqsubseteq \forall$ has_variationInterval Datatype: `http://www.w3.org/2001/XMLSchema#integer`

has_variationSmallerOrEqualThen

$\top \sqsubseteq \leq 1$ has_variationSmallerOrEqualThen

$\exists$ has_variationSmallerOrEqualThen Datatype: `http://www.w3.org/2000/01/rdf-schema#Literal`$\sqsubseteq$ SerieValueConditionActivity

$\top \sqsubseteq \forall$ has_variationSmallerOrEqualThen Datatype: `http://www.w3.org/2001/XMLSchema#float`

has_variationSmallerThen

$\top \sqsubseteq \leq 1$ has_variationSmallerThen

$\exists$ has_variationSmallerThen Datatype: `http://www.w3.org/2000/01/rdf-schema#Literal`$\sqsubseteq$ SerieValueConditionActivity

$\top \sqsubseteq \forall$ has_variationSmallerThen Datatype: `http://www.w3.org/2001/XMLSchema#float`

is_active

$\top \sqsubseteq \leq 1$ is_active

$\exists$ is_active Datatype: `http://www.w3.org/2000/01/rdf-schema#Literal`$\sqsubseteq$ Monitoring

$\top \sqsubseteq \forall$ is_active Datatype: `http://www.w3.org/2001/XMLSchema#boolean`

is_constant

$\top \sqsubseteq \leq 1$ is_constant

$\exists$ is_constant Datatype: `http://www.w3.org/2000/01/rdf-schema#Literal`$\sqsubseteq$ SerieValueConditionActivity

$\top \sqsubseteq \forall$ is_constant Datatype: `http://www.w3.org/2001/XMLSchema#boolean`

is_equalTo

$\top \sqsubseteq \leq 1$ is_equalTo

$\exists$ is_equalTo Datatype: `http://www.w3.org/2000/01/rdf-schema#Literal`$\sqsubseteq$ NumberValueConditionActivity

$\top \sqsubseteq \forall$ is_equalTo Datatype: `http://www.w3.org/2001/XMLSchema#float`

is_greaterOrEqualThen

$\top \sqsubseteq \leq 1$ is_greaterOrEqualThen

$\exists$ is_greaterOrEqualThen Datatype: `http://www.w3.org/2000/01/rdf-schema#Literal`$\sqsubseteq$ NumberValueConditionActivity

$\top \sqsubseteq \forall$ is_greaterOrEqualThen Datatype: `http://www.w3.org/2001/XMLSchema#float`

is_greaterThen

$\top \sqsubseteq \leq 1$ is_greaterThen

$\exists$ is_greaterThen Datatype: `http://www.w3.org/2000/01/rdf-schema#Literal`$\sqsubseteq$ NumberValueConditionActivity

$\top \sqsubseteq \forall$ is_greaterThen Datatype: `http://www.w3.org/2001/XMLSchema#float`

is_recordFrequencyOnChange

$\top \sqsubseteq \leq 1$ is_recordFrequencyOnChange

$\exists$ is_recordFrequencyOnChange Datatype: `http://www.w3.org/2000/01/rdf-schema#Literal`$\sqsubseteq$ BehaviorActivity

$\top \sqsubseteq \forall$ is_recordFrequencyOnChange Datatype: `http://www.w3.org/2001/XMLSchema#boolean`

is_smallerOrEqualThen

$\top \sqsubseteq \leq 1$ is_smallerOrEqualThen

$\exists$ is_smallerOrEqualThen Datatype: `http://www.w3.org/2000/01/rdf-schema#Literal`$\sqsubseteq$ NumberValueConditionActivity

$\top \sqsubseteq \forall$ is_smallerOrEqualThen Datatype: `http://www.w3.org/2001/XMLSchema#float`

is_smallerThen

$\top \sqsubseteq \leq 1$ is_smallerThen

$\exists$ is_smallerThen Datatype `http://www.w3.org/2000/01/rdf-schema#Literal`$\sqsubseteq$ NumberValueConditionActivity

$\top \sqsubseteq \forall$ is_smallerThen Datatype: `http://www.w3.org/2001/XMLSchema#float`

Individuals

Activity

Activity : InternalCondition

has_textType (Activity "true"{=`http://www.w3.org/2001/XMLSchema#boolean`)

Alert_AltitudeAndAnticoagulant

Alert_AltitudeAndAnticoagulant : Action_AlertToPatient

has_message (Alert_AltitudeAndAnticoagulant "Descer imediatamente para uma altitude menor que 2400m. Por você fazer uso de (anti-coagulante ou anti-plaquetário), caso você comece a sangrar, o sangramento pode não parar."{=`http://www.w3.org/2001/XMLSchema#string`)

Alert_AltitudeAndDangersSymptoms

Alert_AltitudeAndDangersSymptoms : Action_AlertToPatient

has_message (Alert_AltitudeAndDangersSymptoms ”É possível que você esteja sofrendo de sintomas da altitude. Por ser vomitos ou febre, é considerado grave. Procure descer para uma altitude menor que 2400m assim que puder.”{=`http://www.w3.org/2001/XMLSchema#string`)

Alert_HealthPlanCoverage

Alert_HealthPlanCoverage : Action_AlertToPatient

has_message (Alert_HealthPlanCoverage ”Lembre-se de verificar se o seu plano de saÃºde cobre o estado que você se encontra.”{=`http://www.w3.org/2001/XMLSchema#string`)

Alert_SymptomQuestionAnswered

Alert_SymptomQuestionAnswered : Action_AlertToPatient

has_message (Alert_SymptomQuestionAnswered ”Obrigado por nos falar sobre o que você esta sentindo! Caso você não tenha conseguido falar sobre seu sintoma atravÃ©s das perguntas feitas, envie uma mensagem de texto para o seu mÃ©dico. Isto será importante para o seu acompanhamento. Seu mÃ©dico receberá sua informação.”{=`http://www.w3.org/2001/XMLSchema#string`)

Altitude

Altitude : ExternalCondition

AltitudeGreaterThenNumber_contextVariable

AltitudeGreaterThenNumber_contextVariable : NumberValueCondition-Activity

is_responsableFor(AltitudeGreaterThenNumber_contextVariable, Altitude)

is_greaterOrEqualThen (AltitudeGreaterThenNumber_contextVariable ”2400”{=`http://www.w3.org/2001/XMLSchema#float`)

AnticoagulatedBlood

AnticoagulatedBlood : Characteristic

BodyTemperature

BodyTemperature : InternalCondition

BreathRate_1/1min

BreathRate_1/1min : BehaviorActivity
is_responsableFor(BreathRate_1/1min, Breath_Rate)
has_recordFrequency (BreathRate_1/1min ”1”{=`http://www.w3.org/2001/XMLSchema#float`)

BreathRate_10/10mins

BreathRate_10/10mins : BehaviorActivity
is_responsableFor(BreathRate_10/10mins, Breath_Rate)
has_recordFrequency (BreathRate_10/10mins ”10”{=`http://www.w3.org/2001/XMLSchema#float`)

BreathRate_5/5mins

BreathRate_5/5mins : BehaviorActivity
is_responsableFor(BreathRate_5/5mins, Breath_Rate)
has_recordFrequency (BreathRate_5/5mins ”5”{=`http://www.w3.org/2001/XMLSchema#float`)

Breath_Rate

Breath_Rate : InternalCondition

ChestPainQuestion

ChestPainQuestion : Question

Chest_Pain

Chest_Pain : InternalCondition

CountryCurrent

CountryCurrent : ExternalCondition

CriticalState1_conditionVariable

CriticalState1_conditionVariable : And_ConditionOrContextActivity

has_conditionActivity(CriticalState1_conditionVariable, IsNotDoingPhysicalActivity_conditionVariable)

has_conditionActivity(CriticalState1_conditionVariable, HeartRateGreaterThenNumber_conditionVariable)

CriticalState2_conditionVariable

CriticalState2_conditionVariable : SerieValueConditionActivity

is_responsableFor(CriticalState2_conditionVariable, RR)

has_variationGreaterThen (CriticalState2_conditionVariable "500"{=`http://www.w3.org/2001/XMLSchema#float`)

has_variationInterval (CriticalState2_conditionVariable "1"{=`http://www.w3.org/2001/XMLSchema#integer`)

is_constant (CriticalState2_conditionVariable "false"{=`http://www.w3.org/2001/XMLSchema#boolean`)

CriticalState3_conditionValue

CriticalState3_conditionValue : Or_ConditionOrContextActivity

has_conditionActivity(CriticalState3_conditionValue, Feeling_ChestPain_conditionVariable)

has_conditionActivity(CriticalState3_conditionValue, Feeling_Faint_conditionVariable)

has_conditionActivity(CriticalState3_conditionValue, Feeling_UnusualTiredness_conditionVaria

has_conditionActivity(CriticalState3_conditionValue, Feeling_Headache_conditionVariable)

has_conditionActivity(CriticalState3_conditionValue, Feeling_Palpitation_conditionVariable)

CriticalState4_attributeVariable

CriticalState4_attributeVariable : AtributeChecker

has_targetConcept(CriticalState4_attributeVariable, Vitor)

has_targetCharacteristic(CriticalState4_attributeVariable, AnticoagulatedBlood)

has_propertyUri (CriticalState4_attributeVariable "`http://patientbuddy/patient.owl#has_diseaseCharacteristic`"{=`http://www.w3.org/2001/XMLSchema#string`)

CriticalState5_conditionVariable

CriticalState5_conditionVariable : And_ConditionOrContextActivity

has_complexActivity(CriticalState5_conditionVariable, PatientWithVomitOrFever_conditionVariable) has_conditionActivity(CriticalState5_conditionVariable, Feeling_Headache_conditionVariable)

Critical_State_1

Critical_State_1 : State
has_behaviorActivity(Critical_State_1, PatientActivity_OnChange)
has_behaviorActivity(Critical_State_1, BreathRate_5/5mins)
has_possibleTransition(Critical_State_1, Physical_Activity_State)
has_action(Critical_State_1, MedicalAssistence_Immediate_SMS)
has_possibleTransition(Critical_State_1, Intermediary_State)
has_complexVariable(Critical_State_1, CriticalState1_conditionVariable)
has_possibleTransition(Critical_State_1, Critical_State_3)
has_behaviorActivity(Critical_State_1, ECG_10/10mins_10secs)
has_behaviorActivity(Critical_State_1, HeartRate_5/5mins)
has_possibleTransition(Critical_State_1, Normal_State)
has_attributeVariable(Critical_State_1, IsPermanentPatient_attributeVariable)
has_possibleTransition(Critical_State_1, Traveling_State)
has_priority (Critical_State_1 "8"{=`http://www.w3.org/2001/XMLSchema#int`)
has_stateName (Critical_State_1 "Estado Crítico 1"{=`http://www.w3.org/2001/XMLSchema#string`)

Critical_State_2

Critical_State_2 : State
has_attributeVariable(Critical_State_2, IsParaxysmalPatient_attributeVariable)
has_possibleTransition(Critical_State_2, Intermediary_State)
has_behaviorActivity(Critical_State_2, BreathRate_5/5mins)
has_possibleTransition(Critical_State_2, Physical_Activity_State)
has_conditionVariable(Critical_State_2, CriticalState2_conditionVariable)
has_behaviorActivity(Critical_State_2, ECG_10/10mins_10secs)
has_possibleTransition(Critical_State_2, Critical_State_3)
has_behaviorActivity(Critical_State_2, PatientActivity_OnChange)
has_action(Critical_State_2, MedicalAssistence_UpTo6hrs_SMS)
has_behaviorActivity(Critical_State_2, HeartRate_5/5mins)
has_possibleTransition(Critical_State_2, Normal_State)
has_possibleTransition(Critical_State_2, Traveling_State)
has_stateName (Critical_State_2 "Estado Crítico 2"{=`http://www.w3.org/2001/XMLSchema#string`)

has_priority (Critical_State_2 ”8”{=http://www.w3.org/2001/XMLSchema#int)

Critical_State_3

Critical_State_3 : State
has_possibleTransition(Critical_State_3, Intermediary_State)
has_behaviorActivity(Critical_State_3, HeartRate_5/5mins)
has_complexVariable(Critical_State_3, CriticalState3_conditionValue)
has_possibleTransition(Critical_State_3, Critical_State_2)
has_possibleTransition(Critical_State_3, Physical_Activity_State)
has_action(Critical_State_3, Alert_SymptomQuestionAnswered)
has_possibleTransition(Critical_State_3, Critical_State_1)
has_possibleTransition(Critical_State_3, Traveling_State)
has_behaviorActivity(Critical_State_3, ECG_10/10mins_10secs)
has_possibleTransition(Critical_State_3, Normal_State)
has_behaviorActivity(Critical_State_3, BreathRate_5/5mins)
has_behaviorActivity(Critical_State_3, PatientActivity_OnChange)
has_priority (Critical_State_3 ”7”{=http://www.w3.org/2001/XMLSchema#int)
has_stateName (Critical_State_3 ”Estado Crítico 3”{=http://www.w3.org/2001/XMLSchema#string)

Critical_State_4

Critical_State_4 : State
has_behaviorActivity(Critical_State_4, PatientActivity_OnChange)
has_possibleTransition(Critical_State_4, Critical_State_3)
has_attributeVariable(Critical_State_4, CriticalState4_attributeVariable)
has_possibleTransition(Critical_State_4, Critical_State_2)
has_possibleTransition(Critical_State_4, Critical_State_1)
has_possibleTransition(Critical_State_4, Intermediary_State)
has_possibleTransition(Critical_State_4, Critical_State_5)
has_behaviorActivity(Critical_State_4, ECG_10/10mins_10secs)
has_possibleTransition(Critical_State_4, Normal_State)
has_action(Critical_State_4, Alert_AltitudeAndAnticoagulant)
has_possibleTransition(Critical_State_4, Traveling_State)
has_contextVariable(Critical_State_4, AltitudeGreaterThenNumber_contextVariable)
has_behaviorActivity(Critical_State_4, BreathRate_5/5mins)
has_behaviorActivity(Critical_State_4, HeartRate_5/5mins)

has_possibleTransition(Critical_State_4, Physical_Activity_State)

has_priority (Critical_State_4 "6"{=`http://www.w3.org/2001/XMLSchema#int`)

has_stateName (Critical_State_4 "Estado Crítico 4"{=`http://www.w3.org/2001/XMLSchema#string`)

Critical_State_5

Critical_State_5 : State

has_behaviorActivity(Critical_State_5, ECG_10/10mins_10secs)

has_possibleTransition(Critical_State_5, Critical_State_4)

has_possibleTransition(Critical_State_5, Critical_State_3)

has_contextVariable(Critical_State_5, AltitudeGreaterThenNumber_contextVariable)

has_behaviorActivity(Critical_State_5, HeartRate_5/5mins)

has_possibleTransition(Critical_State_5, Critical_State_2)

has_possibleTransition(Critical_State_5, Intermediary_State)

has_action(Critical_State_5, Alert_AltitudeAndDangersSymptoms)

has_possibleTransition(Critical_State_5, Critical_State_1)

has_behaviorActivity(Critical_State_5, BreathRate_5/5mins)

has_complexVariable(Critical_State_5, CriticalState5_conditionVariable)

has_possibleTransition(Critical_State_5, Normal_State)

has_possibleTransition(Critical_State_5, Physical_Activity_State)

has_behaviorActivity(Critical_State_5, PatientActivity_OnChange)

has_possibleTransition(Critical_State_5, Traveling_State)

has_priority (Critical_State_5 "5"{=`http://www.w3.org/2001/XMLSchema#int`)

has_stateName (Critical_State_5 "Estado Crítico 5"{=`http://www.w3.org/2001/XMLSchema#string`)

CurrentCity

CurrentCity : ExternalCondition

Delta_FC

Delta_FC : InternalCondition

ECG_10/10mins_10secs

ECG_10/10mins_10secs : BehaviorActivity
is_responsableFor(ECG_10/10mins_10secs, ECG_Signal)
has_recordPeriod (ECG_10/10mins_10secs "10"{=`http://www.w3.org/2001/XMLSchema#float`)
has_recordFrequency (ECG_10/10mins_10secs "10"{=`http://www.w3.org/2001/XMLSchema#float`)

ECG_3/3hrs_10secs

ECG_3/3hrs_10secs : BehaviorActivity
is_responsableFor(ECG_3/3hrs_10secs, ECG_Signal)
has_recordFrequency (ECG_3/3hrs_10secs "180"{=`http://www.w3.org/2001/XMLSchema#float`)
has_recordPeriod (ECG_3/3hrs_10secs "10"{=`http://www.w3.org/2001/XMLSchema#float`)

ECG_6/6hrs_10secs

ECG_6/6hrs_10secs : BehaviorActivity
is_responsableFor(ECG_6/6hrs_10secs, ECG_Signal)
has_recordPeriod (ECG_6/6hrs_10secs "10"{=`http://www.w3.org/2001/XMLSchema#float`)
has_recordFrequency (ECG_6/6hrs_10secs "360"{=`http://www.w3.org/2001/XMLSchema#float`)

ECG_For5mins

ECG_For5mins : BehaviorActivity
is_responsableFor(ECG_For5mins, ECG_Signal)
has_recordPeriod (ECG_For5mins "300"{=`http://www.w3.org/2001/XMLSchema#float`)

ECG_Signal

ECG_Signal : InternalCondition
has_serieType (ECG_Signal "true"{=`http://www.w3.org/2001/XMLSchema#boolean`)

EndOfRun_textContextInformation

EndOfRun_textContextInformation : TextValue

FC

FC : InternalCondition

Faint

Faint : InternalCondition

FaintQuestion

FaintQuestion : Question

Feeling_ChestPain_conditionVariable

Feeling_ChestPain_conditionVariable : TextValueConditionActivity
is_responsableFor(Feeling_ChestPain_conditionVariable, Chest_Pain)
has_textNotEqualTo(Feeling_ChestPain_conditionVariable, None)

Feeling_Faint_conditionVariable

Feeling_Faint_conditionVariable : TextValueConditionActivity
is_responsableFor(Feeling_Faint_conditionVariable, Faint)
has_textNotEqualTo(Feeling_Faint_conditionVariable, No)

Feeling_Fever_conditionVariable

Feeling_Fever_conditionVariable : NumberValueConditionActivity
is_responsableFor(Feeling_Fever_conditionVariable, BodyTemperature)
is_greaterOrEqualThen (Feeling_Fever_conditionVariable "37"{=`http://www.w3.org/2001/XMLSchema#float`)

Feeling_Headache_conditionVariable

Feeling_Headache_conditionVariable : TextValueConditionActivity
has_textNotEqualTo(Feeling_Headache_conditionVariable, No)
is_responsableFor(Feeling_Headache_conditionVariable, Headache)

Feeling_Palpitation_conditionVariable

Feeling_Palpitation_conditionVariable : TextValueConditionActivity
is_responsableFor(Feeling_Palpitation_conditionVariable, Palpitation)
has_textNotEqualTo(Feeling_Palpitation_conditionVariable, No)

Feeling_UnusualTiredness_conditionVariable

Feeling_UnusualTiredness_conditionVariable : TextValueConditionActivity

is_responsableFor(Feeling_UnusualTiredness_conditionVariable, Unusual_Tiredness)

has_textNotEqualTo(Feeling_UnusualTiredness_conditionVariable, No)

Feeling_Vomit_conditionVariable

Feeling_Vomit_conditionVariable : TextValueConditionActivity

is_responsableFor(Feeling_Vomit_conditionVariable, Vomiting)

has_textNotEqualTo(Feeling_Vomit_conditionVariable, No)

Flat_textContextInformation

Flat_textContextInformation : TextValue

Frequently

Frequently : Answer_Monitoring

Headache

Headache : InternalCondition

HeadacheQuestion

HeadacheQuestion : Question

HeartRateGreaterThenNumber_conditionVariable

HeartRateGreaterThenNumber_conditionVariable : NumberValueConditionActivity

is_responsableFor(HeartRateGreaterThenNumber_conditionVariable, Heart_Rate)

is_greaterOrEqualThen (HeartRateGreaterThenNumber_conditionVariable "100"{=`http://www.w3.org/2001/XMLSchema#float`)

HeartRate_1/1min

HeartRate_1/1min : BehaviorActivity

is_responsableFor(HeartRate_1/1min, Heart_Rate)

has_recordFrequency (HeartRate_1/1min "1"{=`http://www.w3.org/2001/XMLSchema#float`)

HeartRate_10/10mins

HeartRate_10/10mins : BehaviorActivity
is_responsableFor(HeartRate_10/10mins, Heart_Rate)
has_recordFrequency (HeartRate_10/10mins "10"{=http://www.w3.org/2001/XMLSchema#float)

HeartRate_5/5mins

HeartRate_5/5mins : BehaviorActivity
is_responsableFor(HeartRate_5/5mins, Heart_Rate)
has_recordFrequency (HeartRate_5/5mins "5"{=http://www.w3.org/2001/XMLSchema#float)

HeartRate_6/6hrs

HeartRate_6/6hrs : BehaviorActivity
is_responsableFor(HeartRate_6/6hrs, Heart_Rate)
has_recordFrequency (HeartRate_6/6hrs "360"{=http://www.w3.org/2001/XMLSchema#float)

Heart_Rate

Heart_Rate : InternalCondition

Intense

Intense : Answer_Monitoring

Intermediary_State

Intermediary_State : State
has_possibleTransition(Intermediary_State, Critical_State_4)
has_possibleTransition(Intermediary_State, Physical_Activity_State)
has_possibleTransition(Intermediary_State, Critical_State_2)
has_possibleTransition(Intermediary_State, Critical_State_1)
has_behaviorActivity(Intermediary_State, PatientActivity_OnChange)
has_behaviorActivity(Intermediary_State, BreathRate_10/10mins)
has_contextVariable(Intermediary_State, AltitudeGreaterThenNumber_contextVariable)
has_possibleTransition(Intermediary_State, Traveling_State)
has_behaviorActivity(Intermediary_State, ECG_3/3hrs_10secs)
has_possibleTransition(Intermediary_State, Critical_State_5)
has_behaviorActivity(Intermediary_State, HeartRate_10/10mins)

has_possibleTransition(Intermediary_State, Normal_State)

has_possibleTransition(Intermediary_State, Critical_State_3)

has_priority (Intermediary_State "2"{=`http://www.w3.org/2001/XMLSchema#int`)

has_stateName (Intermediary_State "Estado Intermediário"{=`http://www.w3.org/2001/XMLSchema#string`)

IsDoingPhysicalActivity_conditionVariable

IsDoingPhysicalActivity_conditionVariable : TextValueConditionActivity

has_textEqualTo(IsDoingPhysicalActivity_conditionVariable, Walking_textContextInformation)

has_textEqualTo(IsDoingPhysicalActivity_conditionVariable, RunningHard_textContextInformation)

has_textEqualTo(IsDoingPhysicalActivity_conditionVariable, Running_textContextInformation)

is_responsableFor(IsDoingPhysicalActivity_conditionVariable, Activity)

has_textEqualTo(IsDoingPhysicalActivity_conditionVariable, WalkingHard_textContextInformation)

IsNotDoingPhysicalActivity_conditionVariable

IsNotDoingPhysicalActivity_conditionVariable : TextValueConditionActivity

is_responsableFor(IsNotDoingPhysicalActivity_conditionVariable, Activity)

has_textEqualTo(IsNotDoingPhysicalActivity_conditionVariable, Flat_textContextInformation)

has_textEqualTo(IsNotDoingPhysicalActivity_conditionVariable, NotMoving_textContextInformation)

IsParaxysmalPatient_attributeVariable

IsParaxysmalPatient_attributeVariable : AtributeChecker

has_targetConcept(IsParaxysmalPatient_attributeVariable, Vitor)

has_targetCharacteristic(IsParaxysmalPatient_attributeVariable, Paroxysmal_Patient)

has_propertyUri (IsParaxysmalPatient_attributeVariable "`http://patientbuddy/patient.owl#has_diseaseCharacteristic`"{=`http://www.w3.org/2001/XMLSchema#string`)

IsPermanentPatient_attributeVariable

IsPermanentPatient_attributeVariable : AtributeChecker
has_targetConcept(IsPermanentPatient_attributeVariable, Vitor)
has_targetCharacteristic(IsPermanentPatient_attributeVariable, Permanent_Patient)
has_propertyUri (IsPermanentPatient_attributeVariable ”`http://patientbuddy/patient.owl#has_diseaseCharacteristic`”{=`http://www.w3.org/2001/XMLSchema#string`)

LackOfAirQuestion

LackOfAirQuestion : Question

Lack_of_Air

Lack_of_Air : InternalCondition

Little

Little : Answer_Monitoring

Mario

Mario : ResponsablePerson

MedicalAssistence_Immediate_SMS

MedicalAssistence_Immediate_SMS : Action_SMS
has_personToContact(MedicalAssistence_Immediate_SMS, Mario)
has_personToContact(MedicalAssistence_Immediate_SMS, Rodrigo)
has_message (MedicalAssistence_Immediate_SMS ”This patient needs immediate medical assistance.”{=`http://www.w3.org/2001/XMLSchema#string`)

MedicalAssistence_UpTo6hrs_SMS

MedicalAssistence_UpTo6hrs_SMS : Action_SMS
has_personToContact(MedicalAssistence_UpTo6hrs_SMS, Rodrigo)
has_personToContact(MedicalAssistence_UpTo6hrs_SMS, Mario)
has_message (MedicalAssistence_UpTo6hrs_SMS ”This patient need medical assistance in up to 6 hours.”{=`http://www.w3.org/2001/XMLSchema#string`)

MonitoringPatientVitor

MonitoringPatientVitor : Monitoring
has_currentState(MonitoringPatientVitor, Normal_State)
has_state(MonitoringPatientVitor, Critical_State_3)
has_state(MonitoringPatientVitor, Critical_State_4)
has_state(MonitoringPatientVitor, Critical_State_5)
has_state(MonitoringPatientVitor, Normal_State)
has_state(MonitoringPatientVitor, Critical_State_2)
has_patientToMonitor(MonitoringPatientVitor, Vitor)
has_state(MonitoringPatientVitor, Intermediary_State)
has_state(MonitoringPatientVitor, Traveling_State)
has_state(MonitoringPatientVitor, Critical_State_1)
has_state(MonitoringPatientVitor, Post_Physical_Activity_State)
has_state(MonitoringPatientVitor, Physical_Activity_State)
is_active (MonitoringPatientVitor ”true”{=`http://www.w3.org/2001/XMLSchema#boolean`)

No

No : Answer_Monitoring

None

None : Answer_Monitoring

Normal

Normal : Answer_Monitoring

Normal_State

Normal_State : State
has_behaviorActivity(Normal_State, HeartRate_6/6hrs)
has_possibleTransition(Normal_State, Physical_Activity_State)
has_possibleTransition(Normal_State, Intermediary_State)
has_behaviorActivity(Normal_State, PatientActivity_OnChange)
has_behaviorActivity(Normal_State, ECG_6/6hrs_10secs)
has_possibleTransition(Normal_State, Critical_State_1)
has_possibleTransition(Normal_State, Critical_State_2)
has_possibleTransition(Normal_State, Traveling_State)
has_possibleTransition(Normal_State, Critical_State_3)

has_stateName (Normal_State "Estado Normal"{=`http://www.w3.org/2001/XMLSchema#string`)

has_priority (Normal_State "0"{=`http://www.w3.org/2001/XMLSchema#int`)

NotMoving_textContextInformation

NotMoving_textContextInformation : TextValue

Palpitation

Palpitation : InternalCondition

PalpitationQuestion

PalpitationQuestion : Question

Paroxysmal_Patient

Paroxysmal_Patient : Characteristic

PatientActivity_OnChange

PatientActivity_OnChange : BehaviorActivity

is_responsableFor(PatientActivity_OnChange, Activity)

is_recordFrequencyOnChange (PatientActivity_OnChange "true"{=`http://www.w3.org/2001/XMLSchema#boolean`)

PatientOutsideCityOfLiving_contextVariable

PatientOutsideCityOfLiving_contextVariable : TextValueCondition-Activity

is_responsableFor(PatientOutsideCityOfLiving_contextVariable, CurrentCity)

PatientWithVomitOrFever_conditionVariable

PatientWithVomitOrFever_conditionVariable : Or_ConditionOrContextActivity

has_conditionActivity(PatientWithVomitOrFever_conditionVariable, Feeling_Vomit_conditionVariable)

has_conditionActivity(PatientWithVomitOrFever_conditionVariable, Feeling_Fever_conditionVariable)

Permanent_Patient

Permanent_Patient : Characteristic

Physical_Activity_State

Physical_Activity_State : State
has_possibleTransition(Physical_Activity_State, Critical_State_3)
has_behaviorActivity(Physical_Activity_State, ECG_10/10mins_10secs)
has_behaviorActivity(Physical_Activity_State, PatientActivity_OnChange)
has_conditionVariable(Physical_Activity_State, IsDoingPhysicalActivity_conditionVariable)
has_possibleTransition(Physical_Activity_State, Post_Physical_Activity_State)
has_possibleTransition(Physical_Activity_State, Critical_State_2)
has_behaviorActivity(Physical_Activity_State, BreathRate_5/5mins)
has_possibleTransition(Physical_Activity_State, Critical_State_1)
has_behaviorActivity(Physical_Activity_State, HeartRate_5/5mins)
has_stateName (Physical_Activity_State "Estado Atividade Física"{=`http://www.w3.org/2001/XMLSchema#string`)
has_priority (Physical_Activity_State "4"{=`http://www.w3.org/2001/XMLSchema#int`)

Post_Physical_Activity_State

Post_Physical_Activity_State : State
has_possibleTransition(Post_Physical_Activity_State, Traveling_State)
has_behaviorActivity(Post_Physical_Activity_State, BreathRate_1/1min)
has_possibleTransition(Post_Physical_Activity_State, Physical_Activity_State)
has_possibleTransition(Post_Physical_Activity_State, Normal_State)
has_conditionVariable(Post_Physical_Activity_State, isEndOfRunActivity_conditionVariable)
has_possibleTransition(Post_Physical_Activity_State, Critical_State_3)
has_behaviorActivity(Post_Physical_Activity_State, ECG_For5mins)
has_possibleTransition(Post_Physical_Activity_State, Critical_State_1)
has_possibleTransition(Post_Physical_Activity_State, Critical_State_2)
has_behaviorActivity(Post_Physical_Activity_State, PatientActivity_OnChange)
has_behaviorActivity(Post_Physical_Activity_State, HeartRate_1/1min)
has_possibleTransition(Post_Physical_Activity_State, Intermediary_State)

has_priority (Post_Physical_Activity_State "3"{=`http://www.w3.org/2001/XMLSchema#int`)

has_stateName (Post_Physical_Activity_State "Estado Pós Atividade Física"{=`http://www.w3.org/2001/XMLSchema#string`)

RR

RR : InternalCondition

Rodrigo

Rodrigo : ResponsablePerson

RunningHard_textContextInformation

RunningHard_textContextInformation : TextValue

Running_textContextInformation

Running_textContextInformation : TextValue

ST

ST : InternalCondition

SomeTimes

SomeTimes : Answer_Monitoring

Traveling_State

Traveling_State : State

has_contextVariable(Traveling_State, PatientOutsideCityOfLiving_contextVariable)

has_behaviorActivity(Traveling_State, PatientActivity_OnChange)

has_possibleTransition(Traveling_State, Physical_Activity_State)

has_behaviorActivity(Traveling_State, ECG_6/6hrs_10secs)

has_behaviorActivity(Traveling_State, HeartRate_6/6hrs)

has_possibleTransition(Traveling_State, Intermediary_State)

has_action(Traveling_State, Alert_HealthPlanCoverage)

has_complexVariable(Traveling_State, CriticalState3_conditionValue)

has_possibleTransition(Traveling_State, Normal_State)

has_conditionVariable(Traveling_State, CriticalState2_conditionVariable)

has_complexVariable(Traveling_State, CriticalState1_conditionVariable) has_priority (Traveling_State "1"{=`http://www.w3.org/2001/XMLSchema#int`)

has_stateName (Traveling_State "Estado Viajando"{=`http://www.w3.org/2001/XMLSchema#string`)

UnusualTirednessQuestion

UnusualTirednessQuestion : Question

Unusual_Tiredness

Unusual_Tiredness : InternalCondition

Very_Intense

Very_Intense : Answer_Monitoring

Vitor

Vitor : PersonToMonitor

VomitQuestion

VomitQuestion : Question

Vomiting

Vomiting : InternalCondition

WalkingHard_textContextInformation

WalkingHard_textContextInformation : TextValue

Walking_textContextInformation

Walking_textContextInformation : TextValue

Yes

Yes : Answer_Monitoring

isEndOfRunActivity_conditionVariable

isEndOfRunActivity_conditionVariable : TextValueConditionActivity

is_responsableFor(isEndOfRunActivity_conditionVariable, Activity)

has_textEqualTo(isEndOfRunActivity_conditionVariable, EndOfRun_textContextInformation)

Datatypes

PlainLiteral

boolean

float

int

integer

string

12
Appendix - Disease Ontology

Classes

Characteristic

$\text{Characteristic} \sqsubseteq \neg\ \text{Condition}$

$\text{Characteristic} \sqsubseteq \neg\ \text{Condition}$

$\text{Characteristic} \sqsubseteq \neg\ \text{Disease}$

$\text{Characteristic} \sqsubseteq \neg\ \text{Condition}$

$\text{Disease} \sqsubseteq \neg\ \text{Condition}$

$\text{Characteristic} \sqsubseteq \neg\ \text{Disease}$

$\text{Condition} \sqsubseteq \neg\ \text{Disease}$

Condition

$\text{Condition} \sqsubseteq \neg\ \text{Characteristic}$

$\text{Condition} \sqsubseteq \neg\ \text{Characteristic}$

$\text{Disease} \sqsubseteq \neg\ \text{Characteristic}$

$\text{Condition} \sqsubseteq \neg\ \text{Characteristic}$

$\text{Condition} \sqsubseteq \neg\ \text{Disease}$

$\text{Characteristic} \sqsubseteq \neg\ \text{Disease}$

$\text{Condition} \sqsubseteq \neg\ \text{Disease}$

Disease

$\text{Disease} \sqsubseteq =\ \text{diseaseName}$

$\text{Disease} \sqsubseteq \exists\ \text{has\_caracteristic Characteristic} \sqcup \exists\ \text{has\_condition Condition}$

$\text{Condition} \sqsubseteq \neg\ \text{Characteristic}$

$\text{Disease} \sqsubseteq \neg\ \text{Characteristic}$

$\text{Characteristic} \sqsubseteq \neg\ \text{Condition}$

$\text{Disease} \sqsubseteq \neg\ \text{Condition}$

Disease $\sqsubseteq \neg$ Characteristic
Disease $\sqsubseteq \neg$ Condition

ExternalCondition

ExternalCondition $\sqsubseteq$ Condition

InternalCondition

InternalCondition $\sqsubseteq$ Condition

Object properties

has_caracteristic

$\exists$ has_caracteristic Thing $\sqsubseteq$ Disease
$\top \sqsubseteq \forall$ has_caracteristic Characteristic

has_condition

$\exists$ has_condition Thing $\sqsubseteq$ Disease
$\top \sqsubseteq \forall$ has_condition Condition

Data properties

diseaseName

$\top \sqsubseteq \leq 1$ diseaseName
$\exists$ diseaseName Datatype `http://www.w3.org/2000/01/rdf-schema#Literal`$\sqsubseteq$ Disease
$\top \sqsubseteq \forall$ diseaseName Datatype `http://www.w3.org/2001/XMLSchema#string`

has_characteristicName

$\top \sqsubseteq \leq 1$ has_characteristicName
$\exists$ has_characteristicName Datatype: `http://www.w3.org/2000/01/rdf-schema#Literal`$\sqsubseteq$ Characteristic
$\top \sqsubseteq \forall$ has_characteristicName Datatype: `http://www.w3.org/2001/XMLSchema#string`

medical_area

∃ medical_area Datatype: `http://www.w3.org/2000/01/rdf-schema#Literal`⊑ Disease

⊤ ⊑ ∀ medical_area Datatype: `http://www.w3.org/2001/XMLSchema#string`

Individuals

Activity

Activity : InternalCondition

Altitude

Altitude : ExternalCondition

Altitude_Sickness

Altitude_Sickness : Disease
has_condition(Altitude_Sickness, Lack_of_Air)
has_condition(Altitude_Sickness, Headache)
has_condition(Altitude_Sickness, Vomiting)
has_condition(Altitude_Sickness, BodyTemperature)
has_condition(Altitude_Sickness, Altitude)
has_condition(Altitude_Sickness, Unusual_Tiredness)
diseaseName (Altitude_Sickness ”Altitude Sickness”{=`http://www.w3.org/2001/XMLSchema#string`)

AnticoagulatedBlood

AnticoagulatedBlood : Characteristic
has_characteristicName (AnticoagulatedBlood ”Sangue descoagulado”{=`http://www.w3.org/2001/XMLSchema#string`)

Atrial_fibrillation

Atrial_fibrillation : Disease
has_condition(Atrial_fibrillation, Faint)
has_condition(Atrial_fibrillation, Lack_of_Air)
has_condition(Atrial_fibrillation, Chest_Pain)
has_caracteristic(Atrial_fibrillation, Permanent_FA_Type)
has_condition(Atrial_fibrillation, Palpitation)

has_caracteristic(Atrial_fibrillation, Paroxysmal_FA_Type)
has_condition(Atrial_fibrillation, Unusual_Tiredness)
diseaseName (Atrial_fibrillation "Atrial Fibrillation"{=`http://www.w3.org/2001/XMLSchema#string`)
medical_area (Atrial_fibrillation "Cardiology"{=`http://www.w3.org/2001/XMLSchema#string`)

BodyTemperature

BodyTemperature : InternalCondition

Breath_Rate

Breath_Rate : InternalCondition

Chest_Pain

Chest_Pain : InternalCondition

Delta_FC

Delta_FC : InternalCondition

FC

FC : InternalCondition

Faint

Faint : InternalCondition

Headache

Headache : InternalCondition

Heart_Rate

Heart_Rate : InternalCondition

Lack_of_Air

Lack_of_Air : InternalCondition

Palpitation

Palpitation : InternalCondition

Paroxysmal_FA_Type

Paroxysmal_FA_Type : Characteristic

has_characteristicName (Paroxysmal_FA_Type "Pacientes Intermitentes"{=`http://www.w3.org/2001/XMLSchema#string`)

Permanent_FA_Type

Permanent_FA_Type : Characteristic

has_characteristicName (Permanent_FA_Type "Pacientes Permanentes"{=`http://www.w3.org/2001/XMLSchema#string`)

RR

RR : InternalCondition

ST

ST : InternalCondition

Unusual_Tiredness

Unusual_Tiredness : InternalCondition

Vomiting

Vomiting : InternalCondition

Datatypes

PlainLiteral

string

13
Appendix - Environment Ontology

Classes

Condition

Condition $\sqsubseteq$ = has_aquisitionAutomaticForm ContextProvider $\sqcup$ = has_aquisitionQuestionForm Question

ContextInformation

ContextInformation $\sqsubseteq$ = has_conditionRelated Condition
ContextInformation $\sqsubseteq$ $\neg$ ContextProvider

ContextInformation_NumberForm

ContextInformation_NumberForm $\sqsubseteq$ ContextInformation

ContextInformation_TextForm

ContextInformation_TextForm $\sqsubseteq$ ContextInformation

ContextProvider

ContextProvider $\sqsubseteq$ $\exists$ has_numberContextInformation ContextInformation_NumberForm $\sqcup$ $\exists$ has_textContextInformation ContextInformation_TextForm
ContextProvider $\sqsubseteq$ = has_information_class
ContextProvider $\sqsubseteq$ $\neg$ ContextInformation

ExternalCondition

ExternalCondition $\sqsubseteq$ Condition
ExternalCondition $\sqsubseteq$ $\neg$ InternalCondition

InternalCondition

$\text{InternalCondition} \sqsubseteq \text{Condition}$

$\text{InternalCondition} \sqsubseteq \neg\ \text{ExternalCondition}$

NumberValue

$\text{NumberValue} \sqsubseteq =\text{has\_number}$

Question

$\text{Question} \sqsubseteq =\text{has\_conditionRelated Condition}$

SerieValue

$\text{SerieValue} \sqsubseteq =\text{has\_number}$

TextValue

$\text{TextValue} \sqsubseteq =\text{has\_text}$

Object properties

has_aquisitionAutomaticForm

$\exists\ \text{has\_aquisitionAutomaticForm Thing} \sqsubseteq \text{Condition}$

$\top \sqsubseteq \forall\ \text{has\_aquisitionAutomaticForm ContextProvider}$

has_aquisitionQuestionForm

$\exists\ \text{has\_aquisitionQuestionForm Thing} \sqsubseteq \text{Condition}$

$\top \sqsubseteq \forall\ \text{has\_aquisitionQuestionForm Question}$

has_conditionRelated

$\top \sqsubseteq\ \leq 1\ \text{has\_conditionRelated Thing}$

$\exists\ \text{has\_conditionRelated Thing} \sqsubseteq \text{Question}$

$\exists\ \text{has\_conditionRelated Thing} \sqsubseteq \text{ContextInformation}$

$\top \sqsubseteq \forall\ \text{has\_conditionRelated Condition}$

has_numberContextInformation

$\exists\ \text{has\_numberContextInformation Thing} \sqsubseteq \text{ContextProvider}$

$\top \sqsubseteq \forall\ \text{has\_numberContextInformation ContextInformation\_NumberForm}$

has_possibleValue

$\exists$ has_possibleValue Thing $\sqsubseteq$ ContextInformation_TextForm

has_precision

$\top \sqsubseteq \leq 1$ has_precision Thing

$\exists$ has_precision Thing $\sqsubseteq$ ContextInformation

$\top \sqsubseteq \forall$ has_precision NumberValue

has_textContextInformation

$\exists$ has_textContextInformation Thing $\sqsubseteq$ ContextProvider

$\top \sqsubseteq \forall$ has_textContextInformation ContextInformation_TextForm

Data properties

has_apk_description_name

$\top \sqsubseteq \leq 1$ has_apk_description_name

$\exists$ has_apk_description_name Datatype: `http://www.w3.org/2000/01/rdf-schema#Literal`$\sqsubseteq$ ContextProvider

$\top \sqsubseteq \forall$ has_apk_description_name Datatype: `http://www.w3.org/2001/XMLSchema#string`

has_apk_name

$\top \sqsubseteq \leq 1$ has_apk_name

$\exists$ has_apk_name Datatype: `http://www.w3.org/2000/01/rdf-schema#Literal`$\sqsubseteq$ ContextProvider

$\top \sqsubseteq \forall$ has_apk_name Datatype: `http://www.w3.org/2001/XMLSchema#string`

has_context_information_interest

$\top \sqsubseteq \leq 1$ has_context_information_interest

$\exists$ has_context_information_interest Datatype: `http://www.w3.org/2000/01/rdf-schema#Literal` $\sqsubseteq$ ContextProvider

$\top \sqsubseteq \forall$ has_context_information_interest Datatype: `http://www.w3.org/2001/XMLSchema#string`

has_information_class

$\exists$ has_information_class Datatype: `http://www.w3.org/2000/01/rdf-schema#Literal` $\sqsubseteq$ ContextProvider

$\top \sqsubseteq \forall$ has_information_class Datatype: `http://www.w3.org/2001/XMLSchema#string`

has_number

$\exists$ has_number Datatype: `http://www.w3.org/2000/01/rdf-schema#Literal` $\sqsubseteq$ SerieValue

$\exists$ has_number Datatype: `http://www.w3.org/2000/01/rdf-schema#Literal` $\sqsubseteq$ NumberValue

$\top \sqsubseteq \forall$ has_number Datatype: `http://www.w3.org/2001/XMLSchema#float`

has_stringRepresentation

$\top \sqsubseteq \leq 1$ has_stringRepresentation

$\exists$ has_stringRepresentation Datatype: `http://www.w3.org/2000/01/rdf-schema#Literal` $\sqsubseteq$ ContextInformation

$\top \sqsubseteq \forall$ has_stringRepresentation Datatype: `http://www.w3.org/2001/XMLSchema#string`

has_text

$\exists$ has_text Datatype: `http://www.w3.org/2000/01/rdf-schema#Literal` $\sqsubseteq$ TextValue

$\top \sqsubseteq \forall$ has_text Datatype: `http://www.w3.org/2001/XMLSchema#string`

has_unity

$\exists$ has_unity Datatype: `http://www.w3.org/2000/01/rdf-schema#Literal` $\sqsubseteq$ SerieValue

$\exists$ has_unity Datatype: `http://www.w3.org/2000/01/rdf-schema#Literal` $\sqsubseteq$ NumberValue

$\top \sqsubseteq \forall$ has_unity Datatype: `http://www.w3.org/2001/XMLSchema#string`

has_uri_provider

$\top \sqsubseteq \leq 1$ has_uri_provider

$\exists$ has_uri_provider Datatype: `http://www.w3.org/2000/01/rdf-schema#Literal` $\sqsubseteq$ ContextProvider

$\top \sqsubseteq \forall$ has_uri_provider Datatype: `http://www.w3.org/2001/XMLSchema#string`

Individuals

Activity

Activity : InternalCondition
has_aquisitionAutomaticForm(Activity, Zephyr_ContextProvider)

ActivityValuePrecision

ActivityValuePrecision : NumberValue
has_number (ActivityValuePrecision "92.0"{=`http://www.w3.org/2001/XMLSchema#float`)
has_unity (ActivityValuePrecision "%"{=`http://www.w3.org/2001/XMLSchema#string`)

Activity_contextInformation

Activity_contextInformation : ContextInformation_TextForm
has_possibleValue(Activity_contextInformation, WalkingHard_textContextInformation)
has_possibleValue(Activity_contextInformation, Walking_textContextInformation)
has_possibleValue(Activity_contextInformation, Running_textContextInformation)
has_conditionRelated(Activity_contextInformation, Activity)
has_possibleValue(Activity_contextInformation, EndOfRun_textContextInformation)
has_possibleValue(Activity_contextInformation, NotMoving_textContextInformation)
has_possibleValue(Activity_contextInformation, RunningHard_textContextInformation)
has_possibleValue(Activity_contextInformation, Flat_textContextInformation)
has_precision(Activity_contextInformation, ActivityValuePrecision)

Altitude

Altitude : ExternalCondition
has_aquisitionAutomaticForm(Altitude, GPS_ContextProvider)

AltitudeValuePrecision

AltitudeValuePrecision : NumberValue

has_unity (AltitudeValuePrecision "metros"{=http://www.w3.org/2001/XMLSchema#string)

has_number (AltitudeValuePrecision "100.0"{=http://www.w3.org/2001/XMLSchema#float)

Altitude_contextInformation

Altitude_contextInformation : ContextInformation_NumberForm

has_conditionRelated(Altitude_contextInformation, Altitude)

has_precision(Altitude_contextInformation, AltitudeValuePrecision)

BodyTemperature

BodyTemperature : InternalCondition

has_aquisitionAutomaticForm(BodyTemperature, Zephyr_ContextProvider)

BodyTemperatureValuePrecision

BodyTemperatureValuePrecision : NumberValue

has_number (BodyTemperatureValuePrecision "1.0"{=http://www.w3.org/2001/XMLSchema#float)

has_unity (BodyTemperatureValuePrecision "graus Celcius"{=http://www.w3.org/2001/XMLSchema#string)

BreathRateValuePrecision

BreathRateValuePrecision : NumberValue

has_number (BreathRateValuePrecision "5.0"{=http://www.w3.org/2001/XMLSchema#float)

has_unity (BreathRateValuePrecision "por minuto"{=http://www.w3.org/2001/XMLSchema#string)

BreathRate_contextInformation

BreathRate_contextInformation : ContextInformation_NumberForm

has_conditionRelated(BreathRate_contextInformation, Breath_Rate)

has_precision(BreathRate_contextInformation, BreathRateValuePrecision)

Breath_Rate

Breath_Rate : InternalCondition
has_aquisitionAutomaticForm(Breath_Rate, Zephyr_ContextProvider)

ChestPainQuestion

ChestPainQuestion : Question
has_conditionRelated(ChestPainQuestion, Chest_Pain)

Chest_Pain

Chest_Pain : InternalCondition
has_aquisitionQuestionForm(Chest_Pain, ChestPainQuestion)

CountryCurrent

CountryCurrent : ExternalCondition
has_aquisitionAutomaticForm(CountryCurrent, GPS_ContextProvider)

CurrentCity

CurrentCity : ExternalCondition
has_aquisitionAutomaticForm(CurrentCity, GPS_ContextProvider)

DeltaFC_ValuePrecision

DeltaFC_ValuePrecision : SerieValue
has_unity (DeltaFC_ValuePrecision "milisegundos (ms)"{=`http://www.w3.org/2001/XMLSchema#string`)
has_number (DeltaFC_ValuePrecision "50.0"{=`http://www.w3.org/2001/XMLSchema#float`)

DeltaFC_contextInformation

DeltaFC_contextInformation : ContextInformation_NumberForm
has_conditionRelated(DeltaFC_contextInformation, Delta_FC)
has_precision(DeltaFC_contextInformation, DeltaFC_ValuePrecision)

Delta_FC

Delta_FC : InternalCondition
has_aquisitionAutomaticForm(Delta_FC, Zephyr_ContextProvider)

ECG_Signal

ECG_Signal : InternalCondition
has_aquisitionAutomaticForm(ECG_Signal, Zephyr_ContextProvider)

EndOfRun_textContextInformation

EndOfRun_textContextInformation : TextValue
has_text (EndOfRun_textContextInformation "Fim atividade fisica"{=`http://www.w3.org/2001/XMLSchema#string`)

FC

FC : InternalCondition
has_aquisitionAutomaticForm(FC, Zephyr_ContextProvider)

FC_ValuePrecision

FC_ValuePrecision : SerieValue
has_number (FC_ValuePrecision "50.0"{=`http://www.w3.org/2001/XMLSchema#float`)
has_unity (FC_ValuePrecision "milisegundos (ms)"{=`http://www.w3.org/2001/XMLSchema#string`)

FC_contextInformation

FC_contextInformation : ContextInformation_NumberForm
has_precision(FC_contextInformation, FC_ValuePrecision)
has_conditionRelated(FC_contextInformation, FC)

Faint

Faint : InternalCondition
has_aquisitionQuestionForm(Faint, FaintQuestion)

FaintQuestion

FaintQuestion : Question
has_conditionRelated(FaintQuestion, Faint)

Fever_contextInformation

Fever_contextInformation : ContextInformation_NumberForm

has_precision(Fever_contextInformation, BodyTemperatureValuePrecision)

has_conditionRelated(Fever_contextInformation, BodyTemperature)

Flat_textContextInformation

Flat_textContextInformation : TextValue

has_text (Flat_textContextInformation "Deitado"{=`http://www.w3.org/2001/XMLSchema#string`)

GPS_CityValuePrecision

GPS_CityValuePrecision : NumberValue

has_unity (GPS_CityValuePrecision "%"{=`http://www.w3.org/2001/XMLSchema#string`)

has_number (GPS_CityValuePrecision "100.0"{=`http://www.w3.org/2001/XMLSchema#float`)

GPS_ContextProvider

GPS_ContextProvider : ContextProvider

has_textContextInformation(GPS_ContextProvider, PatientCurrentCountry_contextInformation)

has_textContextInformation(GPS_ContextProvider, PatientCurrentCity_contextInformation)

has_numberContextInformation(GPS_ContextProvider, Altitude_contextInformation)

has_context_information_interest (GPS_ContextProvider "this.location.latitude"{=`http://www.w3.org/2001/XMLSchema#string`)

has_apk_description_name (GPS_ContextProvider "ContextProviders.apk.desc"{=`http://www.w3.org/2001/XMLSchema#string`)

has_apk_name (GPS_ContextProvider "ContextProviders.apk"{=`http://www.w3.org/2001/XMLSchema#string`)

has_uri_provider (GPS_ContextProvider "https://dl.dropbox.com/u/6681275/ContextProviders
`//www.w3.org/2001/XMLSchema#string`)

has_information_class (GPS_ContextProvider "location"{=`http://www.w3.org/2001/XMLSchema#string`)

GPS_CountryValuePrecision

GPS_CountryValuePrecision : NumberValue

has_unity (GPS_CountryValuePrecision "%"{=http://www.w3.org/2001/XMLSchema#string)

has_number (GPS_CountryValuePrecision "100.0"{=http://www.w3.org/2001/XMLSchema#float)

Headache

Headache : InternalCondition

has_aquisitionQuestionForm(Headache, HeadacheQuestion)

HeadacheQuestion

HeadacheQuestion : Question

has_conditionRelated(HeadacheQuestion, Headache)

HeartRateValuePrecision

HeartRateValuePrecision : NumberValue

has_number (HeartRateValuePrecision "5.0"{=http://www.w3.org/2001/XMLSchema#float)

has_unity (HeartRateValuePrecision "batimentos por minuto (bpm)"{=http://www.w3.org/2001/XMLSchema#string)

HeartRate_contextInformation

HeartRate_contextInformation : ContextInformation_NumberForm

has_conditionRelated(HeartRate_contextInformation, Heart_Rate)

has_precision(HeartRate_contextInformation, HeartRateValuePrecision)

Heart_Rate

Heart_Rate : InternalCondition

has_aquisitionAutomaticForm(Heart_Rate, Zephyr_ContextProvider)

LackOfAirQuestion

LackOfAirQuestion : Question

has_conditionRelated(LackOfAirQuestion, Lack_of_Air)

Lack_of_Air

Lack_of_Air : InternalCondition
has_aquisitionQuestionForm(Lack_of_Air, LackOfAirQuestion)

NotMoving_textContextInformation

NotMoving_textContextInformation : TextValue
has_text (NotMoving_textContextInformation "Parado em pe"{=`http://www.w3.org/2001/XMLSchema#string`)

Palpitation

Palpitation : InternalCondition
has_aquisitionQuestionForm(Palpitation, PalpitationQuestion)

PalpitationQuestion

PalpitationQuestion : Question
has_conditionRelated(PalpitationQuestion, Palpitation)

PatientCurrentCity_contextInformation

PatientCurrentCity_contextInformation : ContextInformation_TextForm
has_precision(PatientCurrentCity_contextInformation, GPS_CityValuePrecision)
has_conditionRelated(PatientCurrentCity_contextInformation, CurrentCity)

PatientCurrentCountry_contextInformation

PatientCurrentCountry_contextInformation : ContextInformation_TextForm
has_conditionRelated(PatientCurrentCountry_contextInformation, CountryCurrent)
has_precision(PatientCurrentCountry_contextInformation, GPS_CountryValuePrecision)

RR

RR : InternalCondition
has_aquisitionAutomaticForm(RR, Zephyr_ContextProvider)

RR_ValuePrecision

RR_ValuePrecision : SerieValue
has_number (RR_ValuePrecision ”50.0”{=`http://www.w3.org/2001/XMLSchema#float`)
has_unity (RR_ValuePrecision ”milisegundos (ms)”{=`http://www.w3.org/2001/XMLSchema#string`)

RR_contextInformation

RR_contextInformation : ContextInformation_NumberForm
has_precision(RR_contextInformation, RR_ValuePrecision)
has_conditionRelated(RR_contextInformation, RR)

RunningHard_textContextInformation

RunningHard_textContextInformation : TextValue
has_text (RunningHard_textContextInformation ”Corrida”{=`http://www.w3.org/2001/XMLSchema#string`)

Running_textContextInformation

Running_textContextInformation : TextValue
has_text (Running_textContextInformation ”Correndo”{=`http://www.w3.org/2001/XMLSchema#string`)

ST

ST : InternalCondition
has_aquisitionAutomaticForm(ST, Zephyr_ContextProvider)

ST_ValuePrecision

ST_ValuePrecision : SerieValue
has_unity (ST_ValuePrecision ”milisegundos (ms)”{=`http://www.w3.org/2001/XMLSchema#string`)
has_number (ST_ValuePrecision ”50.0”{=`http://www.w3.org/2001/XMLSchema#float`)

ST_contextInformation

ST_contextInformation : ContextInformation_NumberForm
has_conditionRelated(ST_contextInformation, ST)
has_precision(ST_contextInformation, ST_ValuePrecision)

UnusualTirednessQuestion

UnusualTirednessQuestion : Question
has_conditionRelated(UnusualTirednessQuestion, Unusual_Tiredness)

Unusual_Tiredness

Unusual_Tiredness : InternalCondition
has_aquisitionQuestionForm(Unusual_Tiredness, UnusualTirednessQuestion)

VomitQuestion

VomitQuestion : Question
has_conditionRelated(VomitQuestion, Vomiting)

Vomiting

Vomiting : InternalCondition
has_aquisitionQuestionForm(Vomiting, VomitQuestion)

WalkingHard_textContextInformation

WalkingHard_textContextInformation : TextValue
has_text (WalkingHard_textContextInformation "Caminhando"{=`http://www.w3.org/2001/XMLSchema#string`)

Walking_textContextInformation

Walking_textContextInformation : TextValue
has_text (Walking_textContextInformation "Andando"{=`http://www.w3.org/2001/XMLSchema#string`)

Zephyr_ContextProvider

Zephyr_ContextProvider : ContextProvider
has_textContextInformation(Zephyr_ContextProvider, Activity_contextInformation)
has_numberContextInformation(Zephyr_ContextProvider, RR_contextInformation)
has_numberContextInformation(Zephyr_ContextProvider, HeartRate_contextInformation)
has_numberContextInformation(Zephyr_ContextProvider, DeltaFC_contextInformation)
has_numberContextInformation(Zephyr_ContextProvider, FC_contextInformation)
has_numberContextInformation(Zephyr_ContextProvider, BreathRate_contextInformation)
has_numberContextInformation(Zephyr_ContextProvider, ST_contextInformation)

has_numberContextInformation(Zephyr_ContextProvider, Fever_contextInformation)

has_context_information_interest (Zephyr_ContextProvider "this.Zephyr.RR"{=`http://www.w3.org/2001/XMLSchema#string`)

has_apk_description_name (Zephyr_ContextProvider "ContextProviders.apk.desc"{=`http://www.w3.org/2001/XMLSchema#string`)

has_information_class (Zephyr_ContextProvider "Zephyr"{=`http://www.w3.org/2001/XMLSchema#string`)

has_apk_name (Zephyr_ContextProvider "ContextProviders.apk"{=`http://www.w3.org/2001/XMLSchema#string`)

has_uri_provider (Zephyr_ContextProvider "https://dl.dropbox.com/u/6681275/ContextProvid
`//www.w3.org/2001/XMLSchema#string`)

Datatypes

float

string

14
Appendix - Patient Ontology

Classes

Characteristic

Disease

Doctor

$\text{Doctor} \sqsubseteq \neg\ \text{Patient}$

EmergencyContact

$\text{EmergencyContact} \sqsubseteq \neg\ \text{Patient}$

Patient

$\text{Patient} \sqsubseteq \exists\ \text{has\_doctor Doctor}$
$\text{Patient} \sqsubseteq \exists\ \text{has\_disease Disease}$
$\text{Patient} \sqsubseteq \neg\ \text{EmergencyContact}$
$\text{Patient} \sqsubseteq \neg\ \text{Doctor}$

Person

Object properties

has_disease

$\exists\ \text{has\_disease Thing} \sqsubseteq \text{Patient}$
$\top \sqsubseteq \forall\ \text{has\_disease Disease}$

has_diseaseCharacteristic

$\exists$ has_diseaseCharacteristic Thing $\sqsubseteq$ Patient
$\top \sqsubseteq \forall$ has_diseaseCharacteristic Characteristic

has_doctor

$\exists$ has_doctor Thing $\sqsubseteq$ Patient
$\top \sqsubseteq \forall$ has_doctor Doctor

has_emergencyContact

$\exists$ has_emergencyContact Thing $\sqsubseteq$ Patient
$\top \sqsubseteq \forall$ has_emergencyContact EmergencyContact

Data properties

Individuals

Altitude_Sickness

Altitude_Sickness : Disease

Atrial_fibrillation

Atrial_fibrillation : Disease

MarioDoctor

MarioDoctor : Doctor

Paroxysmal_Patient

Paroxysmal_Patient : Characteristic

Permanent_Patient

Permanent_Patient : Characteristic

RodrigoEmergencyContact

RodrigoEmergencyContact : EmergencyContact

UncoagulatedBlood

UncoagulatedBlood : Characteristic

VitorPatient

VitorPatient : Patient
has_disease(VitorPatient, Atrial_fibrillation)
has_doctor(VitorPatient, MarioDoctor)
has_diseaseCharacteristic(VitorPatient, Paroxysmal_Patient)
has_disease(VitorPatient, Altitude_Sickness)
has_diseaseCharacteristic(VitorPatient, UncoagulatedBlood)
has_emergencyContact(VitorPatient, RodrigoEmergencyContact)

Datatypes

string

15
Appendix - Person Ontology

Classes

City

$\text{City} \sqsubseteq \neg\ \text{Person}$

Country

$\text{Country} \sqsubseteq \neg\ \text{Person}$

Person

$\text{Person} \sqsubseteq \exists\ \text{has\_phoneNumber Datatype:}$ `http://www.w3.org/2001/XMLSchema#string`

$\text{Person} \sqsubseteq =\ \text{term\_name}$

$\text{Person} \sqsubseteq \neg\ \text{City}$

$\text{Person} \sqsubseteq \neg\ \text{Country}$

Object properties

has_cityOfLiving

$\top \sqsubseteq\ \leq 1\ \text{has\_cityOfLiving Thing}$

$\exists\ \text{has\_cityOfLiving Thing} \sqsubseteq \text{Person}$

$\top \sqsubseteq \forall\ \text{has\_cityOfLiving City}$

has_countryOfLiving

$\top \sqsubseteq\ \leq 1\ \text{has\_countryOfLiving Thing}$

$\exists\ \text{has\_countryOfLiving Thing} \sqsubseteq \text{Person}$

$\top \sqsubseteq \forall\ \text{has\_countryOfLiving Country}$

has_patientLiving

$\top \sqsubseteq \leq 1$ has_patientLiving Thing

$\exists$ has_patientLiving Thing $\sqsubseteq$ Country

$\exists$ has_patientLiving Thing $\sqsubseteq$ City

$\top \sqsubseteq \forall$ has_patientLiving Person

Data properties

has_cityName

$\top \sqsubseteq \leq 1$ has_cityName

$\exists$ has_cityName Datatype: `http://www.w3.org/2000/01/rdf-schema#Literal`$\sqsubseteq$ City

$\top \sqsubseteq \forall$ has_cityName Datatype: `http://www.w3.org/2001/XMLSchema#string`

has_countryName

$\top \sqsubseteq \leq 1$ has_countryName

$\exists$ has_countryName Datatype: `http://www.w3.org/2000/01/rdf-schema#Literal`$\sqsubseteq$ Country

$\top \sqsubseteq \forall$ has_countryName Datatype: `http://www.w3.org/2001/XMLSchema#string`

has_dateTime

$\top \sqsubseteq \leq 1$ has_dateTime

$\top \sqsubseteq \forall$ has_dateTime Datatype: `http://www.w3.org/2001/XMLSchema#string`

has_number

$\top \sqsubseteq \leq 1$ has_number

$\top \sqsubseteq \forall$ has_number Datatype: `http://www.w3.org/2001/XMLSchema#float`

has_phoneNumber

$\exists$ has_phoneNumber Datatype: `http://www.w3.org/2000/01/rdf-schema#Literal`$\sqsubseteq$ Person

$\top \sqsubseteq \forall$ has_phoneNumber Datatype: `http://www.w3.org/2001/XMLSchema#string`

has_text

$\top \sqsubseteq \leq 1$ has_text

$\top \sqsubseteq \forall$ has_text Datatype: `http://www.w3.org/2001/XMLSchema#string`

has_timeMeasuredTime

$\top \sqsubseteq \leq 1$ has_timeMeasuredTime

$\top \sqsubseteq \forall$ has_timeMeasuredTime Datatype: `http://www.w3.org/2001/XMLSchema#dateTime`

has_unity

$\top \sqsubseteq \leq 1$ has_unity

$\top \sqsubseteq \forall$ has_unity Datatype: `http://www.w3.org/2001/XMLSchema#string`

term_name

$\top \sqsubseteq \forall$ term_name Datatype: `http://www.w3.org/2001/XMLSchema#string`

Individuals

Mario

Mario : Person

has_phoneNumber (Mario ”82735218”{=`http://www.w3.org/2001/XMLSchema#string`)

term_name (Mario ”Mario Coelho”{=`http://www.w3.org/2001/XMLSchema#string`)

Rodrigo

Rodrigo : Person

has_phoneNumber (Rodrigo ”82735218”{=`http://www.w3.org/2001/XMLSchema#string`)

term_name (Rodrigo ”Rodrigo”{=`http://www.w3.org/2001/XMLSchema#string`)

Vitor

Vitor : Person

has_cityOfLiving(Vitor, VitorCity)

has_countryOfLiving(Vitor, VitorCountry)

term_name (Vitor "Vitor Pinheiro"{=`http://www.w3.org/2001/XMLSchema#string`)

has_phoneNumber (Vitor "82735218"{=`http://www.w3.org/2001/XMLSchema#string`)

VitorCity

VitorCity : City

has_patientLiving(VitorCity, Vitor)

has_cityName (VitorCity "Rio de Janeiro"{=`http://www.w3.org/2001/XMLSchema#string`)

VitorCountry

VitorCountry : Country

has_patientLiving(VitorCountry, Vitor)

has_countryName (VitorCountry "Brasil"{=`http://www.w3.org/2001/XMLSchema#string`)

Datatypes

dateTime

float

string

16
Appendix - Questionnaire Ontology

Classes

Answer

$\text{Answer} \sqsubseteq = \text{has\_answerText}$

$\text{Answer} \sqsubseteq \neg\ \text{Question}$

$\text{Answer} \sqsubseteq \neg\ \text{Questionnaire}$

$\text{Answer} \sqsubseteq \neg\ \text{Question}$

$\text{Questionnaire} \sqsubseteq \neg\ \text{Question}$

$\text{Answer} \sqsubseteq \neg\ \text{Questionnaire}$

$\text{Question} \sqsubseteq \neg\ \text{Questionnaire}$

Question

$\text{Question} \sqsubseteq = \text{is\_checkBox} \sqcup = \text{is\_comboBox} \sqcup = \text{is\_radioBox} \sqcup = \text{is\_textBox}$

$\text{Question} \sqsubseteq \neg\ \text{Answer}$

$\text{Questionnaire} \sqsubseteq \neg\ \text{Answer}$

$\text{Question} \sqsubseteq \neg\ \text{Answer}$

$\text{Question} \sqsubseteq \neg\ \text{Questionnaire}$

$\text{Answer} \sqsubseteq \neg\ \text{Questionnaire}$

$\text{Question} \sqsubseteq \neg\ \text{Questionnaire}$

Questionnaire

$\text{Question} \sqsubseteq \neg\ \text{Answer}$

$\text{Questionnaire} \sqsubseteq \neg\ \text{Answer}$

$\text{Answer} \sqsubseteq \neg\ \text{Question}$

$\text{Questionnaire} \sqsubseteq \neg\ \text{Question}$

$\text{Questionnaire} \sqsubseteq \neg\ \text{Answer}$

Questionnaire $\sqsubseteq \neg$ Question

Object properties

has_patientAnswer

¡http://patientbuddy/questionnaire.owl#has_patientAnswer¿ $\equiv$ ¡http://patientbuddy/questionnaire.owl#is_answer_of¿$^-$

$\exists$ has_patientAnswer Thing $\sqsubseteq$ Question

$\top \sqsubseteq \forall$ has_patientAnswer Answer

has_possibleAnswer

$\exists$ has_possibleAnswer Thing $\sqsubseteq$ Question

$\top \sqsubseteq \forall$ has_possibleAnswer Answer

has_question

¡http://patientbuddy/questionnaire.owl#has_questionnaire¿ $\equiv$ ¡http://patientbuddy/questionnaire.owl#has_question¿$^-$

$\exists$ has_question Thing $\sqsubseteq$ Questionnaire

$\top \sqsubseteq \forall$ has_question Question

has_questionnaire

¡http://patientbuddy/questionnaire.owl#has_questionnaire¿ $\equiv$ ¡http://patientbuddy/questionnaire.owl#has_question¿$^-$

$\exists$ has_questionnaire Thing $\sqsubseteq$ Question

$\top \sqsubseteq \forall$ has_questionnaire Questionnaire

is_answer_of

¡http://patientbuddy/questionnaire.owl#has_patientAnswer¿ $\equiv$ ¡http://patientbuddy/questionnaire.owl#is_answer_of¿$^-$

$\exists$ is_answer_of Thing $\sqsubseteq$ Answer

$\top \sqsubseteq \forall$ is_answer_of Question

Data properties

date

$\top \sqsubseteq \leq 1$ date

$\exists$ date Datatype: `http://www.w3.org/2000/01/rdf-schema#Literal`$\sqsubseteq$ Questionnaire

$\top \sqsubseteq \forall$ date Datatype: `http://www.w3.org/2001/XMLSchema#dateTime`

has_answerText

$\top \sqsubseteq \leq 1$ has_answerText

$\exists$ has_answerText Datatype: `http://www.w3.org/2000/01/rdf-schema#Literal`$\sqsubseteq$ Answer

$\top \sqsubseteq \forall$ has_answerText Datatype: `http://www.w3.org/2001/XMLSchema#string`

has_questionText

$\top \sqsubseteq \leq 1$ has_questionText

$\exists$ has_questionText Datatype: `http://www.w3.org/2000/01/rdf-schema#Literal`$\sqsubseteq$ Question

$\top \sqsubseteq \forall$ has_questionText Datatype: `http://www.w3.org/2001/XMLSchema#string`

is_checkBox

$\top \sqsubseteq \leq 1$ is_checkBox

$\exists$ is_checkBox Datatype: `http://www.w3.org/2000/01/rdf-schema#Literal`$\sqsubseteq$ Question

$\top \sqsubseteq \forall$ is_checkBox Datatype: `http://www.w3.org/2001/XMLSchema#boolean`

is_comboBox

$\top \sqsubseteq \leq 1$ is_comboBox

$\exists$ is_comboBox Datatype: `http://www.w3.org/2000/01/rdf-schema#Literal`$\sqsubseteq$ Question

$\top \sqsubseteq \forall$ is_comboBox Datatype: `http://www.w3.org/2001/XMLSchema#boolean`

is_radioBox

$\top \sqsubseteq \leq 1$ is_radioBox

$\exists$ is_radioBox Datatype: `http://www.w3.org/2000/01/rdf-schema#Literal`$\sqsubseteq$ Question

$\top \sqsubseteq \forall$ is_radioBox Datatype: `http://www.w3.org/2001/XMLSchema#boolean`

is_required

$\top \sqsubseteq \leq 1$ is_required

$\exists$ is_required Datatype: `http://www.w3.org/2000/01/rdf-schema#Literal`$\sqsubseteq$ Question

$\top \sqsubseteq \forall$ is_required Datatype: `http://www.w3.org/2001/XMLSchema#boolean`

is_textBox

$\top \sqsubseteq \leq 1$ is_textBox

$\exists$ is_textBox Datatype: `http://www.w3.org/2000/01/rdf-schema#Literal`$\sqsubseteq$ Question

$\top \sqsubseteq \forall$ is_textBox Datatype: `http://www.w3.org/2001/XMLSchema#boolean`

Individuals

ChestPainQuestion

ChestPainQuestion : Question

has_possibleAnswer(ChestPainQuestion, Intense)

has_possibleAnswer(ChestPainQuestion, None)

has_possibleAnswer(ChestPainQuestion, Very_Intense)

has_possibleAnswer(ChestPainQuestion, Normal)

has_possibleAnswer(ChestPainQuestion, Little)

is_radioBox (ChestPainQuestion "true"{=`http://www.w3.org/2001/XMLSchema#boolean`)

is_required (ChestPainQuestion "false"{=`http://www.w3.org/2001/XMLSchema#boolean`)

has_questionText (ChestPainQuestion "Sentindo dor no peito?"{=`http://www.w3.org/2001/XMLSchema#string`)

FaintQuestion

FaintQuestion : Question

has_possibleAnswer(FaintQuestion, Yes)

has_possibleAnswer(FaintQuestion, No)

has_questionText (FaintQuestion "Teve desmaio?"{=`http://www.w3.org/2001/XMLSchema#string`)
is_radioBox (FaintQuestion "true"{=`http://www.w3.org/2001/XMLSchema#boolean`)
is_required (FaintQuestion "false"{=`http://www.w3.org/2001/XMLSchema#boolean`)

Frequently

Frequently : Answer
has_answerText (Frequently "Sim, frequentemente"{=`http://www.w3.org/2001/XMLSchema#string`)

HeadacheQuestion

HeadacheQuestion : Question
has_possibleAnswer(HeadacheQuestion, Yes)
has_possibleAnswer(HeadacheQuestion, No)
is_radioBox (HeadacheQuestion "true"{=`http://www.w3.org/2001/XMLSchema#boolean`)
has_questionText (HeadacheQuestion "Sentindo dor de cabeça?"{=`http://www.w3.org/2001/XMLSchema#string`)
is_required (HeadacheQuestion "false"{=`http://www.w3.org/2001/XMLSchema#boolean`)

Intense

Intense : Answer
has_answerText (Intense "Intenso"{=`http://www.w3.org/2001/XMLSchema#string`)

LackOfAirQuestion

LackOfAirQuestion : Question
has_possibleAnswer(LackOfAirQuestion, None)
has_possibleAnswer(LackOfAirQuestion, Intense)
has_possibleAnswer(LackOfAirQuestion, Very_Intense)
has_possibleAnswer(LackOfAirQuestion, Little)
has_possibleAnswer(LackOfAirQuestion, Normal)
is_radioBox (LackOfAirQuestion "true"{=`http://www.w3.org/2001/XMLSchema#boolean`)

is_required (LackOfAirQuestion "false"{=http://www.w3.org/2001/XMLSchema#boolean)

has_questionText (LackOfAirQuestion "Sentindo falta de ar?"{=http://www.w3.org/2001/XMLSchema#string)

Little

Little : Answer

has_answerText (Little "Um pouco"{=http://www.w3.org/2001/XMLSchema#string)

No

No : Answer

has_answerText (No "Não"{=http://www.w3.org/2001/XMLSchema#string)

None

None : Answer

has_answerText (None "Nenhuma"{=http://www.w3.org/2001/XMLSchema#string)

Normal

Normal : Answer

has_answerText (Normal "Mediano"{=http://www.w3.org/2001/XMLSchema#string)

PalpitationQuestion

PalpitationQuestion : Question

has_possibleAnswer(PalpitationQuestion, Yes)

has_possibleAnswer(PalpitationQuestion, No)

is_required (PalpitationQuestion "false"{=http://www.w3.org/2001/XMLSchema#boolean)

is_radioBox (PalpitationQuestion "true"{=http://www.w3.org/2001/XMLSchema#boolean)

has_questionText (PalpitationQuestion "Sentiu alguma palpitação na região do coração?"{=http://www.w3.org/2001/XMLSchema#string)

SomeTimes

SomeTimes : Answer

has_answerText (SomeTimes ”Sim, algumas vezes”{=`http://www.w3.org/2001/XMLSchema#string`)

UnusualTirednessQuestion

UnusualTirednessQuestion : Question

has_possibleAnswer(UnusualTirednessQuestion, No)

has_possibleAnswer(UnusualTirednessQuestion, Frequently)

has_possibleAnswer(UnusualTirednessQuestion, SomeTimes)

is_radioBox (UnusualTirednessQuestion ”true”{=`http://www.w3.org/2001/XMLSchema#boolean`)

has_questionText (UnusualTirednessQuestion ”Sentindo algum tipo de cansaço que você não costumava sentir antes?”{=`http://www.w3.org/2001/XMLSchema#string`)

is_required (UnusualTirednessQuestion ”false”{=`http://www.w3.org/2001/XMLSchema#boolean`)

Very_Intense

Very_Intense : Answer

has_answerText (Very_Intense ”Muito Intenso”{=`http://www.w3.org/2001/XMLSchema#string`)

VomitQuestion

VomitQuestion : Question

has_possibleAnswer(VomitQuestion, Yes)

has_possibleAnswer(VomitQuestion, No)

has_questionText (VomitQuestion ”Vomitou alguma vez?”{=`http://www.w3.org/2001/XMLSchema#string`)

is_required (VomitQuestion ”false”{=`http://www.w3.org/2001/XMLSchema#boolean`)

is_radioBox (VomitQuestion ”true”{=`http://www.w3.org/2001/XMLSchema#boolean`)

Yes

Yes : Answer

has_answerText (Yes ”Sim”{=`http://www.w3.org/2001/XMLSchema#string`)

Datatypes

boolean

dateTime

string