Towards formalisation of situation-specific computations in pervasive computing environments

Reza Shojanoori

School of Electronics and Computer Science

TOWARDS FORMALISATION OF SITUATION-SPECIFIC COMPUTATIONS IN PERVASIVE COMPUTING ENVIRONMENTS

R. SHOJANOORI

PhD
2013

TOWARDS FORMALISATION OF SITUATION-SPECIFIC COMPUTATIONS IN PERVASIVE COMPUTING ENVIRONMENTS

ABDOREZA (REZA) SHOJANOORI

A thesis submitted in partial fulfilment of the requirements of the University of Westminster for the degree of Doctor of Philosophy

May 2013

To my wonderful wife, Afsaneh, for her continuous love and support, and to our amazingly understanding and patient boys, Hossein, Hamed and Erfan.

Acknowledgment

I cannot thank enough the Magnificent the Merciful for giving me the passion and strength to complete this thesis, during which I felt more how frail we are in his glorious Kingdom.

I have to thank my friend, colleague and supervisor Dr. Radmila Juric who has given me much pause for thought during this research. Lively, sometimes heated, discussions we had within our small research group is unforgettable. Her friendship, professionalism, and research enthusiasm is exemplary. I cannot possibly forget and not acknowledge the help of all the people from whom I have learned, my teachers, my colleagues and most of all my “students”. I would also like to thank my friend and colleague Colin Everiss for his help and support throughout.

I am grateful to my wife and the boys for their extraordinary support and forbearance. During the course of this research I could not fulfil my responsibilities as a husband and father and for this I can only apologise and hope I will be able to make it up to them.

Abstract

We have categorised the characteristics and the content of pervasive computing environments (PCEs), and demonstrated why a non-dynamic approach to knowledge conceptualisation in PCEs does not fulfil the expectations we may have from them. Consequently, we have proposed a formalised computational model, the FCM, for knowledge representation and reasoning in PCEs which, secures the delivery of situation and domain specific services to their users. The proposed model is a user centric model, materialised as a software engineering solution, which uses the computations generated from the FCM, stores them within software architectural components, which in turn can be deployed using modern software technologies. The model has also been inspired by the Semantic Web (SW) vision and provision of SW technologies. Therefore, the FCM creates a semantically rich situation-specific PCE based on SWRL-enabled OWL ontologies that allows reasoning about the situation in a PCE and delivers situation specific service.

The proposed FCM model has been illustrated through the example of remote patient monitoring in the healthcare domain. Numerous software applications generated from the FCM have been deployed using Integrated Development Environments and OWL-API.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

ACM	Association for Computing Machinery
AI	Artificial Intelligence
ASeCS	Assisted Self Care System
CML	Context Modelling Language
CoBrA	Context Broker Architecture
CQ	Competency Question
DL	Description Logic
FCM	Formal Computational Model
GOnto	Generic Ontology
GUI	Graphical User Interface
HCI	Human Computer Interaction
HTML	Hyper Text Markup Language
IDE	Integrated Development Environment
IRI	Internationalized Resource Identifier
JDBC	Java Database Connectivity
OWL	Web Ontology Language
PCE	Pervasive Computing Environment
RDF	Resource Description Framework
RDFS	RDF Schema
RMI	Remote Method Invocation
SE	Software Engineering
SeCH	SelfCare Home
SeCHOnto	SeCH Ontology
SOCAM	Service oriented context-aware middleware
SQL	Structured Query Language
SW	Semantic Web
SWT	Semantic Web Technology
SWRL	Semantic Web Rule Language
UC	Ubiquitous Computing
UML	Unified Modelling Language
URI	Uniform Resource Identifier
Web	World Wide Web
W3C	World Wide Web Consortium
XML	Extensible Markup Language
XML-S	XML Schema

CHAPTER 1
INTRODUCTION

1.1 Research Domain

Pervasive computing is leading the way in a fast-growing trend of integrating transparently physical heterogeneous computational devices into our private and professional lives. The ubiquity of these devices and advances in developing software solutions across domains, have raised hopes for the creation of true widespread pervasive computing environments (PCE). The research advances in PCE in the last two decades, and the demand growth for computations on mobile, wireless and handheld devices has earned a growing eagerness within the software and hardware industry for a greater involvement in creating PCEs, particularly in the last ten years. The interplay of research and technological achievements along with the perpetual effort by researchers and practitioners working on various PCE projects on one hand, and maturity of distributed systems and mobile computing on the other, have paved the way for the richness of PCEs in the post-PC era. These efforts and triumphs not only have brought about technical devices perceived as 'exotic' in the 1980s and even 1990s, but also contributed towards a gradual realisation of powerful PCEs spread across many domains.

The changing nature of software applications and pervasiveness of computational environment has not and could not have been addressed by traditional computing, basically because the nature of computation has changed. Also the way we perceive computing today has changed. Not only do we compute with our handheld devices at any time, we also have started to become in charge of computations. For example, we recall not long ago, when in a bank in central London a customer who did not have any form of identification with him, was begging the fastidious branch manager to let him withdraw £50 from his account. Now, provided your mobile phone is registered with your bank, you can withdraw cash without a debit card. Your only proof of identification is your mobile phone. Until not long ago, we had a very systematic and ordered way of computing through business applications, which were run somewhere by someone. Leaving us quite isolated from it. Now, we are in charge of the way we compute. Mobile Web-based services, mobile shopping, trading, commodity market probing, following the news, watching your favourite TV program, managing your schedule and appointments with your doctor, monitoring local civic services are just a few examples of what users can do whenever they want and while on the move. Although the Web has changed our lives, not all applications are run on the Web, and even Web applications offer, by and large, limited functionality. However, being in a PCE provides users with a different experience from being exposed to a Web application.

Considering the far reaching impact of PCEs on our everyday life, their diverse characteristics and trans-disciplinary nature, and of course being a relatively new topic, it is understandable that the literature does not show any consensus on the definition of PCEs. Nevertheless, the user role and involvement in a PCE is generally accepted as a key feature of the PCE. This varies in different publications, but by and large, the user is an indispensable part of PCEs who expects some kind of service from it. Still, the realisation of this objective is viewed from different angles by different researchers and practitioners. However, what is commonly accepted is the fact that the complexity of the computing infrastructure had to be hidden from users for minimal intervention by them to administer computation. This has been a reflection of what the visionary Weiser articulated, about the experience users have had in the past and the obligations of the computer scientists to avoid such complexities (Weiser, 1991).

This lack of clarity, and that sometimes researchers and practitioners focus on specific aspect(s) and not PCE as a whole, introduced several synonyms for pervasive computing in the literature. The most commonly interchangeable term used for pervasive computing is ‘ubiquitous computing’. Other terms include ‘proactive computing’, ‘ambient computing’; some such as Streitz and Nixon (2005) and Lyons (2002), consider ‘ambient intelligence’ is also pervasive computing.

1.2 Research Problem

In this section we will explain what the core of the research problem is. However, prior to that for a better appreciation of the problem we would like to elaborate on our motivation, vision, and eventually our concerns for the research.

1.2.1 Research Motivation

Users are now in charge of the way they compute. In this section, we will look into this phenomenon from two perspectives, new role of computing and the inescapability of embracing new technologies.

1.2.1.1 A New Role for Computing

We have to assess new ways of ‘thinking about computers’. In the second decade of the 21st century, our attitude towards computing in general and rapidly changing perception on what we compute and how we compute must reflect the ever changing and increasing demand for computing in general across domains. In the era of daily creation and the availability of new Android or iOS apps which can be programmed by users and posted on various open source platforms (at the time of writing this the number of app downloads exceeded 25 billion), at the time when sensor-equipped smart phones are commonplace, and we are being enabled to manage our electronic ‘possessions’ from wherever we are, we really need to re-think where the computations stand today. The times when we were only concerned to compute in an orderly manner, as in traditional business transaction or general-purpose processing supported by highly structured database repositories, have gone.

We may argue that the Web changed everything and that our software applications today are hosted by the internet technologies and numerous solutions which perceive software as a service. However, we still compute in an old manner: applications are solely dependent on huge data repositories, and our computation is delivered through traditional programming procedures supported by components and service oriented technologies. Pervasiveness of modern computational spaces may use such solutions, but it would be impossible to model user behaviour and the environment to constantly address users' situation-specific expectations. Very often we associate the expectations users may have in PCEs with services that may be delivered to them. Realizing Weiser's view to bring computers to users, requires timely information on the situation the user is in (Banavar et al., 2000)(Chen et al., 2004b)(Saha and Mukherjee, 2003)(Sullivan and Lewis, 2003). Users, their location, their preference and wishes appropriate to the situation, their activity, the present environment, surrounding devices and their up-to-moment state and the like are typical necessary information for any reasoning mechanism to compute the situation, and to reason upon to deliver a service to the user. Database management systems cannot support such requirements.

If we would like to see PCEs as domain and situation-specific environments, then we should assume that for each situation a new situation-specific computational space, which consists of a situation-specific model, data, and computations, will be created. It is also important to address the impact and role of user in the creation of instances of PCEs. User adoption of such instances of PCE is crucial for their existence. Assuming that we can secure the existence of an situation of PCE without user participation is wrong. The user must be willing to co-exist in the situation of the PCE, decide when computation should take place, and when necessary, indicate their preferences at runtime.

1.2.1.2 Adopting New Technologies

We have to examine the impact of the software and hardware changes on our way to creating situation of PCEs. We know how technology has had impact on our lifestyles, businesses, education and governance, and are aware of our dependence on the benefits of adopting technological changes as they appear. It is not the question "when and if" we have to embrace new technologies. Changes of modern technology in today's everyday life is drastic. There are always so many of them around us that we have opportunities to use them immediately, thanks to their modern and easy-to-use interfaces. Taking into account the generation of young people that is growing up with modern gadgets, one can safely argue that technology is a ‘must’ and not ‘if’. We have to envisage how quickly we could respond to new technologies and what this means in terms of creating computational spaces in PCEs. Will these computational spaces be of the same nature as in the environments, which are not pervasive? From that perspective, software technologies and the Semantic Web in particular, have delivered interesting solutions for PCEs. Solutions that may be able to address the needs of a new computational model for supporting situations in PCEs. Increasing popularity of the SW technologies shows that such new computational models which focuse on the manipulation of the semantics of situations in PCEs work very well, particularly if user expectations and involvement in such spaces are taken into account.

We have learned the lessons from the past in terms of standardisation of software technologies before we start exploiting them across various domains. Therefore W3C attempts is a good example of preparing the Semantic Web technology (SWT) stack, which can be used within and outside its original purpose of manipulating the meaning within the Web pages across the Internet (Horrocks et al., 2005). The SWTs have brought us new types of computational space which can manipulate the meaning of information available on the Web and therefore securing its handling. However, we are able to extend the same mechanism of manipulating the semantics of the Web towards any other form of computations not necessarily related to the Internet. In other words, we have to pay attention to the power of new languages, which do exactly that: handle the semantic of computational spaces and allow us to manipulate it outside traditional highly structured repositories of databases and SQL like exploitation and manipulation of its content.

So, we are encountered with pervasiveness of modern computational spaces that cannot be tackled through traditional computations on one hand, and availability of technology on the other. So, these were the triggers to start our thinking. But thinking is just the starting point.

1.2.2 Research Vision

We have to agree that pervasive computing has now penetrated into our everyday life in almost all domains of computing and that certain software applications have been taken for granted when we create software solutions for PCEs. Examples are pervasive healthcare (Thoyib et al., 2011), (Zhang et al., 2011a), ubiquitous learning (Hwang et al., 2011), (Tsai et al., 2001), ubiquitous manufacturing (Zhang et al., 2011b), just to name a few. We have also been aware that software production today is heavily supported by numerous integrated development environments (IDEs), tools, ready-made code available on open sources and forums (Truong and Dustdar, 2010), (Ayala et al., 2011). Most of our modern software applications interpret PCE as 'pervasive spaces' (Helal, 2011), 'context-aware smart spaces' (Chen et al., 2011), some offer solutions for the management of pervasive environments (Bourcier et al., 2011), some are more technological and domain specific such as (Milosevic et al., 2011), and some are more generic (Lukowicz et al., 2012). As a result, we would like to find out what we can perceive as computation in PCEs. We want to find out whether in this modern world of pervasiveness of technology we can make successful computations as in the past. We were supposed to provide structured data through some interfaces and the computing system was supposed to give results at the end. Now, we have to review our perception of 'data' and what 'computing' in PCE is.

Therefore, our vision is that computations in PCEs must :

1) Capture semantics of a situation in a PCE.
2) Address constant changes of situations that may occur in a PCE, which may differ from moment to moment. That is, we may witness many different situations in the PCE.
3) Take into account that understanding situations in a PCE depends on how successfully we can model their semantics , manipulate and reason upon it.
4) Improve earlier research visions on context awareness (Dey, 2001)(Schmidt et al., 1999), its implementations and roles in modern pervasive computing spaces (Ellenberg et al., 2011)(Gellersen et al., 2002)(Sang et al., 2003)(

Adlam and Orpwood 2004). This will help to strengthen our motivations, outlined in paragraph 1.2.1.

1.2.3 Concerns

The feasibility of putting forward our vision on creating computations for PCEs will depend on the ability to answer many questions.

For example, what is the difference and borderline between collecting interpreted contextual-data and defining a situation in a PCE? Do both of them have any or similar impact on the creation of the computationally significant representation of PCE?

If we assume that we would still like to use traditional software engineering (SE) principles of applying computations with imperative and declarative programming, upon highly structured repositories of databases and Web pages, can we use the same principles in defining situations in PCEs and delivery of a situation-specific service(s) to users? Would such SE principles take into account our vision of creating computational spaces for PCEs as itemised 1)- 4) in paragraph 1.2.2?

If we assume that defining a situation in a PCE requires understanding of the semantic of the PCE or maybe the meaning/purpose of each situation in PCE (as in 3) above), then we might need computations with reasoning mechanisms which should be a part of our SE principles?

If we agree that inference and reasoning mechanisms in PCEs would support the definition of situations and delivery of a situation-specific service(s) to users in PCEs, then what would the computations in PCEs be?

Finally, if we agree on the presence and the purpose of inference and reasoning in PCEs, which technology should be used in order to achieve both: the definition of situations and delivery of services in PCEs?

Even if we do not know the immediate answers to our concerns above, we assume that the traditional Artificial Intelligence (AI) techniques and methods to reason and secure inference might not be the best possible option for defining situations and delivering services in PCE. For example, the evolution from the expert systems generation to cognitive systems and then to ‘intelligent systems’ and even ‘intelligent assistant systems’, and now ‘context-based intelligent assistant systems’ generation, is still following traditional interpretation of context and reasoning upon it with minimal regards for users and changes in situations, if at all any. There is no evidence that these systems would work in PCEs with 1)-4) in mind. In a testimony of the last 25 years of AI, Brezillon (2011) acknowledges that a radically different perception of context, situation and its relationships with users is necessary if we wish to address crucial aspects of pervasive computing.

Consequently, we envisage that without SE principles in creating computational models, which define situations in PCEs and reason upon them in order to deliver services to users in PCEs, we cannot claim that PCEs exist. Therefore, the SE community has a key role to play to materialise what the AI community regards as 'intelligent software systems' and what could be delivered through defining constantly changeable situations of PCEs.

1.2.4. The Core of the Research Problem

We can summarise from the discussion above that the core of our research problem is polarised around thefollowing questions.

- What would be the computationally significant representation of a situation in PCE?
- If it is a computational model that defines a situation in a PCE and secures delivery of situation-specific services to users in PCEs, what would it consist of?
- How would we support the semantics in a PCE and what would be our inferring mechanism to support reasoning in PCEs?
- What should we reason about when defining a situation in a PCE?
- Shall we create new reasoning mechanism which do not rely necessarily on reasoning techniques available in AI?
- Should we strive a formal computational model, which answers all our questions and address 1) - 4) in paragraph 1.2.2?
- How would we use the formalised computations in real life examples and what would be needed for their implementations?

Consequently, our research should

- Address the role of programming languages in representing situations in PCE and their (in)ability to represent the semantic in PCEs.
- Show that the “intelligence” or “smartness” of PCEs are not always guaranteed by computational power of various algorithms which are stored in computational models, nor by the management of computations through component and service oriented platforms, while perceiving software as a service and using excessive object-oriented programming.
- Contemplate that we must use modern software technologies which allow conceptualization of knowledge and reasoning in order to address the “intelligence” and “smartness” in PCEs. One of the best options might be the SWTs which have been used for interpreting the meaning/semantics of numerous websites, where understanding of the website contents has been supported by languages and reasoning from the SWT stack.

Therefore, our way forward in proposing a new role for computing, that allows adopting transparently new technologies in PCEs, is a formalised computational model which address 1) – 4) in 1.2.2, using SWT and utilising the SE principles.

1.3 Research Objectives

The aim of this research is to specify a formal computational model that can represent computationally significant semantics of different situations in domain-specific PCEs. The model can accommodate semantics of different domains and is therefore reusable across any domain of interest. In view of this, the research objectives are to:

1) Analyse and summarise the problems in and shortcomings of pervasive computing and assess the way it has been addressed in SE in the last decade.
2) Create a list of common characteristics of PCE and set the foundation for defining situations in PCEs through a formal computational model, which would satisfy SE principles.
3) Define a formal computational model which will allow representation of computationally significant semantics of any situation in PCEs for the purpose of delivering situation-specific services.

4) Illustrate and implement the proposed formal computational model in a domain of interest, using SWRL enabled OWL ontology.

1.4 Research Approach

Following an in depth analysis of PCEs in general and developing a formal computational model, we will perform a systematic proof-of-concept implementation approach to illustrate the functionality of our proposed formal computational model. The model will represent computationally significant semantics of different situations in domain-specific PCEs, and support reasoning upon it to deliver services to the users of PCEs.

The proof-of-concept implementation will not only validate the proposed model for PCEs, but will also demonstrate the potentials of SW technologies in developing computational models in PCEs.

1.5 Research Method

There is no consensus in the software development community on what PCE is. Considering the complexity of the subject matter and its multidisciplinary nature we start our research by

1) Investigating what PCE is, and what constitutes a PCE. We explore what the characteristics of PCEs are, and what the expected delivered output of a PCE is.

We would like to emphasize that due to the lack of widely accepted PCE definition, we provide our own definition and observe it throughout.

Despite being a relatively new topic in the realm of computer science, there has been growing attention to PCE by the research community and equally by practitioners. Therefore, the next aspect of the research method is to

2) study how in the past situations in PCEs have been formally defined (presented), and what are their weaknesses considering 1) above and lack of widely accepted definition of PCEs.

Rather than providing a specific solution to a problem-specific and domain-specific situation, we favour a SE solution to the creation of PCE situations. This entails a model that is generic, i.e. applicable across domains and situations in PCEs, and simple enough to be used for different situations in PCEs. To achieve a generic solution for PCEs considering 1) above we need a formal computational model. Therefore, our next aspect of the research method is to

3) develop a formal computational model that allows definition (representation) of situations in PCEs, and reasoning upon it to deliver services to the user of PCEs.

We would also like to emphasize that our formal computational model should primarily serve software engineers and guarantee that certain computations will take place with the expected outcome. Consequently, no further proof of the formal computational model would be required except introducing its concepts and their relationships which will enable the computations typical of SE principles. Ultimately, the role of the formal model would be to guide software engineers to perform computations in pervasive spaces.

To illustrate the above formal model, it will be applied to an example scenario. Therefore, our eventual aspect of the research method is to

4) apply the formal model as a result of 3) in a typical PCE environment.

1.6 Thesis Outline

Following this introductory chapter, in Chapter 2 background information on ubiquitous computing and pervasive computing is provided. In this chapter we also elaborate on the perception of context awareness of the past and present. Our definition of PCE and what it constitutes, which will be followed throughout the thesis, is stated in this chapter.

In Chapter 3 we go through related works that provide any generic solutions to address any aspect of PCEs in general. Through the chapter, we point out the problems and concerns and contrast our proposal to existing solutions. Expectations, limitations of existing solutions, and common characteristics of PCEs are summarised in separate tables in this chapter. An explanation on what pervasive computing is and how we do it is provided at the end of the chapter to set ground for the following chapter.

In Chapter 4, we define the formal computational model through definitions and axioms. These definitions and axioms will define the creation of domain and situation-specific taxonomical structure. The steps towards creation of this structure is provided in pseudo code and in diagram format. How reasoning upon the taxonomical structure element is done to deliver a service to PCE users is also explained in this chapter.

In Chapter 5, we introduce a particular case study in a health domain environment, and a software architecture supporting deployment of the formal computational model. Formalism delivered in Chapter 4 are mapped to the case study to illustrate how the semantics of the example scenario as a situation are represented by following the formal model. The illustration is done through a Java application that communicates with the ontological model and SWRL reasoning engine via OWL API.

In Chapter 6, we evaluate and reflect on our achievements through the course of this research. We also provide our conclusions of the research including the contributions and advantages of the research outcome, followed by future works in this chapter.

CHAPTER 2
BACKGROUND OF THE RESEARCH

In Chapter 1 we discussed our motivation for the research and a new role for computing in the 21st century. We briefly also stated different experiences users of PCEs have comparing with any other form of computations. Pervasive or ubiquitous computing has increasingly attracted the attention of researchers in the last two decades. Equally, the wide range of applications this promising paradigm has offered, has encouraged practitioners and policy makers too. Proactive involvement of both public and private sector in the on-going efforts of realising seamless integration of heterogeneous computational devices into our everyday personal and professional life, is now assuring. Consequently, we need background information on ubiquitous and pervasive computing with examples. The context awareness that pervasive computing applications entail, along with context modelling, is also discussed in this chapter. How SWTs, particularly RDF, OWL ontology and SWRL (W3C 2004a,b) have become handy in realising pervasive computing applications are elaborated here with inclusion of a comprehensive, although not exhaustive, research examples.

2.1 Ubiquitous Computing

Advances in information systems, technology and communication brought to reality the once science-fiction concept of 'cyberspace' (Gibson, 1984). In less than a decade after William Gibson coined the term cyberspace, the integration of communication tools and devices to allow wide range of interactive communication was no longer fiction. The advances in microchips and computer industry in general, in parallel with continuing progress in telecommunication brought about a new vision of the cyberspace. If three decades ago the ratio of microchips to humans was 1 to 100, by mid 90s this was 1 to 1. With the same growth rate the 1 to 1 was reversed in 2005 to 100 to 1 (McCullough, 2004). Regardless of the statistics, we do feel the saturation of our surroundings with a variety of computational devices as they have been integrated into our lives to a point, where it is difficult to imagine our lives without them.

2.1.1 Expansion of Computing

This development and the emergence of relatively affordable computers in the mid 1970's, which resulted in the ubiquity of personal computers and devices with embedded computational capabilities by the 1980's, inspired researchers to question and attend to some of the deep issues the computing community was facing, with the enthusiastic scientists and researchers both in the academia and industries starting to talk about a new paradigm or era in computing.

The pervasiveness of personal computers meant that people could easily interact with the cyberspace via their computers with a keyboard or a mouse. In other words, the whole cyberspace was accessible by everyone at the same time through their desktop. These huge steps have gone much further than imaginations of visionaries such as Gibson. Computing, not only in personal computers but also as embedded microchips or integrated processors in a vast variety of devices, has now extended the cyberspace perspective to an extraordinary level.

2.1.2 The New Paradigm and HCI Community

The concept of the new paradigm, or the new trend advocating a new way of thinking about computing was not shared by the human computer interaction (HCI) community. The pervasiveness of PCs and the fact that they occupied the 'centre of attention' of their users was not convincing enough for the HCI community to see the new trend, and instead they focused on one side effect of the phenomenon by promoting the creation of more intuitive interfaces to ease the interaction with computers.

Mike Weiser criticised this approach and wrote in his influential article *The computer of the 21st century* (Weiser, 1991) that the deep issue surrounding personal computers 'is not just a user interface problem'. He had a more holistic approach to determining the problem. Questions such as why computers are too complex and hard to use, too demanding of attention, too isolating from people and their activities, and too dominating as they have occupied our desktops and our lives (Weiser, 1991), were well received by the computing community.

This view was further elaborated in (Weiser et al., 1999). The focus of the new vision was on how transparently technology can be integrated into users' daily social activity wherever they happened to be and whenever it is. A new way of thinking about 'computers' that took into account the human world, and that computers are working transparently at the background was promoted. The HCI perception of the problem at hand was inclined towards devices, whereby the more open-minded revisionists focused their attention on users in a way that they can comfortably perform daily activity without having to bother to attend to direct interaction with computers or computing devices. One view was focused on computers as a source of information hence justifying their research on how to interact with computers, whereas the other view led by Weiser regarded computers just as an interface to information and advocated the view that computers should not occupy users attention and it is the information that users need to interact with, not the computers.

Weiser described the new trend in computing founded on earlier trends: mainframe, personal computer, and widespread distributed computing-Internet (Weiser and Brown, 1998) and named it 'ubiquitous computing' (UC). Therefore, in the new paradigm the UC is fundamentally characterised by artefacts, computers or devices embedded with processors that are computationally interconnected. This is why ubiquity of microprocessor-embedded objects and the Internet are considered as forerunners of UC.

2.1.3 AI is Challenged

The emergence of the new trend not only challenged the HCI community's view but the traditional practice of AI was also in question. The UC advocators argue for a vision of 'calm technology' (Weiser and Brown, 1996) whereby the availability of more detailed information to the user puts the 'user at the centre' and brings about calmness.

The ubiquity of computing-enabled devices that are networked, empower the user by expanding his 'periphery' so that he can switch his attention from centre to a non-centre (periphery) whenever necessary (Weiser and Brown, 1998). Pousman and Stasko (2006) state that with such technologies 'users are more able to focus on their primary work tasks while staying aware of non-critical information that affects them'. As an analogy to appreciate the difference between these two views consider the following scenario. You are making breakfast for yourself just before setting off to work. The centre of your attention is making breakfast. Simultaneously, you are listening to the traffic radio station so as not to miss any warning that might affect your daily route to work. The TV set is also switched on so that you will notice when your long awaited program on mortgage advice, which is scheduled to be on the morning program appears on screen. You are exposed to various periphery data without attending to them, but as soon as, for example, you see that your expected TV program is showing, your attention will be diverted to the TV. This availability and not overloading of information gives one a sense of serenity.

A network of microprocessor embedded devices in a UC environment gives exactly that unobtrusive serenity to users. Weiser calls this awareness of periphery without explicit attention 'locatedness'. In their criticism against the AI community, defenders of the new vision believed that if a computer device becomes aware of only where it is located, (locatedness), it can adapt its behaviour significantly "without even a hint of AI" (Weiser, 1991). McDermott (1981) complained also that programs in AI to a great degree are "problems rather than solutions".

2.1.4 Recognition of the New Paradigm

This shift of paradigm towards UC as a result of the new thinking about computing was sensed by ACM (Association for Computing Machinery) when they organised in March 2000 a conference expertly entitled ‘Beyond Cyberspace’. In an editorial note (Crawford, 2000), this vision was stated as *“The era of pointing, clicking, or typing is giving way to new seamless, intuitive links between the two worlds (humans and computers)… It’s clear we’ve reached a turning point in the way we interact with computers.”*

In March 2000 Intel’s persuasion of participants at Intel's Computer Continuum (Intel 2000) to explore future computing possibilities reads

> *“Computing, not computers, will characterize the next era of the computer age. The critical focus in the near future will be on ubiquitous access to pervasive and largely invisible computing resources. A continuum of information processing devices ranging from microscopic embedded devices to giant server farms will be woven together with a communication fabric that integrates all of today's networks with networks of the future. Adaptive software will be self-organizing, self-configuring, robust and renewable. At every level and in every conceivable environment, computing will be fully integrated with our daily lives”.*

Therefore, the penetration of a wide variety of networking devices, wireless networks, personal digital assistants, sensors, actuators, and numerous other mobile embedded devices into the field of computing, together with their increased deployment by users, have all led to the emergence of UC. The new trend started in the nineties but became the focus of the research community’s attention in the following decade. Furthermore, it aims to integrate computing and communication devices in and with the environment transparently so as to enable users to focus their attention on what they actually want to know or to do rather than on their interaction with computing devices per se.

The increasing attention to UC has introduced new terminologies, sometimes as synonyms for UC. Some of these are more commonly used than others in the literature , among which is pervasive computing.

2.2 Pervasive Computing

The new vision about computers and computing in general, along with the unprecedented advancement and expansion of technology of various types of devices with embedded microprocessors, extending from domestic every-day use to sophisticated professional tools in the healthcare domain, manufacturing, defence, public transport systems, public offices and similar, has inspired more and more researchers to divert their attention to this vision. The emergence of serious research and applied projects such as ParcTab(Schilits et al., 1993), Stick-e Note(Brown et al., 1997), Personal Tour Guide (Abowd et al., 1997), Intelligent Room (Coen, 1998), Bat (Harter et al., 1999), Context Toolkit(Salber et al., 1999b), Aura (Garlan et al., 2002), Vivago(Korhonen et al., 2003), One World (Grimm, 2004), CoBrA Intelligent Meeting Room (Chen et al., 2003a), CareMedia(Bharucha, 2006) made it clear to those still in doubt that the new vision is not hype and it is here to stay and prosper.

2.2.1 Launch of Pervasive Computing

At the start of the 21st century, coincidentally two articles were published, one from IBM (2001) and another from Satyanarayanan from Carnegie Mellon University (Satyanarayanan, 2001) both addressing their perception of the required infrastructure of the future networks of computers and computationally enabled devices. The former introduced the term ‘autonomic computing’ and the latter ironically advocated ‘pervasive computing’ that IBM had publicised. The first evidence of the appearance of the term “pervasive computing” in the literature appears to be in 1999 when IBM devoted one issue of its journal, ‘IBM Systems Journal’ to this topic. In an editorial article, Ark and Selker (1999) considered the emergence of the term pervasive computing as a continuation of the work by the HCI community on the interaction of users with pervasive computers.

Throughout this issue of the journal, the terms pervasive and ubiquitous computing were used interchangeably. Both articles addressed equally low and high level aspects of the new computing era, i.e. the technical structure and deployment of technologies along with the design and creation of software architectures supporting environments in which users are empowered by receiving necessary services without disturbances.

IBM has stated in its manifest that the growing complexity of the IT industry has hindered its purpose which is to serve people. They suggested that their proposed architecture of autonomic computing would deal with the complexity of existing computing infrastructure systems rather than relying on human intervention and administration. Both (IBM, 2001) and (Satyanarayanan, 2001) were in favour of embedding the complexity in the software and hardware system infrastructure and automating its administration. To allow users to concentrate on what they want to do and not how they want to do it, exactly what Weiser was trying to convince the HCI community.

When we look at the eight characteristics that IBM proposed for their vision, however, little on dealing with users' expectations of the 'autonomic computing' system is found. From the same perspective Satyanarayanan demonstrates consideration for the user and wrote that a PCE system subsumes all the known features of distributed and mobile computing and therefore it must be scalable while allowing environments with different physical and computational provisions to communicate. He summarised PCE systems to be proactive systems which allow linking knowledge from different parts to infer knowledge and anticipate with minimal user distraction (Satyanarayanan, 2001; 2011).

2.2.2 Review of PCE Perceptions

Viewing the system from users' perspective has attracted a growing number of researchers. Henricksen et al. (2002) emphasises that pervasive computing demands applications that are capable of operating in highly dynamic environments and of placing fewer demands on user attention, hence PCE needs to be sensitive to its context that is 'highly interrelated', 'imperfect, and that exhibits a range of 'temporal characteristics' and that has many 'alternative representations'.

In their expectations of the kind of support PCE should provide, Abowd and Mynatt (2000) describe activities in such environments as 'rarely have a clear beginning or end’, ‘interruption is expected’, ‘multiple activities operate concurrently’ and ‘context-shifting among multiple activities is assumed’. Knowledge on ‘user preferences’, ‘device capabilities’, and ‘application requirements’ are considered by Arbanowski et al. (2004) as the key requirements for PCE systems.

These spaces have characteristics. Bacon (2002) and Intille (2002) have listed a number of features for a pervasive computing environment. Intille declared a PCE as a ‘Non-intrusive’ environment with abilities to ‘empower’ and help people, and learn from the behaviour of its inhabitants. In a separate paper he raises his concern about the level of presence of such systems in people’s lives and warns that PCEs should not “… strip people of their sense of control over their environment’ (Intille, 2006).

A PCE system according to Satyanarayanan (2001, 2011) has to be context-aware to be minimally intrusive. In his paper, how context is internally represented, how frequently the context information has to be consulted and where to store it, whether historical context is useful are also outlined. Saha and Mukherjee (2003) add the need for perceptual information about the environment as another differentiating feature of PCE and traditional computing.

Chen et al. (2004b), who believe that PCE system is a natural extension of mobile computing, describe their vision of PCE as computer systems seamlessly integrated into the life of everyday users, providing them with services and information in an “anywhere, anytime” fashion. Communication between different pervasive computing environments, PCEs, according to (Chen et al. 2004b) requires interoperable devices and sensors with computing capabilities involved in such environments to be able to ‘share knowledge’, and ‘reason about their environment’.

From this perspective Sullivan and Lewis (2003) considered ‘massive scalability’, ‘heterogeneity of processor forms’, ‘poor application portability over embedded processors’, ‘heterogeneity of access networks’ as the most challenging tasks in developing PCEs. Hellenschmidt (2006) outlined the requirements for the assembly of ‘intelligent’ environments from distributed devices and components to achieve pervasive software infrastructures. He summarised these as extensibility, to add devices at runtime, exchangeability of devices, autonomy of devices from each other, decentralization, provision of conflict resolution mechanisms, ease of implementation of software applications using the environment, and real-time responses to support users. Ark and Selker (1999) in their leading article highlight four major aspects of PCE. Computing being spread throughout the environment; users being mobile; increasingly information appliances become available; and communication between the elements of the system including users is made easier. They stress that users should not carry with them devices containing their personal information, and at the same time a device will be more aware of its user and surroundings. Hence they called this type of computing ‘context-based computing’. Cassou et al. (2009), on the other hand, stress on the heterogeneity’, ‘dynamicity’, and ‘lack of structuring’ between interrelated components of PCE as the main features. Saha and Mukherjee (2003) characterise PCE with proactivity, scalability, heterogeneity, integration and invisibility. They consider ‘intelligent environment’ as a prerequisite to PCE systems.

Along the same line, researchers generally view the unprecedented ubiquity of computing capabilities and advances in semantic web technologies as an unprecedented opportunity for designing and deploying systems in PCEs that empower end-users doing their day-to-day activities with greater comfort, simplicity and support (Yoo, 2010; Romero et al., 2011; and Rolim et al., 2011).

2.2.3 Pervasive Computing Versus Ubiquitous Computing

The Oxford Dictionary defines ubiquitous as “present, appearing, or found everywhere” and explains pervasive as “(especially of an unwelcome influence or physical effect) spreading widely throughout an area or a group of people”. The English Collins Dictionary defines ubiquitous as “having or seeming to have the ability to be everywhere at once” and includes it in the list of synonyms for pervasive.

Both in academia (Chen and Kotz, 2000; Satyanarayanan, 2001; Intille, 2002;

Sullivan and Lewis, 2003; Henricksen and Indulska, 2006; Bettini et al., 2010) and industry (Kephart and Chess, 2003; Mühlhäuser and Gurevych, 2008) pervasive and ubiquitous computing are used, by and large, interchangeably. Nevertheless, some including Singh et al. (2006) believe in disparity between pervasive and ubiquitous computing, and stress that when the former is combined with mobile computing, the latter is achieved. To emphasise on the importance of pervasive computing for UC, they perceive realization of 'true' UC through knowledge sharing between individual PCEs. At the same time some authors are more lenient when using UC, for example, to Walker et al. (2001) any ubiquity of computing devices where their performance is transparent is a UC environment.

To summarise, going through research publications, for example those mentioned already, one can arguably state that pervasive computing is now being commonly used in the literature as a 'welcome influence' and as synonym for ubiquitous computing. Therefore, in this thesis we are not differentiating between PCE and UC. We attempt to use PCE throughout this document as it is more commonly used in the literature. Nevertheless, at times the term UC might be used interchangeably without any intention to indicate any difference unless otherwise stated.

We have mentioned in section 1.2.1.1 that the computational spaces in PCEs consist of model, data and computations. The data of a situation in a PCE is referred to in the literature as context. We have said in section 1.2.2 that we would like to witness improvement of earlier visions on context awareness and its implementations and roles in modern pervasive computing spaces in PCEs. We would also like to know what the difference and borderline between collecting interpreted contextual-data and defining a situation in a PCE is (section 1.2.3). This leads us to the next section on context.

2.3 Context in Computing

The term context means 'the circumstances that form the setting for an event' in the Oxford dictionary and 'the situation within which something exists or happens, and that can help explain it' in the Cambridge dictionary. Despite this general understanding of what context is, in the computer science discipline the perception is different. There has been growing research in this area, but to the best of our knowledge there is yet an agreed upon definition of context to emerge. In this section some of the most referenced definitions and views in the literature are presented.

We would like to point out here that in the literature sometimes the term 'situation' is being used for 'context'. However, some publications like (Gessler et al., 2005), or (Coutaz et al., 2005) distinguish these from each other. They regard situation is related to location only, and context is related to conditions such as temperature, weather, or lighting in that location. To Dey and Abowd (1999), these two terms are different when they say context 'characterize the situation'.

2.3.1 Definition and Categories of Context

The definitions of context in different publications are presented here in chronological order. The earliest publication on context in computing is a 1994 research paper by Schilits et al. (1994). Without providing any definition for context they stated their perception of context in terms of three important aspects of context which are 'where you are', 'who you are with', and what resources are nearby'. They categorised context into three categories, computing context, user context, and physical context and listed some examples such as 'lighting', 'noise level', network connectivity', 'social situation' to emphasise that context involves more than just the user's location.

Reviewing the research publications since then until 1999 shows no consensus on the definition of context. In their 1999 survey examining fourteen 'context-aware' applications, Dey and Abowd (1999) have concluded that researchers including themselves (Dey, 1998) have either provided definition of context 'by example' or provided synonyms rather than offering a concrete definition. After their view of the past they presented their definition of context as *'any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves'* (Dey et al., 1999). To Dey et al. the situation of a particular entity is characterized by location, identity, time and activity.

Chen and Kotz (2000) adopted (Schilits et al., 1994) categories and added two more categories, time context, and context history. ‘Time’ to represent the timing of when the contextual information was collected, and ‘history’ to refer to any past information saved in a persistent repository. Not convinced of the existing definition of context, Chen and Kotz gave their definition of context as ‘context is the set of environmental states and settings that either determines an application’s behaviour [active context] or in which an application event occurs and is interesting to the user [passive context]’.

Although further elaborations are given in the following chapters regarding ‘situation’ as opposed to ‘context’ we would like to make a clear distinction here between these two terms as far as PCEs are concerned. We refer to context as any useful piece of information that describes a particular element in a PCE. Whereas situation is referred to the collection of all contextual information and also information acquired through inference on the collected context.

Schmidt (2000) describes four different types of context. One is when the environment is active and detects a device and communicates with the detected device. The inverse case of the first type is the second in which the device senses the environment. Third, is a direct input by user to the application, and the forth is information retrieved from a database.

In response to (Dey et al., 2001), Winograd (2001) disputes their definition of context and argues that terms such as ‘any information’ or ‘characterize’ the situation that they use in their definition are broad. He goes on and says that context should be used in a more specific way, to characterize its role in
communication. Acording to Winograd ‘context is an operational term’, that is, something is context because of the way it is used in interpretation, not due to its inherent properties. Interestingly, Roy et al. (2011) subscribe to the Winograd definition of context in their context-aware agent system.

Henricksen et al. (2002) refer to context as circumstances or situations in which a computing task takes place. They stress on the lack of clarity in the literature about context and instead of offering their definition, they have provided some characteristics of context information in terms of association, which is a uni-directional relationship linking an entity to its attributes or other entities. They consider context information is more than the 'current state' of the context and therefore, can vary from 'atomic facts' to 'complex histories'. In (Henricksen and Indulska 2004; 2005), and (Henricksen et al. 2005) they categorise context according to their distinctive characteristics into four groups: sensed, static, user-supplied, and derived information.

Ranganathan and Campbell (2003) define context as the information "… about the circumstances, object, or conditions surrounding a user that is considered relevant to the interaction between the user and the ubiquitous computing environment".

Chen and Finin (2003) in their context broker architecture CoBrA define context as any information that can be used to characterise the situation of a person, a computing device, or a software agent. In another publication Chen et al. (2003a, 2003b) expressed their meaning of context as a location, its environmental attributes such as noise level or temperature, and the people, devices, objects and software agents it contains.

Ranganathan and Campbell (2003) divide the context into seven categories of physical (e.g. time, location), environmental (e.g. weather, light), informational (e.g. stock exchange information), personal (e.g. health, mood), social (e.g. group activity), application related (e.g. email) and system related (e.g. network traffic).

In an article about the importance of context in UC, Coutaz et al. (2005) explained why context is not merely the current state of a predefined environment and wrote that context is 'part of a process of interacting with an ever-changing environment'. They believe context is too complex to be pre-programmed and suggested explicitly coded responses to 'situations and contexts' should be replaced with 'a higher level, more knowledge-intensive use of machine- readable strategies coupled with reasoning and learning'(Coutaz et al., 2005).

Khedo (2006) categorises use of context in context aware applications into three different classes: presenting information and services, automatically executing a service, and tagging detected context information for future use.

Xiaosheng et al. (2006) have categorized context as manually acquired (e.g. the information provided by user) and automatically acquired (e.g. user's identity, location, or activity). They argue that many context elements are "raw" (e.g. time, location, temperature) because they are simply acquired through various forms of sensors; thus such elements are regarded as 'low-level' context. On the other hand, elements like activity are 'synthetic' context which is inferred through reasoning based on raw context; thus they are regarded as 'high level' context.

Paganelli et al. (2006) refer to the overall contextual information of an 'entity' as 'entity context' and any specific characteristic of the entity as 'context item'. According to them context items are of location, physical data, activity, instrumental context, or social context category, and context entity is composed of one or more of these context items.

Bolchini et al. (2007) who provide a survey of context models, distance themselves from the Dey and Abowd view and seems to subscribe to the (Coutaz et al., 2005) view that context is not just a 'profile' but it is also an 'active process'. They regard context as the element which impacts 'the way humans (or machines) act and how they interpret things.' With regards to the complexity of context representation, they advocate models that support only a specific context sub-problem and do not condone systems with a completely general aim that support any possible application (Bolchini et al., 2007).

Nguyen and Choi (2008) describe context as always-changing-phenomena where its change leads to a change of behaviour by the people who are subjected to it. Roy et al. (2008) define context as a set of data that provides a model of the real world and therefore they don't make any distinction between different types of information as far as context is concerned. To them the data related to a location is as important as the data related to what happens within the location.

In recent publications it seems that (Chen and Kotz, 2000) definition received more attention. For example Bellavista et al. (2012) in a survey of context data distribution for mobile ubiquitous systems adopted Chen's and Kotz's definition of contexts according to which context is a four dimensional space composed of computational context, physical context, time context and user context. Chen and Kotz consider context history applicable to some applications also, but Bellavista et al. have disregarded this dimension.

Mehra (2012) describes context as the information halo that implicitly surrounds objects of interest and includes those pertinent supplementary facts, rules or axioms whose consideration makes our situations understandable by devices, people, or organisations seeking to provide us with content or services. Mehra argues that in addition to 'situational context' there is also the 'large context' which is a historically, socially and semantically expanded model of a user's context. Educational affiliation, sports team loyalty, locations visited are some of the examples of 'large context' that their inclusion not only will improve personalisation of delivered 'content, alert and advertisement' but will also simplify the interaction with users.

We have mentioned in section 1.3 in Chapter 1 that one of the objectives of this thesis is to define a formal computational model which will allow representation of computationally significant semantics of any situation in PCEs for the purpose of delivering situation-specific services. The representation of the semantics of a situation requires correct contextual information about the particular situation in the PCE. Availability of context is therefore, essential for the success of PCEs. In the literature, 'context awareness' is used as a qualifying adjective for describing software applications capable of adapting themselves to the context. Our view of awareness and use of context in PCEs is different, but in the following section we present what is commonly perceived as context awareness.

2.3.2 Context Awareness

Schilits et al. (1994) define this form of computing based on their PARCTab pioneer experiment (Schilits et al., 1993) as systems that adapt according to the location of use, the collection of nearby people, and accessible devices. Such systems will react automatically, according to them, to any changes to the environment. They have categorised context-aware applications into four categories. These categories are proximate selection, automatic contextual reconfiguration, contextual information and commands, and context-triggered actions. In the proximate selection category, the user's current location determines the area from which located objects will be identified for ease of selection by the user. In the automatic contextual reconfiguration, the connectivity of devices in the environment is managed automatically. That is, adding or removing components or changing connections between existing components. The contextual information and commands category exploits the user's current location to respond to context information queries and commands. Shilits et al. describe the context-triggered actions category of applications as systems that support 'simple IF-THEN rules used to specify how context-aware systems should adapt'.

Pascoe (1998) identifies four generic contextual capabilities when he describes his prototypical development of a wearable computing system. Sensing, adaptation, resource discovery, and augmentation are these capabilities that context-aware applications should support. Harter et al. (1999) considers context-aware applications as those which adapt their behaviour to a changing environment.

Dey and Abowd (1999) believe the goal of context-aware computing 'should be to make interacting with computers easier'. They recommended that a context-aware application should collect contextual information 'through automated means' and let the application designer decide what information is relevant and how to deal with it'. They express the relevance of context-awareness to UC as 'to best support the human–computer interaction'. With regards to features of context-aware applications, Dey and Abowd combined ideas from (Schilit et al., 1994) and (Pascoe, 1998) and suggested three key categories of context-aware features that a context-aware application should support. These are ‘presentation of information and services to a user, automatic execution of a service, and tagging of context to information for later retrieval’ (Dey and Abowd, 1999).

Chen and Kotz (2000) provide two context-aware computing definitions to reflect their active and passive definition of context. They define active context awareness as an application that automatically adapts to discovered context, by changing the application’s behaviour, and define passive context awareness as an application that presents the new or updated context to an interested user or makes the context persistent for the user to retrieve later.

Dey (2001) gives a more generalized description of a system to be context-aware. He believes if the system uses context ‘to provide relevant information and/or services to the user, where relevancy depends on the user’s task’, then it is a context-aware system. In describing their context broker architecture, CoBrA, Chen and Finin (2003) describe context-aware computing as allowing systems to act more autonomously and take initiative, but informed by a better model of what their users need and want. This is an example of context model inflexibility.

The environment according to Khedr and Karmouch (2004) becomes context-aware when it can capture, interpret, and reason about physical characteristics, such as location, the system, such as applications running, and the user such as presence.

Hong et al. (2009) adopted the definition by Byun and Cheverst (2004) and also Khedo (2006) that a system is context aware if it can adapt its functionality to the current context it detects and interprets. Mehra (2012) emphasizes that the real power of context-aware computing is in its automated reasoning based on facts and rules describing the environment around a user or event. He considers ‘gathering, sharing and obtaining’ data about ‘large context’ is the challenge the community is facing to realize context-aware computing.

Hong et al. (2009) provides a literature review of 237 journal articles between 2000 and 2007 in which context-awareness is a core essence of the article. They have categorised the focus of this research into five abstract layers. These layers in order of number of publications are, concept and research, application, network infrastructure, middleware, and with the least publication user interface.

There have been many suggestions by researchers designing and implementing context-aware PCE about the way different elements of a PCE communicate with each other to support the context-aware applications. To this end, several architectures, frameworks or middleware tools were developed and surveyed (Henricksen et al., 2005), (Singh and Conway, 2006), (Baldauf et al. 2007), (Kjær, 2007), (Miraoui et al., 2008). We found Henricksen et al. and Miraoui et al. more detailed in terms of categorisation of the architecture according to the essential features pertinent to PCE. The comparison of different architectures that authors of (Miraoui et al., 2008) have done is based on context abstraction level, communication model, reasoning system, extensibility and reusability.

Even by 2012 there is no agreed upon definition on context and context awareness. In their survey on current research on context data distribution in mobile ubiquitous environment, Bellavista et al. (2012) acknowledge this lack of consensus and offer their own definition. According to them, context awareness is 'the ability to provide services with full awareness of the current execution environment'.

2.3.3 Context Modelling

Context aware application systems are qualified with detecting, collating, storing and disseminating contextual information at the lowest level and aggregating it into increasingly more abstract models (Khedo 2006). To support such a system a context model is required to handle contextual information.

Harter et al. (1999) present a platform that enables applications to constantly monitor users as they move around a building. Their location-aware system is based on 'a persistent distributed object system' which will provide context accessible to various applications. They used object-oriented approach to model the contextual information since they view the environment to consist of a collection of real objects.

Henricksen et al. (2002), after exploiting Entity relationship model and the class diagram of UML, have concluded that these information systems modelling techniques are ‘neither natural nor appropriate’ for describing context. To overcome the shortcomings of these techniques they have devised their own object-based modelling technique using special constructs designed with the characteristics of context in mind. According to their abstract directed graph model for context information in terms of its characteristics in PCE systems, context is characterised through a number of associations, which is a uni-directional relationship linking an entity to its attributes or other entities. These associations are divided at a high-level abstraction as static or dynamic associations. The dynamic association is divided further to sensed, derived and profiled association. Temporal association which is attached to a time interval and dependencies between associations also play a role in their model.

In a follow up work to (Henricksen et al. 2002), McFadden et al. (2004) developed a context modelling technique that provides two levels of abstractions. They called these levels as ‘facts’ and ‘situations’. Situations, according to them, are defined by constraints on context facts expressed using predicate logic. In their distributed approach to the development of context-aware communication applications, they used their own developed Context Modelling Language (CML) and some Java technologies such as JDBC and RMI.

Chen et al. (2003a and 2004b) have expressed their reason for using OWL ontology to model context for 1) it is much more expressive than RDF or RDF-S (Brickley and Guha, 2010) allowing to build more knowledge into the ontology, 2) OWL has been suggested by W3C as the standard language, 3) OWL provides a means to share context knowledge, and 4) OWL ontology helps their CoBrA context broker to reason about context and detect knowledge inconsistency.

In SOCAM, at the bottom of the architecture (Gu et al., 2004) are ubiquitous sensors. These sensors will feed the internal context service providers. The internal and external context providers deliver their service to the middleware, which is called context interpreter. The context interpreter consists of a context knowledge base and a reasoning engine, which is based on some inference rules. At the top of the architecture are the context–aware services that can use contextual information directly from the context providers or from the middleware and adapt themselves to the context. In SOCAM context knowledge base stores contextual facts in a persistent relational database. Context ontologies are divided into upper ontology and domain specific ontologies. The former has computing entity, location, person and activity as its key classes that Gu et al. consider to be common in all PCEs. Preferences of the user are not represented in the upper ontology of SOCAM. Although the authors acknowledge that 'context may quickly become out-of-date', they do not offer any solution for this key issue. This is an example of contextual information in traditional context-aware applications can quickly become outdated. As SOCAM is service oriented, it has to offer a service-locating service to discover available contextual information. It is not clear how SOCAM can maintain this provision in PCEs where different services are required at any given moment. This is an example of Provision of insufficient contextual information will result in delivery of unexpected services to the user.

The ontology-based CoBrA architecture (Chen et al 2003a) serves agents, which could be applications within a device or services provided by devices, through a centralised broker. The reasoning mechanism that CoBrA developers offer over context information for resolving inconsistent context information, applying privacy policies and inferring additional context information, is within OWL ontologies without additional rule support. Provision of contextual information to distributed agents when CoBrA encounters complex situation in which multiple context information are interlinked, is therefore limited. CoBrA uses a central database to store and fuse collected contextual information. The COBRA-ONT (Chen et al 2004a) ontologies, in which time is an important concept, is predefined and in all situations all contextual information, even if they may not be relevant to the situation, are collected from various devices distributed in the PCE. This unnecessary information overload would make processing the context to deliver expected service to the user very expensive. This is an example of Unnecessary information overload caused by pre-defined contextual model and architecture that detects all contextual information would make processing the context to deliver expected service to the user very expensive.

SOCAM and CoBrA are examples of PCE architectures for context-aware applications that are specific for a domain and therefore require additional effort for their adaptation to other domains.

Sense Everything Control Everything, SECE (Boyaci, et al 2012) although categorised in the literature under context-aware computing, has little to offer to represent a holistic view of the context. SECE is designed to facilitate user's communication and is an event-driven system which acts on user's behalf automatically, and is a non-user involvement system. It is a rule based system in which actions take place without any interaction from the user on the basis of the description of the event as the body of a rule.

As Khedo (2006) indicates a challenge in making context aware applications is to go beyond reading one sensor data and act upon it. It is more about 'context recognition' to detect for example complex activities and to differentiate between different users not defined by a single sensor. SECE is an example of a simplistic view of PCEs in which context awareness is more than localization.

(Henricksen and Indulska 2004; 2005), and (Henricksen et al. 2005) present a different context modelling approach. They adopted and extended Object-Role Modelling and developed a graphical context modelling approach CML. They have implemented their prototyping system in Java, and used a relational database management system to store contextual information to be queried by context-aware applications. This is another example of unnecessary information overloading.

Earlier 'context-aware' applications that were more localization-aware applications such as The Active Badge (Want et al., 1992), ParcTab (Schilits et al., 1993), Stick-e Note (Brown et al., 1997), and Cyberguide (Abowd et al., 1997) did not support context abstraction and therefore, no context model for context representation is present.

Fahy and Clarke (2004) provide an abstraction of contextual information for their CASS system that was modelled using object oriented techniques. Hydrogen three-layered architecture (Hofer et al. 2002) also uses an object oriented model to represent context. (Wang et al., 2004), (Nguyen and Choi, 2008), and (Khalil et al., 2008) provide more detailed insight into the process of modelling context, but (Strang and Linhoff-Popien, 2004) and (Bolchini et al., 2007) provide more detailed analysis on different classes of context models.

Strang and Linhoff-Popien (2004) classify Context models by the "scheme of data structures used to exchange contextual information" and provide six categories as: Key-Value Models, Mark-up Scheme Models, Graphical Models, Object Oriented Models, Logic Based Models, and the last but surely not the least, Ontology Based Models. Strang and his colleague evaluate instances of their classification based on six criteria of a) compatibility with a distributed application environment, b) capability to partially validate contextual knowledge on structure, c) richness and quality of information, d) model's ability to compensate data incompleteness or ambiguity due to hardware inefficiencies, e) level of formality and finally f) applicability to existing environments. In their evaluation they brand key-value models as "weak" due to disability to fulfil the first five criteria and regard the ontology modelling as "the most promising asset" for context modelling in PCE.

In their survey of context models Bolchini et al. (2007) have evaluated and classified sixteen applications based on a proposed analysis framework; the classification is made based on the use of context (i.e. context as a matter of channel-device presentation, as a matter of location and environment, as a matter of user activity, as a matter of agreement on sharing a context). They have concluded that systems which are aimed at providing non-proprietary support to context data tend to be ineffective. They state: "Different context sub problems and applications have almost incompatible requirements…; as a consequence the context model should be chosen depending on the target application." This survey shows how common the issue of the inflexibility of context models is in context-aware applications.

Baldauf et al. (2007) also provide a brief survey of context-aware systems. Authors review the architecture design and context modelling of some of the solutions and concluded that in the reviewed models although abstract context sources are used,but only physical sensors (detecting from physical sensors) are mainly used in practice, and virtual (context information detected from software applications or services) and logical (inferring additional information based on physical and virtual information and some other pre-defined information) sensors need to be looked at too. The other observation they made is that every system uses its own format to describe context and its own communications mechanisms between context sources and users.

Having different format to describe context in a PCE that naturally deals with heterogeneous devices has raised concern among the researchers. Roy et al. (2008) stress on this issue and point out that moving from controlled intelligent spaces to open intelligent spaces will create more problems if the heterogeneity of context models is not addressed.

We would like to conclude this section by referring to the joint view of three renowned researchers, who first coined the term Semantic Web in 2001, on traditional knowledge representation. Considering the requirements of the Semantic Web, Berners-Lee et al. (2001) dismissed the traditional knowledge-representation systems for being capable of handling the ever expanding web of information. These systems' centralized nature that requires all users 'to share exactly the same definition of common concepts' is their major concern. They believe such systems will be 'unmanageable' and concluded that ontologies are the most appropriate way of modelling when complex set of concepts are involved.

2.4 Semantic Web

Visionary Tim Berners-Lee depicted his thoughts about the future Web in 1989 as shown in Figure 2.1. The links between the nodes in the diagram, like 'wrote', 'refers to' and 'describes', show his proposal to his employer on how to annotate documents in the Web. The motivation for his proposal was increasing amount of unstructured information in the company's intranet network which was not supported by any meta data. Tim Berners-Lee was proposing that his idea will improve collaboration beween different departments and staff as they would have more information about each document they access. The immediate solution to the HTML-encoded Web was XML technology (W3C). Although the design objective of the new language was encoding for machine processing, it does not have the necessary sound formalism to provide meaning of the terms, for example, ‘wrote’, ‘refers to’ and ‘describes’ are meaningless to a machine processing the XML code.

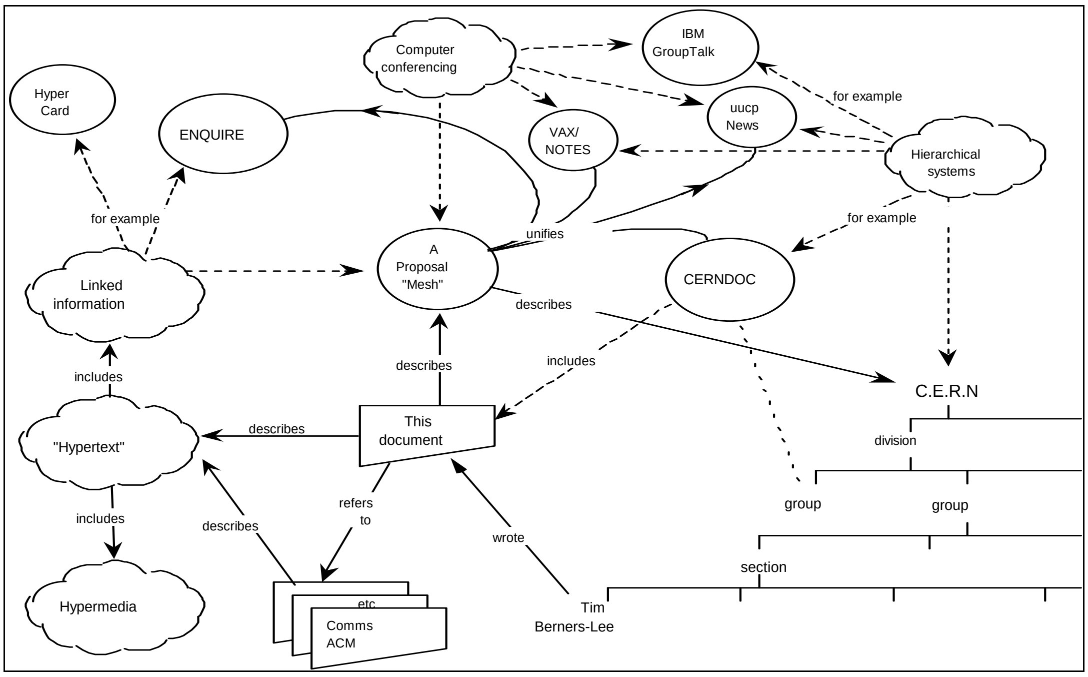


Figure 2.1: Tim Berners-Lee’s original Web information management view (Berners-Lee, 1989)

In other words, XML and XML-Schema could not attribute meaning to tagged terms of a document to be able to share knowledge. However, the other issue with XML was that it could only tagged the internal content of a document and therefore external sources linked to a document could not be represented. To address this issue the RDF (Resource Description Framework)(W3C, 2004c) was promoted. RDF is graph-based and comprised of nodes and edges. Each edge is a binary relationship between two nodes. The complete graph is a set of binary statements based on XML syntax. The structure of each statement is a triple: subject, predicate, object. The structure, therefore, provides a data model represented in triples. Subjects and objects are the nodes and predicates are the edge linking subjects to objects. Subjects and predicates are URI (Uniform Resource Identifier) resources but an object can be either a URI resource or literal values.

So, when the Semantic Web (Berners-Lee et al., 2001) was introduced XML and RDF technologies were already available. Tim Berners-Lee's and his colleagues' vision about the Semantic Web was to improve the traditional Web so that the huge amount of information available in the Web can be shared. Enterprises, businesses, public offices, health centres, or individuals can then be strengthening by exchanging and sharing their knowledge with each other. To this end, XML, RDF and RDF-Schema were not sufficient. The philosophy they were advocating was a 'Web of data' that is structured and formalised to be processed by computers. Given that XML and RDF were both machine processable and available at the time, any new formalism had to be based syntactically on these technologies.

To ascribe meaning to sources of RDF triples, Berners-Lee et al. recommended the Semantic Web stack shown in Figure 2.2. Ontology layer is a key layer of this architecture. Although AI community has used ontology modelling for quite some time, there was no standard and guideline on use of a formal specification language for ontology. It did not take long before W3C announced its standard Web ontology language, OWL (W3C, 2004), (W3C, 2004a), (Kadak, T. and Kleerova, 2006) and only recently suggested OWL2 as the revised ontology language (Grau et al. 2008). The use of ontology in the stack is to provide vocabulary and formal meaning to concept used in the taxonomical structure of the ontology. To infer new knowledge based on existing knowledge, using ontological model, a logical rule layer was felt necessary. A compatible rule engine with OWL ontology that execute the logical rules reason about the semantics of the environment to deliver a situation-specific service. The stack depicted in Figure 2.2 also shows that any rule to support reasoning is based nontology.

Like any other technology, the rate of SW technology success depends on how widely it has been adopted by key industries. SW technologies have already started conquering various computing domains and have shown that, apart from managing

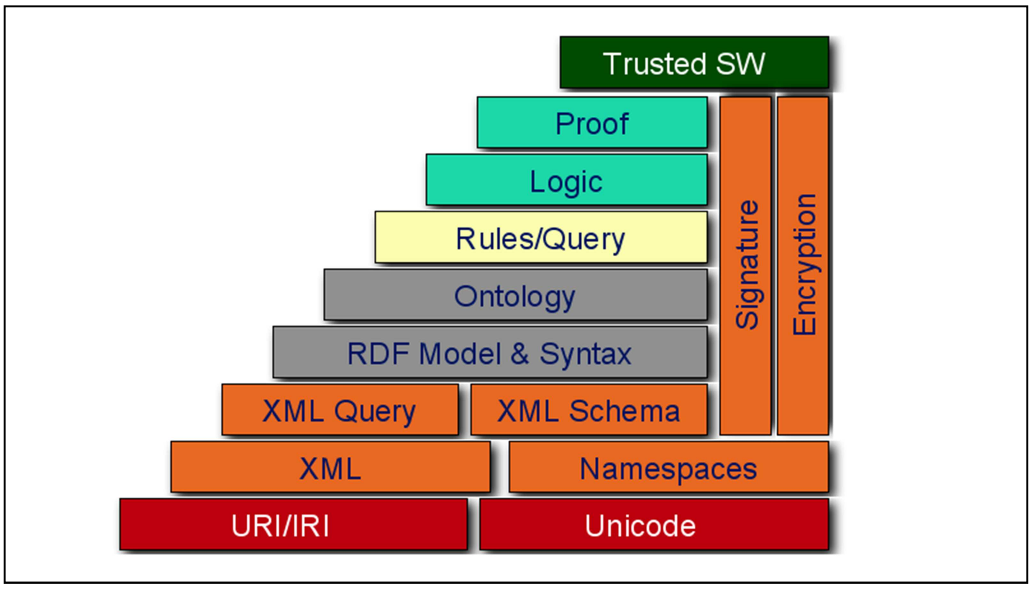


Figure 2.2: Semantic Web Stack (W3C 2004b)

the semantics of the Web, and bringing ‘structure to the meaningful content of Web pages’, they can be extremely powerful for building semantics of any computational environment. Applications of SWTs across domains range from business intelligence and semantic management to interoperability, communications, and data sharing. The expressive capability of OWL to allow context information to be represented for context-aware applications, on one hand, and the formalised structure of knowledge representation allowing reasoning upon acquired contextual information, on the other, demonstrated that SW technologies, including OWL, are suitable technologies for the SW requirements.

2.4.1 Ontology Definition

In the information systems community the term ontology refers to a particular type of conceptual model of some entities of interest or ‘concepts’ as in the famous ‘Ogden Triangle’ (Ogden 1923). The model represents the entities in a way that it facilitates shared understanding and sharing information about the entities. The representation is formalised by following some standard set of constructs. Gruber (1993) describes ontology as ‘specification of conceptualisation’. Formal specification of domain ontological models requires a language. The language has to be expressive enough to allow the representation of the domain but at the same time decidable to allow reasoning on existing contextual information to detect any inconsistencies among acquired information, but also to infer new knowledge from the existing knowledge.

2.4.1.1 OWL Ontology

The first ontology language recommended by W3C was RDF (W3C, 2004c). Another recommendation of W3C is RDF Schema (W3C, 2004d) that adds some features as metadata to RDF to complement it. Same as RDF, the statements in RDFS are binary relationships; i.e. predicates are binary. However, despite some added features to RDF, RDFS is undecidable because of the fragment of the logic it is based on. Decidability however, is a key feature of applications generated upon ontologies. Any fully-fledged PCE system built on ontologies require support for reasoning, without which the extent of support they provide for users will be limited. This shortcoming in RDFS necessitates the development of a new language. The first version of OWL introduced in 2004, then in 2009 the second version was announced followed by W3C announcement in December 2012, to make OWL2 as the consortium's recommendation for ontology language (W3C, 2009).

The first OWL (OWL 1 as now recognised by W3C) has three varients, OWL Lite, OWL DL, and OWL Full. OWL Lite is subset of OWL DL, and OWL DL is a subset of OWL Full. In OWL 2, where new features and new vocabularies have been introduced, three more sublanguages of OWL DL are also offered. It is outside the scope of this thesis to elaborate more on this, but keen readers are referred to (Baader et al., 2007) for more detailed information on Description Logic (DL), to (W3C, 2004e) for OWL in general, and to (W3C, 2012a) for more information on OWL 2 new features and vocabularies.

We provide here a brief explanation of some of the constructs of OWL, and refer interested readers to (W3C, 2012b) for extensive information. As Figure 2.1 shows and mentioned earlier OWL ontology is based on RDF and the data model represented in RDF is a set of triples. Therefore, as it is for RDF, it is the same for OWL that resources whether being 'subject' or 'object' or 'predicate' are the key building blocks of OWL. The terminology W3C uses for these resources in OWL is 'entity' and considers entities as 'fundamental building blocks of OWL' (W3C, 2012b). As depicted in Figure 2.3 an ‘entity’, or ‘concept’ as is sometimes referred to in the literature, can be a ‘class’, ‘object property’, ‘data property’, a ‘data type’, or ‘named individual’. Unlike other entities, ‘annotation property’ entity is not expressed formally in OWL. In that, we will not discuss this type of entity any further. As the diagram shows each entity has one Internationalized Resource Identifier (IRI). IRI has replaced URI in OWL2 (W3C, 2012b).

2.4.1.2 Constraints and Assertions with OWL

As literals are not individuals in OWL, they are not shown as a specialised type of ‘Entity’. In OWL only individuals make members of an OWL class. It is worth mentioning here that although the ontology schema is represented in OWL the individuals that populate the ontology are represented in RDF.

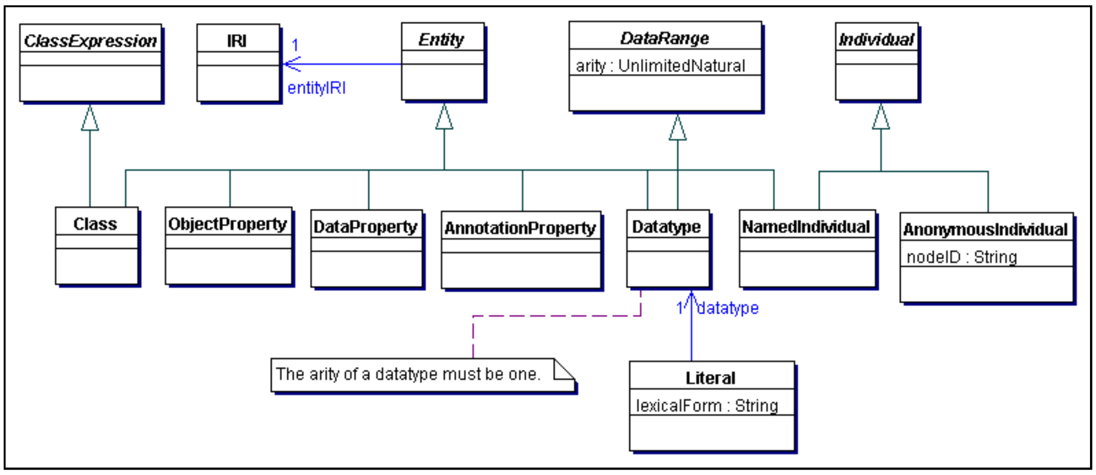


Figure 2.3: Building blocks of OWL using UML notations (W3C, 2012b)

An object property is a binary predicate that relates two individuals being member of the same or different classes. Each object property has two predefined properties, `rdfs:domain` and `rdfs:range`. Domain restricts the subject of the predicate to be an individual of a particular class, likewise, range restricts the object of the predicate to be an individual of a particular class. In Listing 2.1 the description of two object properties, `belongsTo` and `isCurrentlyIn` in RDF/XML format are presented. The first object property indicates that the ‘subject’ of the predicate `belongsTo` is always an individual of class `OBJECT`, and the ‘object’ of the predicate is always an individual of class `PERSON`.

```
<owl:ObjectProperty rdf:about="&Ontology1356022592295;belongsTo">
        <rdfs:domain
rdf:resource="&Ontology1356022592295;OBJECT"/>
        <rdfs:range rdf:resource="&Ontology1356022592295;PERSON"/>
</owl:ObjectProperty>

<owl:ObjectProperty
rdf:about="&Ontology1356022592295;isCurrentlyIn">
       <rdfs:range
```

Listing 2.1: An example of description of two object properties in RDF/XML format

An example of asserting named individual in OWL in RDF/XML format is shown in Listing 2.2. In this example an individual of class `Heater`, `heater152` is asserted.

```
<owl:NamedIndividual rdf:about="&Ontology1356022592295;heater152">
    <rdf:type rdf:resource="&Ontology1356022592295;Heater"/>
    <status rdf:datatype="&xsd;string">off</status>
    <belongsTo rdf:resource="&Ontology1356022592295;margaret"/>
    <isCurrentlyIn rdf:resource="&Ontology1356022592295;room101"/>
</owl:NamedIndividual>
```

Listing 2.2: An example of asserting a named individual in OWL

The object properties described in Listing 2.2 in addition to some datatype properties are added to describe and represent the object `heater152`.

The Listing 2.1 and 2.2 is shown in graphics in Figure 2.4. Please note that the Heater class is related to `OBJECT` class through the following predicate:

`<rdfs:subClassOf rdf:resource="&Ontology1356022592295;OBJECT"/>`

because of this subclass relationship any individual of `Heater` is also an individual of `OBJECT`.

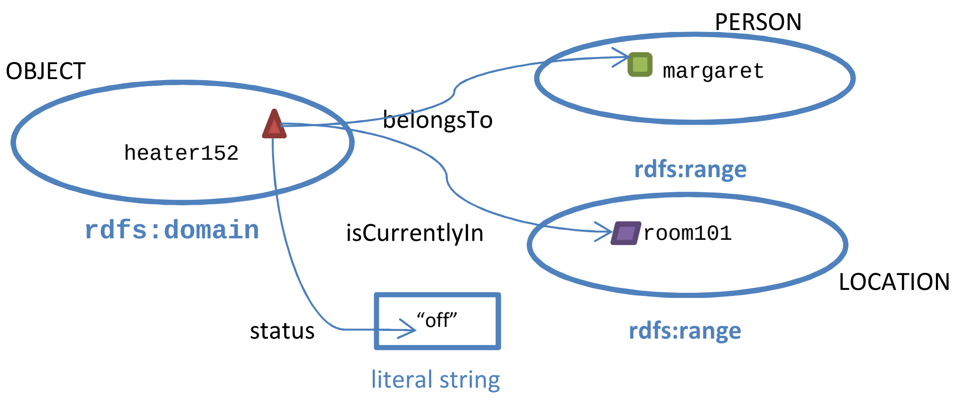


Figure 2.4: Graphical representation of Listings 2.1 & 2.2

2.4.2 SWRL Rule

OWL ontologies cannot provide advanced reasoning when complex interrelationships between several concepts of ontology are involved. Whether SW tecknologies are used to add structure to the content of the Web, or are used in any PCEs for making use of the semantics of the computational environment, shortcomings of ontologies for reasoning upon existing context are compensated with some deductive rules. Rules are a set of IF, THEN statements where one or more than one statement (or premises) represent the condition (also known as body or antecedent of the rule), and one or more statements represent the action (also known as head or consequent of the rule). These rules can be set for a variety of purposes. They can be used, for example, to apply user preferences, to apply business or service policies, to implement some security measures, to maintain smooth network operation, or they can be set to trigger an actuator in response to a particular situation in a PCE.

Despite its expressive power and reasoning mechanism, complex rules where several OWL 'entities' are involved cannot be handled by OWL ontology. This deficiency on one hand, and the necessity for any rule to be added on top of the ontology to run using the same reasoning engine, on the other, required a new language for writing rules in SW. Therefore, W3C recommended the Semantic Web Rules Language, SWRL, as another language that allows the integration of OWL ontology and a rule layer that sits on top of it (Horrocks, et al. 2004). SWRL, which is also based on DL, complements OWL ontology. How non-monotonic SWRL and monotonic OWL can work together hand in hand to offer a decidable reasoning system is outside the scope of this thesis. Interested readers are referred to (Eiter et al. 2007) which provides a good overview of rules and ontologies for the SW.

2.4.2.1 The Role of Competency Question (CQ)

The competency questions provide the information necessary to develop a new ontology or extend an existing one. The set of CQs will therefore specify the criteria for the design of the ontology in terms of its terminology and constraints; i.e. to ensure the design decisions made for the ontology are correct. This means that the CQs are to verify whether the ontology meets the requirements specified in the questions, and not to direct towards a particular design. In other words, any particular set of CQs cannot be related to one and only one ontological model. As the ‘words’, or ‘phrases’ used in the CQs will form the terminologies used in the formal ontology specification, the formulation of the competency questions to capture all necessary requirements for the design of the ontology is very important.

CQs are usually defined in a stratified manner (Uschold and Gruninger, 1996), so that the rational for the lower level questions is how the answer to the CQ is used to answer a higher level, more complex CQ. We have explained the role of RDF technology in SW in previous sections. Sentences, phrases and words in CQs will determine the RDF triples (subject, predicate, object). For example, the informal sentence *“the status of the heater is off”, which is an excerpt from a CQ is translated into a formal CQ:* `status (? h,"off")`. This entails that the concept Heater must have data type property status.

We conclude this section with a statement from Tim Berners-Lee (2000) to stress on the importance of capturing terms of the CQ to realize the SW vision and use of SW technologies in PCE.

> “The philosopy was: What matters is in the connections. It isn’t the letters, it’s the way they’re strung together into words. It isn’t the words, it’s the way they’re strung together into phrases. It isn’t the phrases, it’s the way they’re strung together into document.”

2.4.3 Ontology Applications

OWL ontologies are increasingly being used across domains, more noticeably in health domain. Ontologies, particularly in the health care domain, are used mostly for vocabulary specification or for serving context aware applications. Examples for the former are, the well-known Gene Ontology Consortium with the role to produce a dynamic, controlled vocabulary for all being gene (GO, 2000), a big biomedical vocabulary like Thesaurus for cancer research (Hartela et al., 2005), large scale clinical terms SNOMED (Spackman et al., 2000), or more to serve the needs of a particular community such as phenotype ontologies (Mungall et al., 2010). Examples where ontologies are used for more than vocabulary purposes, and used in PCE applications to serve users are (Chen and Finin, 2003), (Wang et al. 2004), (Ko, et al., 2007), (Paganelli and Giuli, 2007), (Bardram and Christensen, 2007), (Niyato et L., 2009), (Arnich et al., 2010), (Polza et al., 2010), (Coronato, 2010), (Romero, et al., 2011), and (Zhang et al., 2011).

Chen and Finin (2003) have used ontology for modeling contexts using OWL language. They consider ontologies as ’key requirements’ for building context-aware PCE systems. In another project Chen et al. (2004b) have developed a shared ontology SOUPA (Standard Ontology for Ubiquitous and Pervasive Applications) for supporting UC and PCE applications. They believed that their generic ontological model, that was developed using OWL, can be a step towards the standardization of a shared ontology to be reused by ontology-driven application developers.

In their agent-based system to negotiate context information, Khedr and Karmouch (2004) use an ontology agent that provides the semantic functionalities that other agent can use to represent and share context in the system. To represent a unified context model, they have taken an ontology-based approach, and defended their choice by arguing that other approaches do not support extensibility and interoperability with other context-aware systems.

Wang et al. (2004) use a set of ontologies to describe and represent contextual information within their architecture, SOCAM. These ontologies are divided into domain-specific and generalised ontologies.

In the ontology based U-HealthCare, Ko et al. (2007) have defined three context ontologies for Person, Device and Environment, and their model does not include the element of time. Their ontologies are semantically divided into general context ontologies and domain context ontology, similar to (Wang et al., 2004).

Paganelli and Giuli (2007) provide a more detailed ontological model than (Ko et al., 2007) for their ‘Kamer’ project. They have provided four ontologies to represent patients, other people patient encounters with, the physical environment, and an alarm management ontology. In the ‘Patient Personal Domain Ontology’ they include patients physiological information. They have used OWL language and some first order logic rules to reason upon the context. Some of ontological entities, particularly classes are being repeated in different ontologies to run predefined rules.

A very detailed, and yet, general view of the ontologies for pervasive systems is explained by (Stevenson et al., 2009). They have adapted requirements for modelling context that (Strang and Linnho- Popien, 2004) and (Henricksen et al., 2005) have elaborated in terms of "capturing the quality of data and supporting temporal data". In addition to these requirements for capturing properties of sensed data, Stevenson et al added requirements for modelling properties and capabilities of sensors too. Some of the ontologies in the Stevenson et al.'s Ontonym model are Time, Location, People, Event, Resource, and Device. They suggest that an approach to evaluate a context model is through the application that is supposed to use it, and how the model "fits" the application.

We would like to add that not all ontology based applications and context management use OWL language. Korpipää et al. (2003) for example use RDF for context representation. Jahnke et al. (2004) also offers an ontology-based context aware system in which the context representation is divided into two ontologies, domain dependent and domain independent ontology. They have used a graph-based database for storing contextual facts. They are using their own encoding for contextual representation and for communication between the context sensors and the context users. However, since W3C recommendation in 2004 endorsing OWL as the ontology language standard, efforts to develop ontologies in other languages have declined.

2.5 Summary

In this chapter, we have provided different perspectives of pervasive and ubiquitous computing, and how this new paradigm challenged previous views on computing. The diverse characterisation of pervasive computing environments and systems, gathered through a collection of published documents was also given in this chapter. Realisation of pervasive computing depends on context-awareness. Various definitions and viewpoints of this concept is reflected in several prototypes developed. In most early context-aware applications and in mobile-computing the study reveals that context-awareness is used as a synonym for location-awareness. There is a growing trend in the pervasive computing community to look far beyond location as context. Different approaches to context modelling and growing interest in use of ontologies to abstract contextual data was also covered.

Increasing inclination towards the use of Semantic Web technology as a software solution to problems and concerns of pervasive computing systems is also covered in this chapter. A brief explanation of some of the SW technologies that interest us, particularly RDF, OWL ontology and SWRL rule along with concept and vocabulary used in the realm of SW is also given in the chapter.

CHAPTER 3
PROBLEMS WITH PERVASIVE COMPUTING

In this chapter we analyse the research problem from two different perspectives. Firstly, in section 3.1. we debate expectations we all have from pervasive computing and we single out only those which cannot be delivered through the traditional approach to creating computing environments. We use word "traditional" to refer to all computing practices, which have existed in SE for developing applications in businesses, science, governance and leisure, in the last 3 decades. They all heavily depend on numerous transaction processing, manipulation of databases and knowledgebases, information retrieval, scientific calculations and similar. We do not wish to claim that PCE cannot be developed using the traditional SE principles, but we wish to point out that computational environments have changed significantly and our well established way of creating computational solutions may not be appropriate for creating PCE. Secondly, in section 3.2 we look at pitfalls in the field of context awareness, which promised to address the real nature of PCE, but delivered results only fractionally. Consequently in section 3.3 we summarise the shortcomings of pervasive computing.

We would like to think that the way forward is to create a common consensus on the characteristics of PCE (Section 3.4.1) and define the role of situations in them (section 3.4.2.). We would also like to bring forward our vision on what exactly we need to compute in PCE (section 3.4.3). Our perception of PCEs is drawn from our own research (Shojanoori et al. 2010, 2012),(Shojanoori and Juric, 2013) and outcome of discussions in section 3.2.1 and 3.2.2.

3.1 Expectations from Pervasive Computing

It is difficult to find published sources of information which itemise expectations we all have from pervasive computing. However, from the background reading and experiences of defining pervasiveness across different domains (Shojanoori et al. 2010, 2012),(Shojanoori and Juric, 2013) (Koay et al., 2010a,b), we have grouped and contrast them with common perceptions of traditional computing in the paragraphs below.

Defining inputs dynamically: Computing, from the outset until very recently, has been regarded as the process of achieving some desired output when some pre-defined set of inputs is provided. Programmers have to define and program functions that convert the inputs to outputs. The computers were designed to be general-purpose computers, which at the same time are expected to compute different functions, and therefore have, sometimes “difficult-to-design” algorithms, ready. Obviously, these programmed algorithms would work only if the input to them is known and very well defined. By contrast, in the 21st century computing, the set of inputs for computation is not always known in advance. For example, in an intensive care unit, where the health status of patients are constantly monitored, different supporting devices, depending on the patient’s current status, might be needed and consequently they might produce different types of input which should be accepted at run time. Also, the choice of devices might not be known in advance, i.e. before we set up a PCE, but we should be able to use these devices as situation requires. In cases of using biometrics and similar technologies in Border Control environments, when we identify a passenger through face or iris recognition, they both produce viable input sources for the recognition system, which can be used interchangeably or altogether. In some cases we may use iris recognition because of relatively lower reliability of face recognition, which simply might not work for all passengers.

Computing output not guaranteed: The general-purpose computers always return a result, be it information retrieved from a persistent repository or some physical output such as dispensed money at the end of a transaction using an automated teller machine. Although individual computational devices in PCEs may have outputs, computing functions may not have any output at all. For example, if there is no change of condition of a particular patient who is monitored remotely, his/her PCE will not take any “action”, which means that PCE will produce no output. However, this lack of “output” doesn’t mean that there has not been any computation at all. Computing to infer that “*there has not been any changes*” is an example of not having any output as a result of the computation.

Distribution of devices and computing: In traditional computing, computation is limited to general-purpose computers. Their centralised functionality is not distributed among devices embedded with computing capabilities designed for a specific task. By contrast computing devices in pervasive computing are equipped with their own microprocessor and therefore have a specific purpose, which is defined within their own computing function. This is actually a key feature that may distinguish pervasive from traditional computing. For example, a thermometer sensor that provides body temperature of a person could be augmented to produce contextual information rather than raw data. That is, instead of providing the exact number (e.g. temperature, in remote patient monitoring) it may give an information whether the person’s body temperature is normal, high, or low.

Pervasiveness of user-specific devices: In traditional computing many users share the same general-purpose computing device. Opposite to the practice in traditional computing, computing devices, or devices embedded with some computational power, by and large, are user-specific and therefore are not shared by several users. At the same time each user is surrounded by numerous devices, which makes his/her environment pervasive, but full of user-specific devices. For example, a patient in a hospital ward, who has just had an operation, may carry or be attached to a number of devices, and each of them may monitor some aspect of the patient’s health. They are all specific to that patient and not shared by anyone else.

Flexible interfaces: Software applications generated from a traditional computing system make use of very limited ways of interaction with the environment. Direct keyboard or mouse input from the user are by far the most common ways of providing input to computing functions in traditional computing. However, to support users of pervasive computing system with as much flexibility as possible, there cannot be constraints on the type of interfaces between the devices generating information and the software applications that are making use of them. For example, identifying people in a particular PCE can be done through face recognition, by voice tone, by finger print, or by iris of the eye. In all these cases we use devices with completely different interfaces for achieving the same goal: identify a person. Normally all these interfaces are parts of various devices, integrated into a PE, which enable the recognition.

Implicit interaction: While in traditional computing the interaction between human and computer is explicit, in pervasive computing where the user is surrounded by devices embedded with computing capabilities, the interaction between the human user and information is implicit. For example, a patient who is being monitored remotely and falls unexpectedly may not interact with any system to notify his unfortunate incident. The sudden change of the patient's body position can alert the remote monitoring system and there will be an implicit interaction between the patient and his/her PCE.

Redundancy of historical information: Traditional computing applications, more often than not, rely on persistent data, whereas pervasive computing may compute the moment without any regard to the historical information. This, by no means dismisses the usefulness of historical information in a PCE. The issue is that the PCE is not responsible for it, and its provision has to be taken care of at the level where we run a software application which supports the PCE. For example, a PCE user might decide to block incoming calls, and right at the next moment he/she changes his/her mind and wants to receive calls. The decision made in the previous moment cannot be judged for the next. Or even if he/she blocks incoming calls at a particular time of day in the last month, this historical information cannot be considered as a basis for inferring the same action at any other moment in future without the PCE user's consent.

Premature end of computing procedure: There is always a pre-defined start-to-end procedure in traditional computing, but in pervasive computing activities may end prematurely. Our everyday life is full of examples of opting for one task and then, without completing it, switching to another is very common. One of the best examples are user clicks on the web. We very often end-up retrieving information form websites which we did not plan at the beginning of our retrieval session. A user of a PCE shall not coordinate himself/herself with PCE, it is the other way round.

User-centric computing: In traditional computing user is in the “background”. The general-purpose computing devices carry out computations with minimal, if at all any, user involvement. However, in pervasive computing environments, computing often does not happen without user consent. User preferences, requirements and requests have a role to play in PCE and they determine which services should be provided to them. No action can take place in a PCE without the user consent or authorisation. This may be in advance when users subscribe to the PCE and/or at the time the PCE is being used. A resident of a care home might have given in advance an authorisation to be remotely monitored while asleep, but he/she can change his/her mind at any moment.

Computing transparency: In traditional computing, applications are device-oriented, whereas in environments where users are empowered by the surrounding wireless and mobile computational devices, applications become user-oriented. Transparency of computations in traditional computing is a non-existent feature, but important in PCE because users have to interact with the PCE at all times. They are well aware of their interaction, and they may persist until the desired output is produced. Computing transparency involves automatic and seamless integration of computing devices, as its key feature, which is essential in PCE. For example, monitoring the presence and location of inhabitant of a homecare, their activity, who they are with at a particular moment can be handled in a PCE without the users being disturbed or even noticing it.

In Table 3.1 we summarise the expectations debated in this section and give their brief explanations.

	Expectation	Description
E1	Defining inputs dynamically	Inputs to computers to perform a task cannot be limited to a prescribed list of inputs given in advance; inputs should be defined dynamically at run-time.
E2	Computing output not guaranteed	In pervasive computing we may not have any output at all.
E3	Distribution of devices and computing	Centralising computing functionality, particularly when lots of computation takes place at any moment is not feasible, practical any demanded any more. Computing responsibility can be delegated to variety of devices, with a variable computational power and each of them can be designed for a specific task.
E4	Pervasiveness of user-specific devices	Computing devices are user-specific and each user may also use multiple devices. Identifying users from each other to deliver services appropriate for one and not the other is typical of pervasive computing.
E5	Flexible interfaces	There cannot be constraints on the type of interfaces used in pervasive computing. Devices and applications can use use numerous and constantly changeable devices (as they change).
E6	Implicit interaction	Interaction between human and computer may be historically seen as explicit, but in pervasive computing, where user is surrounded by computational devices, the interaction between may often be implicit.
E7	Redundancy of historical information	Pervasive computing is not responsible for historical information; they have to deal with situation they encounter in PCEs. Each situation in PCE might require different computations and services to be delivered.
E8	Premature end of procedure	In traditional computing there is always a pre-defined start-to-end procedure, but in pervasive computing activities may end prematurely.
E9	User-centric computing	Pervasive computing applications are user-oriented rather than device-oriented.
E10	Computing transparency	Seamless integration of devices with minimal user explicit interaction is a key feature of pervasive computing.

Table 3.1: Expectations from pervasive computing

3.2 Problems with Context-Awareness in Pervasive Computing

In chapter 2, section 2.3 we elaborated on the research relevant for contexts in computing. We highlighted the lack of common consensus on what exactly context may mean; we discussed the complexity and limitations of software applications which are developed with 'context in mind' and which consequently become '*context-aware*' and we also looked at the issue of '*context modelling'* in such applications. Al three of them have highlighted a range of limitations of context aware applications. In this section we elaborate on problems in context aware computing and give our further justification on why 'context awareness' has not delivered most of expectations E1-E10 form Table 3.1.

Imperfection of context awareness: (Henricksen et al. (2002) argue that because PCEs are highly dynamic, information describing them can quickly become out of date. According to them 'context histories (past and future)' will frequently form part of the context description and therefore applications in PCE are interested in more than the 'current state of the context'. We agree with (Henricksen et al., 2002) that information describing an situation of a PCE can quickly become out of date, but question their view that software applications in PCE are interested in more than the current state of the context. We believe that situation specificity is a key feature of context-awareness in PCE, which changes constantly and at any given time we may encounter a different situations in the same PCE. We would also like to think that one change in a PCE must trigger the existence of a new situation and the acquired information of the situation is always related to one moment in a PCE, which might be completely immaterial for the next one. From that perspective, situation might be much more than "the current state of the context", as Hendrickson claims, but not in terms of relying on historical information and "previous" contexts. The appropriateness of the information generated in and by PCE and their usage by software applications, will always depend on the situation triggered by a change in the PCE. Whenever there is a need for historical information in PCEs, we can deal with it at the application levels, i.e. applications can make the right decision in situations where another source of information is required, such as a piece of data retrieved from a database, what (Bellavista et al, 2012) refers to as 'virtual sensors'. We would like to emphasise that by looking at various situations in PCE we come closer to the idea of situation awareness, as defined and exploited in (Matheus et al., 2003)(Matheus, 2005)(Bahrami, et al., 2007)(Gauvin, et al., 2003). According to their contextualisation of information in pervasive computing, situations describe pervasiveness "better than context", which is close to our own idea of creating a PCE based on 'a situation' in it.

Localisation-aware systems dominate: Most of the available and fully developed context-aware applications are not actually context aware but are rather 'localization-aware' systems. Active Badge (Want et al 1992), ParcTab (Schilits et al., 1993), Cyberguide (Abowd et al 1997), the Intelligent Room (Coen, 1998), the

Bat (Harter, et al 1999), the Context Toolkit (Salber et al 1999b), (Garlan et al 2002), CASS (Fahy and Clarke 2004) are primarily concerned about the location context of the user in order to make a decision. Most of these localization-aware systems are systems that adapt themselves to only the new location of the user, so as to provide mobile functionality appropriate to the location. Although location-aware systems have gained popularity in recent years when using mobile devices, we believe information generated in context-aware PCEs is far more diverse and complicated, i.e. context awareness in PCEs is more than localization.

Contextual information is not always abstracted: Some proposed context-aware architectures, including the above examples, are dependent on the hardware infrastructure. Consequently in software applications generated form such architectures do not support context abstract at all, or at best to a limited extent. For example, the location information detected from sensors is not interpreted to produce a higher level of abstraction and physical sensors are also not hidden behind needed abstraction. As a consequence we created systems which were NOT scalable; i.e. capable to allow expansion. In other words the system was not able to accommodate any additional sensors and could not cope with the dynamic nature of PCEs. We believe that the way out is in abstracting contextual information through models which can be materialised with appropriate languages that allow reasoning upon detected contextual information and infer situational information.

Lack of context reasoning: Some proposed architectures, such as The Context Toolkit (Salber et al 1999b) do not have an easy-to-apply context reasoning mechanism to infer either "high context information" or situational information. Toolkit represents context using key-value model of context representation. The keys do not have meanings and therefore do not allow further than basic reasoning upon contexts. Hydrogen architecture (Hofer et al 2002) is also short of a reasoning mechanism. Although it uses an object oriented model to represent context, each sensor is being interpreted through a highly coupled adaptor and therefore its software architecture does not provide a good abstraction of context. In the CASS architecture (Fahy and Clarke, 2004) the contextual information which is limited to location is passed to a relational database. This means that the context processing takes place in a relational data model which cannot support any reasoning mechanism. It is also important to note that most of the available architectures for context-aware applications in PCE are specific to an application domain and, as Miraoui et al (2008) have concluded in their survey, they require additional effort for their adaptation to other domains. With regards to the healthcare domain, although many research papers can be found on pervasive healthcare systems, few of them focus on the context model and its structure.

The lack of high level abstraction: When the processing of context is done in a key-value model, relational model or object-oriented model, because the terms used do not have meaning, reasoning to infer higher-level context, if it exists at all, must be done through programming. This lack of high-level abstraction requires the context models to be highly coupled with the rest of the system and therefore the addition of more devices dynamically is not supported. We believe that due to the rich formalism available in ontologies for representing contextual information, we should use them for reasoning upon contex and to inferring situational information in PCEs. Service oriented context-aware middleware, SOCAM (Gu et al., 2004), (Wang et al., 2004), and context broker architecture, CoBrA (Chen et al., 2003a), (Chen et al., 2004a) are among the most cited examples in the literature that have used OWL ontology for processing the contextual information. However, these are not examples which can model a situation in PCE for two reasons. The first one is that their interpretation and manipulation of contextual data is focused on managing the sensor generated data, which is not always sufficient for interpreting situation in PCEs. There is an abundance of data in PCEs, which is NOT generated by sensors, but play an important role for defining the semantics of the situation in PCE. The second is that, all these solutions are knowledge based systems that manage enormous amount of data. This has been considered as cumbersome SE solutions which are very difficult to manage. Even if we assume that we can distribute such complex SE solutions across clouds, then it is questionable how could these distributed computations answer the our view that PCEs exist because they can accommodate constantly changing and seamlessly integrated devices of variable computational power.

Table 3.2. summarises the shortcomings of context awareness in addressing needs of PCEs.

▪	Limitation	Description
L1	outdated contextual Information	Contextual information in traditional context-aware applications can quickly become out of date.
L2	Domination of localisation-aware systems	context awareness in PCEs is more than localization.
L3	Lack of context abstraction	physical sensors are not hidden behind abstractions.
L4	Lack of context reasoning	Inappropriate context model is used that does not support context reasoning mechanism to infer high-level context and reason about complex situations.
L5	Insufficient context-information	Provision of insufficient situational information will result in delivery of unexpected services to the user.
L6	Information overloading	Unnecessary information overload caused by pre-defined contextual model and architecture that detects all contextual information would make processing the context to deliver expected service to the user very expensive.
L7	Context model inflexibility	Most of the available architectures for context-aware applications in PCE are specific to an application domain; they require additional effort for their adaptation to other domains.

Table 3.2: Limitations of context-aware applications

3.3 Summarising Shortcomings of Pervasive Computing

The analysis of current problems in Pervasive computing itemised in Table 3.1, and the limitations of well-known applications of context awareness listed in table 3.2, have polarised our research problem and send two important messages.

Firstly, however hard in the past we tried to create new SE solutions which respond to either technological changes or ever changing business demands, it is always difficult to predict that such software solutions will be a definitive answer to problems we experience in new computing environments. Building Context Awareness in computing is not a silver bullet for creating PCE, as highlighted in the previous section. It is difficult to imagine that we can exploit the 15 years of research in modelling and managing context, which mainly manipulates sensor generated data, when modelling and manipulating PCE and, at the same time, achieve expectations E1-E10 from Table 3.1.

Secondly, the list of expectations from table 3.1 is not ‘science fiction’. It is more a common perception of modern computational environments. We can claim that it is difficult to perceive modern computing without having *pervasiveness in mind*. Therefore if we advocate that our PCE exist because we can seamlessly integrate numerous heterogeneous devices into our everyday life, and compute with them at any time and place, then we will immediately stumble upon the traditional perception of contextualising PCE. Tables, 3.1 and 3.2 send clear messages:

I. Traditional context are not sufficient for defining and manipulating PCE and therefore we should replace it with situations in PCE, as found in Table 3.2 and in Objective 2 on page 9.
II. We should re-think or re-assess the way we perceive computing in PCE. This is obvious from Table 3.1 but also dictated by Objective 3 on page 10.

3.4 The Way Forward in This Research

In this section we pave the way towards a possible solution which can address I. and II. from section 3.3. We have to identify the way of meeting expectations E1-E10 from Table 3.1. and addressing limitations L1-L8 from Table 3.2. The outcome is a list of categorised characteristics of PCE itemised in Table 3.3, which should be used when addressing I. and II.

3.4.1 What would be Common Characteristics of PCE?

In this section we create a set of categorised PCE characteristics which are based on expectation we itemised in Table 3.1.

When addressing **E10 (Computing transparency in PCEs),** we primarily need a **seamless integration** of devices within a particular PCE**.** However, if we would like to think that PCEs are physical environments which are augmented with numerous heterogeneous devices with embedded computational power, then these devices should be interconnected with each other. They can be stationary or mobile, and there is no restriction on the number of them. However, the existence of their computational and communicational power is essential in order to have them seemingly integrated into PCE. Hence, we have characteristics P1 and P2 in Table

3.3. In addition if we assume that any number of devices can comprise a particular PCE then we would like to think that **no pre-defined device setting is needed.** The setting of heterogeneous computational devices in a PCE is not definite. That is, adding a new device, replacing a malfunctioning device with a working one, or removing a device without replacement is possible in a PCE at runtime with ease and without affecting any other part of the PCE. There is no pre-defined structure between devices of a PCE, and there is no operational dependency between devices. Otherwise devices would be highly coupled with each other and therefore their replacement or removing from the setting would be difficult if not impossible. Thus characteristics P3 in Table 3.3.

When addressing **E9 (User-centric computing)** we have to create computational environment which **serves the user**. We would like to think that the PCE empowers its users at any time/place, favours unobtrusiveness and respects/adopts user preferences as they interact within PCE. Consequently, the devices integrated into a PCE penetrate into our private and professional lives and enable us to perform our daily tasks comfortably. Thus P8, P9, P11, P12 in Table 3.3.

When addressing **E6 (Implicit interaction)** we would like to think that, apart from securing as minimal intrusion for the user as possible, the PCEs guarantees implicit and not solely explicit interactions between users and PCE. This may mean that users may use and be a part of a PCE without "attending" to any device. Devices are transparently integrated into a PCE in order to secure the delivery of expected services and the most important outcome is that these services should exactly match users expectations and preferences, without assuming that the interaction between users and PCE is explicit. The user in a PCE can focus on what they want to do, not how to do it. Thus characteristics P12, P13, P14 in Table 3.3.

When addressing **E7 (Redundancy of historical information),** we need to be concerned only about the *situation specific information* within a particular PCE. We would like to think that the availability of an extensive amount of information, which might have been cumulated within the PCE will overload the user and is not in itself a virtue. This may mean that the power of PCE is in its ability to handle a particular "moment" or a situation, which may not depend on historic information at all. What matters is to have all the relevant information about the situation that makes a difference to the user. Consequently, we would like to think that a situation in a PCE is based on information, which is important for that "moment" and may be completely immaterial for the next one. If any monitoring, tracking, or historical information is indeed needed in PCEs, it may be managed by other means, e.g. through a separate or additional software applications which run in PCE. Finally, we would like to think that a PCE will make use of the information provided by its seamlessly integrated devices (also referred to as 'information appliances' (Norman 1999)) in a particular situation and make a decision on appropriate service to the user for that situation. Thus characteristics P14, P16, P19, P20, P21 in Table 3.3.

When addressing **E1 and E8** we need proactivity in PCE whilst not being autonomous. We would like to think that PCEs in spite of being user-centric environments, should also be "proactive systems", i.e. they may take actions without being instructed. Although users' serenity and composure necessitates their minimal distraction in PCEs, which is also supported by PCEs unobtrusiveness, we would like to believe that it is not very wise to condone a completely autonomous PCE where users' changing preferences and dynamic input into a PCE do not play any role. We can also add that, as a consequence, we might witness a premature end of computing procedures in PCE because it would be difficult and inappropriate to predict exactly what a particular situation in PCE would be and which decisions are to be taken in that PCE. Thus characteristics P3, P5, P6, P7, P17 in Table 3.3.

However, we would like to think that PCE accommodates intelligence which enable us to address **E1, and E6-E10**. We do need provision of some reasoning mechanism that can infer new knowledge (we may call it higher level, i.e. situational information) from the collected contextual information (we may call it "low level"). Furthermore, supporting execution of reasoning rules in situations where several pieces of situational information are related to each other and where the relationships cannot be represented in any abstract PCE model, will result in addition of a set of reasoning rules to the semantic of PCEs. Thus characteristics P8-P17 and P19-22 in Table 3.3.

Finally, all PCEs are **domain specific**. The setting of devices and information in a PCE is determined by the domain. Conceptualisation of a PCE without a domain is not possible and we would like to think that it is not practically possible to have a generic PCE applicable to all domains. The pre-defined situations are difficult to enumerate and not advisable. Therefore, there cannot be any predefined PCE "model" applicable across domains. Thus characteristics P18 in Table 3.3.

In Table 3.3 we summarise the characteristics of PCE which is done according to the analysis above and in sections 3.1. We have 4 groups of PCE characteristics which focus on different aspects of PCEs. We can look at

- PCEs from the perspective and nature of devices which are seamlessly integrated into them and consequently create various computational / communication settings;
- PCEs from the user perspective because users are a major driving force behind the creation of any particular PCE.
- the performance of PCEs in terms of the nature of software solution which accompany PCE and focus on the fact that situations in PCE are to be modelled and computed in order to justify their existence: the delivery of expectations/services to their users.

Each line in Table 3.3. is labelled as a particular "P" which should be taken into account if we wish to address the shortcomings of pervasive computing as itemised in I and II from section 3.3. We highlight that most of the "Ps" in Table 3.3 are consequences of EXPECTATIONS we have in PCE, i.e. all L1-L10 which appeared in Table 3.1. However, the issue of situation awareness and capability of reasoning in computations within PCE are consequences of limitations of the current context-aware solutions in pervasive computing, as itemised in Table 3.2. If we wish to create a SE solution which deals with P1-P18, we will implicitly address the issue of creating and manipulating situational information. In other words, focusing on devices and users in and performance of PCE would implicitly require the existence of situational information.

	PCE/Device/User/Situation	Characteristic
P1	PCE (and devices)	is a physical environment with seamlessly integrated devices.
P2		contains heterogeneous networked devices with various computational and communicational power.
P3		is scalable to allow extension of devices dynamically at runtime.
P4	PCE (and computational/ communication setting)	are stationary or mobile.
P5		have no definite setting.
P6		have no pre-defined structure between them.
P7		have no operational dependency between them.
P8	PCE (and its users)	is to empower the user.
P9		is to empower the user anywhere anytime.
P10		is to avoid information overloading.
P11		is user-centric.
P12		is non-autonomous, adopts user desire and preferences.
P13	PCE (and its performance)	is unobtrusive.
P14		has minimal distraction to the user.
P15		has no tracking of user behaviour.
P16		has no interest of historical information.
P17		is proactive.
P18		is domain specific.
P19	PCE (and its situation)	is situation-aware.
P20		is situation specific.
P21		is capable of inferring situational information.
P22		is capable of reasoning upon situations.

Table 3.3: Summary of PCE characteristics

It is also important to note, that not all "P" from Table 3.3. are essential in the creation of PCE. Therefore it remains to be discovered how to approach the implementation of these requirements itemised in P1-P22 and how to find out which one of them is more important than other. Obviously, P1-P7 are solely related to the existence, purpose and capabilities of various heterogeneous devices which surround us, but P8-P22 are more focused on software solutions which we must develop in order to materialise PCE. They are more focused on the semantics which will be stored in PCEs and manipulated though computations in order to deliver expected services to users involved in the PCE.

3.4.2 What Would be the Role of situations in Pervasive Computing?

PCE, based on our discussions in the previous sections, can be seen as a physical environment that contains a collection of heterogeneous computational devices, which are seamlessly networked and integrated. The collection or setting of devices in a PCE, which can be stationary or mobile, can change even at runtime, owing to the fact that there is no pre-defined structure and operational dependency between devices. Systems designed with PCE in mind are user-centric and they observe user preferences; PCE without a user is inconceivable. The purpose of PCE, as a non-autonomous environment, is to empower users, without overloading them with information, at anytime and anywhere. PCE ARE proactive and, at the same time, unobtrusive, with minimal distraction to their users. They are also *domain* and *situation specific*. This means that they are *situation-aware* and capable of *inferring situational information* from low-level contextual data and able to reason about situations Thus P19, 16, 21 and 22 in Table 3.3.

3.4.3 What Do We Compute in PCEs and Why?

If we wish to address the shortcomings of pervasive computing as underlined in section 3.3 and pave the way towards creating PCEs which can fit all situations we may encounter within them, then we need primarily a SE solution, with its computations, which can generate both: a definition of the semantics in a particular situation in a PCE and the delivery of services within it. The way forward would be to create the formal computational model, which should be materialised through a SE solution, which

A. uses the computations generated from the formal model
B. is built from a software architectural model which accommodates the computations defined in the formal model and
C. secures the deployment of software architectural elements, by using the available technologies.

Without A.-C above we cannot claim that we offer a SE solution which creates a PCE. The formalised computational model per se is no guarantee that we will be able to deliver results. We need a software architecture which accommodates proposed computations created form the formal model, which in turn guarantees that its components will be deployable with available software technologies.

The background research in section 2.4 has highlighted that the use of SWTs is the way forward, if we wish to create a new era of SE solutions based on the semantic and understanding of our computational environments across domains. The SWT stack (W3C 2004b) has been created with semantic “Web” in mind, but the same philosophy, ideas and languages in particular can be re-used outside the semantic Web domain. If we can transfer the interpretation of and reasoning upon the content of web and its URLs to any computational environment, then we can achieve almost identical result as on the Web. In our particular domain of *creating situations in PCEs and reasoning upon them in order to deliver services*, we need the same mechanism: describe the domain (situation within a PCE) using SWT languages, and reason upon it using SWRL in order to create a computational result ("deliver a service").

The common characteristics of PCEs from Table 3.3 might be sufficient to manage the description of semantics in situations we may encounter in PCEs. However, without having a formal model which will define exactly

1) which elements we must have, and
2) which computational steps we must perform in order to secure the computations which deliver a service,

we may not claim that the we have created a PCE with characteristics itemised in Table 3.3. This type of formalisation might have a double purpose:

a) it may systemise our perception on what PCEs are and what we expect from them across wider research community, and
b) it may guarantee that, if we follow the formalised model, we will be able to define and create a situation in a PCE which will deliver an expected service.

Furthermore, by knowing that we will use the SW technology stack, the formal model should be expressed in *vocabulary* and following the *terms* of the suitable technology, which can secure the implementations of formalised computations through the deployment of software architectural components. Both of them, *vocabulary* and *terms*, must influence and determine the format and the content of the formal computational model, which should not be confused with numerous formalisations available in (Baader, et al., 2011), (Krdzavac and Gasevic, 2010), (Klinov and Parsia, 2008), (Pan et al., 2006).

It is expected that our formalised model should give a finite set of definitions and axioms which will help us to formalise the knowledge on PCE. We also need a structure within the model which can create and manage formalised knowledge. From that respect OWL as a language and OWL ontologies in general, as a formalisation of the knowledge we wish to define and manipulate, seems to be a perfect choice. At the same time we have to think how to use the formalised model for computing implicit consequences or knowledge the model describes or creates. Implicit consequences are beneficial in SE because they can successfully be implemented and can support any automation we may create through our computations, which is very important in PCEs.

Furthermore, our formal model should create cheap computations, i.e. our solutions should be computationally cheap. This means that building any knowledge base and excessive persistence, which could underpin the delivery of services in PCEs, is out of question. In the era of highly accessible mobile and wireless computing, we should assume that all our implementations should run on mobile devices. Therefore, we should be very cautious with the possibility of hosting our SE solution in various clouds and using application hosts as an answer to the deployment of our computations.

3.6 Summary

This section itemises expectations we have in pervasive computing environments and analyses limitations encountered in software solutions focused on context awareness which attempted to address the complexity of pervasive computing. Therefore we pushed forward the issue of situation in PCEs and created a common set of PCE characteristics which could help us to achieve one of the objectives of our research and pave the way for the creation of the formal computational model which can create a PCE and which can be deployable using SE principles and modern software technologies.

Chapter 4
A Formalised Computational Model for PCEs

In this chapter, we propose a formal computational model (FCM) for delivering a situation-specific service in PCEs.

The formal computational model (FCM) secures the delivery of a situation-specific service(s) in a particular situation PCE_{Δ} by

1) creating a situation-specific taxonomical structure $PCE_{\Delta}T$ for the PCE_{Δ}; and
2) reasoning upon the $PCE_{\Delta}T$ taxonomical elements in order to deliver a situation-specific service for the PCE_{Δ}.

Consequently, the purpose of computations in FCM is to secure

a) the existence of a *generic taxonomical structure $PCE_{\Delta}T$*, which can fit any situation PCE_{Δ} found in PCE. However, each PCE is always domain-specific (P18, Table 3.3) and when dealing with a situation PCE_{Δ} in PCE, we have to have domain and situation-specific taxonomical elements, as noted in 1) above. This implies that the FCM should be able to create the exact $PCE_{\Delta}T$ for each detected situation, i.e. to create a *situation-specific taxonomical structure*, which may be *extended* from the generic taxonomical structure $PCE_{\Delta}T$.
b) the delivery of a situation-specific service for PCE_{Δ}, based on the semantics found in the (possibly) *extended generic taxonomical structure $PCE_{\Delta}T$*. This implies that in each situation PCE_{Δ}, the FCM must be able to manipulate the semantics of the situation PCE_{Δ}, through reasoning upon $PCE_{\Delta}T$ elements.

Obviously the result of such reasoning is always a delivery of a situation-specific service(s).

If the FCM has to give us a formal computation for a) and b) above, then we must define it in either pseudo code or in a graphical format. In both cases each part of FCM should be self-explanatory and each term of the FCM should be defined through either definitions or axioms. Therefore in section 4.1. we give a set of definitions and axioms which are essential for understanding the FCM. They are summarised into a *generic taxonomical structure $PCE_{\Delta}T$*, which can be extended by the FCM into a *situation-specific taxonomical structure.* In section 4.2. we define steps which secure the creation of the generic and extended (i.e. situation-specific) elements of the $PCE_{\Delta}T$ in order to deliver a situation-specific service.

It is important to note that the creation of a *situation-specific taxonomical structure* is a powerful mechanism for delivering a correct service(s). The power of the proposed FCM is creating a semantically rich taxonomical structure in the first place, without which we cannot secure the delivery of the service, because it is based on the semantics found in the taxonomical structure.

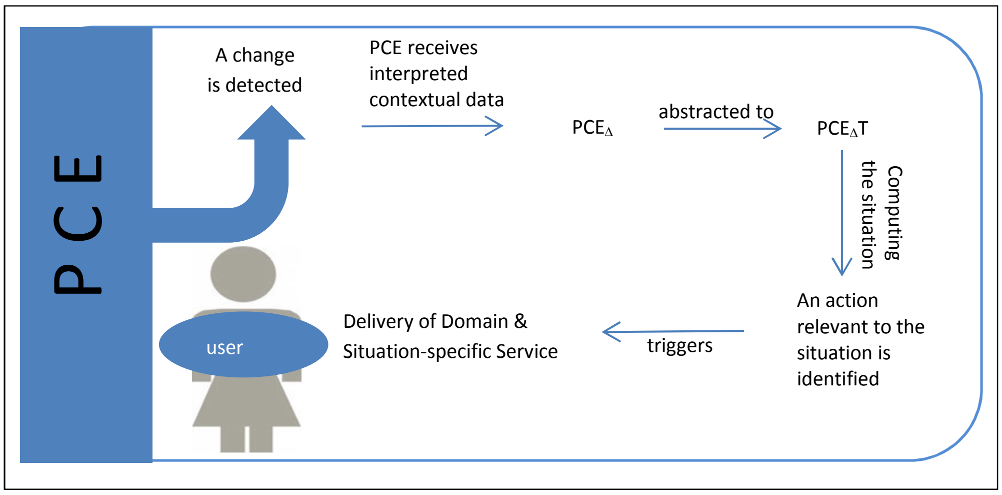

PCE
A change is detected
PCE receives interpreted contextual data
PCE∆
abstracted to
PCE∆T
Computing the situation
An action relevant to the situation is identified
triggers
Delivery of Domain & Situation-specific Service
user

Figure 4.1: FCM computation of a situation-specific service in a PCE

The reader should pay attention to the fact that the FCM cannot fully specify the exact computation of the delivered service(s), because services are domain and situation-specific. However, the FCM ensures that the taxonomical structure $PCE_{\Delta}T$

is sufficiently rich and may trigger reasoning for the delivered service(s) automatically. Figure 4.1 shows that once a change in a PCE is detected, The PCE receives interpreted contextual data to create a situation PCE_{Δ}. Based on the situation PCE_{Δ}, the FCM will create the taxonomical structure $PCE_{\Delta}T$. The FCM will also reason upon the elements of the $PCE_{\Delta}T$ to trigger a domain and situation-specific service(s) to be delivered to the user of the PCE.

4.1 Taxonomical Structure of PCE

Terms used in FCM, irrespective of whether it is defined in pseudo code or in a graphical format, must be defined in advance for the FCM to be self-explanatory.

Therefore, we give a set of definitions and axioms which are essential for understanding the FCM. These definitions and axioms are summarised into a generic taxonomical structure $PCE_{\Delta}T$, which can be extended by the FCM into a domain and situation-specific $PCE_{\Delta}T$.

4.1.1 Definitions of PCE, Situations and Delivered Services in PCE

PCE is a cyber-physical environment in which there are various devices with computation and communication power, and people who wish to be empowered by the advanced technologies surrounding them. Given that PCEs are cyber-physical, they are naturally occupied with physical and tangible objects that do not necessarily bear any resemblance to a device. These objects could be tagged and equipped with appropriate microchips and sensor pads to act like a device, lending themselves to a more diverse and expanded PCE. There are also intangible objects present within PCEs. Software programs (computing entities) which support PCEs, including those that integrate various devices into a PCE, or software applications which generate and manipulate data created within the environment are examples of such cyber objects. Each of these cyber and physical objects that are seamlessly interconnected through a wireless network and allow pervasiveness of computing and communication of data at anytime and anywhere, have a purpose and role to play in various situations users may encounter within a PCE. Their main role is to deliver services to a user of a particular PCE. This means that in each situation in PCEs we know exactly which cyber physical objects are interconnected with the user, and what the user expects from the PCE at that moment. User's expectations are very often associated with services which should be delivered by a particular PCE in a particular situation. Based on our description of PCE in section 3.4.1, and P1, P2, P9, P11 in Table 3.3 section 3.4.1,

> **Definition 1:** PCEs are cyber-physical environments that allow pervasiveness of computing and communication of data at anytime and anywhere to support users who are in charge of such environments.

If we say that in each situation within a PCE, we may have any number of cyber-physical objects, interconnected with the user, who is in charge of that particular PCE and expects services from it, then we should assume that in each situation within a PCE we have only one user. Therefore, In each situation in a PCE we should know

1) who the user is,
2) which services the user expects from the PCE and
3) which cyber-physical objects are interlinked in that PCE to deliver the services. In other words we should always know exactly what a particular PCE "contains" in a particular situation and how it supports the user of the particular PCE.

In order to address 1)-3) above we introduce a situation Δ in PCE as "a particular moment in real world". Each situation Δ within a specific domain triggers the existence of a particular PCE which we name PCE_{Δ}. In each PCE_{Δ} we consequently know exactly which cyber-physical objects exist, who is in charge of it (who is "the user") and which services will be delivered, i.e. expected by the user. Therefore, cyber-physical objects and the user are essential participants in PCE_{Δ} and the situation Δ consequently creates "user-centric" PCE_{Δ}; i.e. user decides on participation in the PCE_{Δ} and on the number and type of cyber-physical objects within PCE_{Δ} from the domain-specific PCE (P3, P11, P20, and P18 of Table 3.3 in Chapter 3).

> **Definition 2:** PCE_{Δ} is a particular situation Δ in a PCE, i.e. it is a specific PCE where any number of cyber and physical objects coexists for the purpose of delivering a domain-specific service to the user, who is in charge of the PCE_{Δ} .

For each situation Δ, we will have one and only one user of the PCE_Δ, who will be surrounded by the chosen cyber and physical objects, and will expect services from the PCE. These "expected services" from the PCE are often associated with functionalities defined within it. They are always domain-specific and situation-specific; therefore we can say that in each PCE_Δ we may have various domain specific services delivered to the user of the PCE_Δ. However, delivering of services in PCEs is always based on the situation. We have to decide how a situation creates a particular PCE_Δ ("what are its participants?") and how to secure the delivery of appropriate services in the PCE_Δ ("what does user expect from the PCE_Δ?"). Based also on P9, P17, and P18 of Table 3.3 section 3.4.1,

> **Definition 3:** A domain and situation-specific service for the user of a PCE_Δ is a functionality triggered by computations in the PCE in order to provide timely and appropriate assistance to the user.

Finally, recognising situations Δ in real world, and PCE_Δ created by them, depends on data and information we must have on PCE and main participants of PCE_Δ. If we really want to know which choices of cyber and physical objects we may have for the creation of a particular PCE_Δ and which user is in charge of it, we should be able to "detect" them from within the PCE. In other words their availability for and presence in a particular PCE_Δ should be detected, or "known" if we wish to deliver services to the user (P2, P3, P11, and P19 in Table 3.3). The "detection" itself may have various forms and may result in various data being available in the PCE_Δ. We may be able to detect PCE_Δ participants by

(i) using sensors and sensor generated data;

(ii) exploiting users' inputs, which may often carry information on users' preferences and intentions in a particular PCE_Δ;

(iii) retrieving any persistent data which might be available in semantically rich PCE environments. It is important to note that the richer "detection", the more successful delivery of services in a PCE_Δ will be.

Therefore any combination of (i)-(iii) is assumed when creating a PCE_Δ.

4.1.2 Instance $\mathtt{ins_t}$ and Category $\mathtt{Ctg_i}$ as Abstractions in $PCE_\Delta T$

Each PCE_Δ comprises a finite number of real world instances. The obvious examples are the PCE_Δ's user or the cyber and physical objects involved in the PCE_Δ. Where real world instances share the same features, they are grouped together under an encompassing abstract name. The most important rationale behind grouping real world instances is to facilitate the delivery of domain-specific services (Definition 3). If we know that real world instances, $\mathtt{ins_t}$s, share semantics, they might be grouped together to enhance the delivery of domain specific services to different users in different PCE_Δs. We group real world instances into categories $\mathtt{Ctg_i}$ where $\mathtt{i} \in \mathbb{N}$. In other words, group of real world instances which are abstracted into a category Ctg_i are actually its instances, $\mathtt{ins_t}$s, participating in a PCE_Δ.

Definition 4: An instance ins_t is a real world participant in a PCE_Δ. The set of all instances of a PCE_Δ is $INS = \{ins_t \mid t = 1 .. n, n \in \mathbb{N}\}$ where n denotes the number of instances participating in a particular PCE_Δ.

Definition 5: A category Ctg_i is an abstraction of a subset of INS. Ctg_i are classified into CTG, representing all abstractions of subsets of INS in a particular PCE_Δ; i.e. the set of all categories in a particular PCE_Δ is $CTG = \{Ctg_i \mid i = 1 .. n, n \in \mathbb{N}\}$ where n denotes the number of subsets (categories) of the set of all real world instances of the PCE_Δ.

4.1.3 Levels λ of Categories $\mathtt{Ctg_i}$ in PCE_Δ

If we see $\mathtt{Ctg_i}$s as abstractions of real world instances ins_t then we have to bear in mind that the higher the level of abstraction, the less the precision of the semantics of its instances $\mathtt{ins_t}$s. When more precision is required to represent the semantics of some real world instances, lower levels of abstractions are needed. Categories $\{\mathtt{Ctg_i}, \mathtt{I}\ \mathtt{i} \in \mathbb{N}\}$ should therefore, have subsets which allow for levels of abstraction. These subsets should also allow richer interpretation of semantics of instances $\mathtt{ins_t}$ which make $\{\mathtt{Ctg_i}, \mathtt{I}\ \mathtt{i} \in \mathbb{N}\}$; i.e. an instance $\mathtt{ins_t}$ being represented by a lower level abstract category is semantically richer than by a higher level abstract category.

When an ins_t of a $\mathtt{Ctg_i}$ is detected in a PCE_Δ,

1) all subset categories of the $\mathtt{Ctg_i}$ are also detected;

2) $\mathtt{ins_t}$ may belong to any subset of its $\mathtt{Ctg_i}$, depending on the situation Δ and the precision of abstraction required.

We must know exactly which subset of category $\mathtt{Ctgi}$ the $\mathtt{ins_t}$ belongs to. To differentiate between different subsets of a $\mathtt{Ctg_i}$, each of them is levelled. For each $\mathtt{Ctg_i}$ there are a finite number of ordered levels, λ

$\lambda = (\mathtt{Lev_1}, \mathtt{Lev_2}, \ldots \mathtt{Lev_j}, \ldots \mathtt{Lev_{m-1}}, \mathtt{Lev_m}), 1 \leqq j \leqq m$

which determines, to which subset of $\mathtt{Ctg_i}$ real world instance $\mathtt{ins_t}$ belongs. Each level will encompass its lower levels as depicted in Figure 4.2.

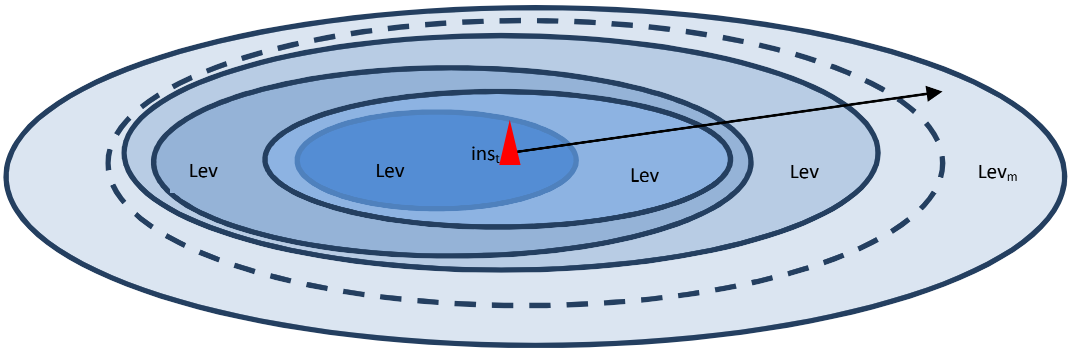


Figure 4.2: An $\mathtt{ins_t}$ being a "Subset of" a number of $\mathtt{Ctg_i.\ Lev_j}$

λ is an order set with a binary relation "a subset of"between each two elements. This indicates the order among any two members (two subsets of the $\mathtt{Ctg_i}$) of λ such that for all $1<j<m$, $\mathtt{Lev_1}$ is "a subset of" $\mathtt{Lev_j}$ and $\mathtt{Lev_j}$ is "a subset of" $\mathtt{Lev_m}$ as shown in Figure 4.3.

Definition 6: λ is an order set of subsets of a Ctg_i of m levels where "a subset of" relation exists between each two subsets.

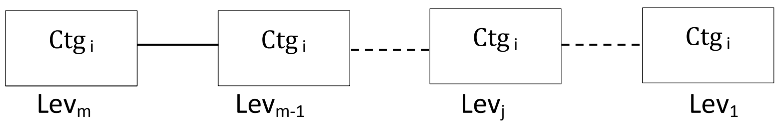


Figure 4.3: Each "subsets of" a $\mathtt{Ctg_i.\ Lev_j}$ is qualified with a level λ

The association of the qualifying factor of each subset of a $\mathtt{Ctg_i}$ with a level means that each $\mathtt{Ctg_i}$ and its subsets will have to be denoted with two values. One of these values is for $\mathtt{Ctg_i}$ and the other for its $\mathtt{Lev_j}$. Therefore, all categories in

PCE_Δ are shown as category level $\mathtt{Ctg_i.Lev_j}$ where i, j ∈ ℕ, if we wish them to accommodate instances $\mathtt{ins_t}$. The set of all $\mathtt{Ctg_i.Lev_j}$ is shown as $\mathtt{CTG.}\lambda$.

Axiom 1: $\forall$ x ∈ CTG , $\exists$ $Lev_j \in \lambda$ where x. Lev_j ∈ CTG.λ

This implies that an ins_t is always known by the $\mathtt{Ctg_i.Lev_j}$ that it is a member of. This membership is called `Mbr`.

Definition 7: A category membership Mbr is the membership of an instance ins_t in a category level $Ctg_i.Lev_j$ and denoted as Mbr (ins_t, $Ctg_i.Lev_j$) where i, t, j = 1 .. n, n ∈ ℕ.

Definition 7 lends itself to a new inevitability. Combination of detected $\mathtt{ins_t}$s, their $\mathtt{Ctg_i.Lev_j}$ and $\mathtt{Mbr(ins_t, Ctg_i.Lev_j)}$ in a PCE_Δ necessitates a need for a taxonomical structure which can hold $\mathtt{ins_t}$s and $\mathtt{Ctg_i.Lev_j}$s with their Mbr together. This taxonomical structure is called $PCE_\Delta T$.

4.1.4 Taxonomical Structure $PCE_\Delta T$ with Leaf and Root Categories

Definition 8: $PCE_\Delta T$ is the taxonomical structure of the real world participants in PCE_Δ described through INS, CTG.λ and Mbr (ins_t, $Ctg_i.Lev_j$).

$$PCE_\Delta T = \{Mbr (x, y) \mid x \in INS, y \in CTG.\lambda\}$$

When abstracting real world instances, the most specialised $\mathtt{Ctg_i.Lev_j}$ of a real world ins_t is where j = 1; that is $\mathtt{Mbr\ (ins_t,\ Ctg_i.Lev_1)}$. This special category, $\mathtt{Ctg_i.Lev_1}$, is called leaf category and is shown as $\mathtt{LCtg_i.Lev_j}$. On the other hand, the most generic $\mathtt{Ctg_i.Lev_j}$ that the real world $\mathtt{ins_t}$ has Mbr is where j = m, the maximum level in λ. This special category, $\mathtt{Ctg_i.Lev_m}$, is called root category and is shown as $\mathtt{RCtg_i.Lev_j}$.

Definition 9: A leaf category $LCtg_i.Lev_j$ ∈ CTG.λ is a category where $Lev_j \in \lambda$ and it is the infimum ($Lev_j \equiv Lev_1$).

Consequently, Axiom 2 says that for each instance $\mathtt{ins_t} \in \mathtt{INS}$, there will always be one and only one leaf category where $\mathtt{Mbr(ins_t,\ LCtg_i.Lev_j)}$.

Axiom 2: ∀x ∈ INS , ∃! Mbr(x, y) ∈ { $PCE_\Delta T$ | x ∈ INS, y is $LCtg_i.Lev_j$ }
For each ins_t there is always one and only one element of taxonomical structure that is a leaf category where Mbr(ins_t, $LCtg_i.Lev_j$) holds.

Definition 10: A root category $RCtg_i.Lev_j$ ∈ CTG.λ is a category where $Lev_j \in \lambda$, and it is the supremum($Lev_j \equiv Lev_m$).

Consequently, Axiom 3 says that for each instance ins_t there is always one and only one element of taxonomical structure which is a root category where $\mathtt{Mbr(ins_t, RCtg_i.Lev_j)}$ holds.

Axiom 3: $\forall x \in INS, \exists! \, Mbr(x, y) \in \{ PCE_\Delta T \mid x \in INS, y \text{ is } RCtg_i.Lev_j \}$

Considering Definition 6 and Axiom 1, we can deduce Axiom 4 which states that for every $\mathtt{ins_t \in INS}$ we will have $\mathtt{Mbr\ (ins_t,\ Ctg_i.Lev_j)}$ for all available values of j.

Axiom 4: $\forall$ x where Mbr(x, $LCtg_i.Lev_j$) holds $\exists \sum_{j=1}^{m}$ Mbr (x, $Ctg_i.Lev_j$)

4.1.5 Occurrence of Root categories $\mathbf{RCtg_i.Lev_j}$, in $\mathbf{PCE_\Delta T}$

The root $\mathtt{RCtg_i.Lev_j}$ is a very important element of $\mathtt{CTG.\lambda}$ because we know that

- $\mathtt{RCtg_i.Lev_j}$ is not a subset of another category (Definition 10);
- $\mathtt{RCtg_i.Lev_j}$ embrace all ins_ts of $\mathtt{Ctg_i.Lev_j}$ (Axiom 4);
- each $\mathtt{ins_t}$ is inevitably a member of a $\mathtt{RCtg_i.Lev_j}$ (Axiom 3).

Therefore we need to illustrate the power of $\mathtt{RCtg_i.Lev_j}$ by defining its occurrences. They will also allow us to

1) illustrate abstraction of ins_t into different $\mathtt{Ctg_i}$; and
2) axiomatise different $\mathtt{Ctg_i}$

There are many occurrences of $\mathtt{RCtg_i.Lev_j}$ in a PCE. We denote "occurrence" with the symbol "≡". First example of $\mathtt{RCtg_i.Lev_j}$ occurrence is an abstraction of users in PCE_Δ (see Definitions 1-3). There is always a user who is in charge of the PCE_Δ. However, users can have distinguishing roles, and therefore should have different $\mathtt{ins_t}$s. Whatever the role of the user is, he or she is a person and in that the $\mathtt{RCtg_i.Lev_j}$ of all $\mathtt{ins_t}$ of users in a PCE_Δ is named $\mathtt{Psn}$ for Person.

Definition 11: Psn is an occurrence of $RCtg_i.Lev_j$ encompassing all possible abstractions of real world instances of users, ins_ts, in any PCE_Δ.

For each situation $PCE_\Delta T$ there exists one and only one occurrence $\mathtt{Psn}$ of $\mathtt{RCtg_i.Lev_j}$.

Axiom 5: $\forall$ $PCE_{\Delta}T$, $\exists!$ Psn, where Psn $\equiv$ $RCtg_i.Lev_j$

For each situation $PCE_{\Delta}T$ there exists one and only one instance $ins_t \in INS$ where $Mbr(ins_t, Psn)$ holds.

Axiom 6: $\forall$ $PCE_{\Delta}T$, $\exists!$ $Mbr(x, y) \in \{ PCE_{\Delta}T \mid x \in INS, y \text{ is } Psn\}$
For all PCE_{Δ} there exists one and only one ins_t of Psn

The second example of $RCtg_i.Lev_j$ occurrence is an abstraction of cyber and physical objects in the PCE_{Δ}. We name such occurrence Ojt for Object.

Definition 12: Ojt is an occurrence of $RCtg_i.Lev_j$ encompassing all possible abstractions of real world instances, ins_ts, of cyber and physical objects in any PCE_{Δ}.

For each situation, PCE_{Δ} there exists one and only one occurrence Ojt of $RCtg_i.Lev_j$.

Axiom 7: $\forall$ $PCE_{\Delta}T$, $\exists!$ Ojt, where Ojt $\equiv$ $RCtg_i.Lev_j$

For each situation PCE_{Δ} there exists at least one instance $ins_t \in INS$, where $Mbr(ins_t, Ojt)$.

Axiom 8: $\forall$ $PCE_{\Delta}T$, $\exists$ x, where $Mbr(x, y) \in \{PCE_{\Delta}T \mid x \in INS, y \text{ is } Ojt\}$

The Third example of $RCtg_i.Lev_j$ occurrence is an abstraction of all domain-specific information. Every service offered by a PCE_{Δ} to the user is related to information specific to the domain in which the PCE_{Δ} has occurred. We named such occurrence Fld for Field.

Definition 13: Fld is an occurrence $RCtg_i.Lev_j$ encompassing all possible abstractions of real world instances, ins_ts, of domain-specific information in any PCE_{Δ}.

Axiom 9: $\forall$ $PCE_{\Delta}T$, $\exists!$ Fld, where Fld $\equiv$ $RCtg_i.Lev_j$

For each situation PCE_{Δ} there exists at least one instance $ins_t \in INS$, where $Mbr(ins_t, Fld)$.

Axiom 10: $\forall$ $PCE_{\Delta}T$, $\exists$ x, where $Mbr(x, y) \in \{ PCE_{\Delta}T \mid x \in INS, y \text{ is } Fld\}$

Therefore, according to Axiom 6, 8, and 10 it is reasonably justifiable to conclude that it is not conceivable at all to have a $PCE_{\Delta}T$ without three specific $Mbr(ins_t, RCtg_i.Lev_j)$ where $RCtg_i.Lev_j$ occurrence for the first instance is Psn, for the second instance is Ojt, and for the third instance is Fld.

More often than not the set INS of a PCE_{Δ} has ins_ts of other occurrence of $RCtg_i.Lev_j$ than Psn, Ojt and Fld (P12 of Table 3.3 Chapter 3). Therefore the next example of the $RCtg_i.Lev_j$ occurrences might be

(i) preferences of the user of Psn,

(ii) his/her location within PCE_{Δ},

(iii) the position of the ins_ts of Ojt involved in a PCE_{Δ}.

(i)-(iii) may also have an important part in delivering a particular service in the PCE_{Δ}.

Therefore, the fourth example of $RCtg_i.Lev_j$ occurrence is an abstraction of all user preferences as in i) above. We name such occurrence Pfc for Preference. Inclusion of preferences means that the software system which supports the user in a PCE_{Δ} is less intrusive and more personalised.

Definition 14: Pfc is an occurrence of $RCtg_i.Lev_j$ encompassing all possible abstractions of real world instances, ins_ts, of preferences of users in any PCE_{Δ}.

For each situation in PCE_{Δ} there may exist one and only one occurrence Pfc of $RCtg_i.Lev_j$.

Axiom 11: $\forall$ $PCE_{\Delta}T$, may $\exists!$ Pfc, where Pfc $\equiv$ $RCtg_i.Lev_j$

For each situation PCE_{Δ} there may exist an instance $ins_t \in INS$, where $Mbr(ins_t, Pfc)$.

Axiom 12: $\forall$ $PCE_{\Delta}T$, may $\exists$ x, where $Mbr(x, y) \in \{PCE_{\Delta}T \mid x \in INS, y \text{ is } Pfc\}$

Finally, the fifth example of $RCtg_i.Lev_j$ occurrence is an abstraction of all locations as in (ii) and (iii) above. We name such occurrence Lcn for Location.

Definition 15: Lcn is an occurrence of $RCtg_i.Lev_j$ encompassing all possible abstractions of real world instances, ins_ts, of physical or cyber locations in any PCE_{Δ}.

For each situation in PCE_{Δ} there may exist one and only one occurrence Lcn of $RCtg_i.Lev_j$.

Axiom 13: $\forall$ PCE$_\Delta$, may $\exists!$ Lcn, where Lcn $\equiv$ RCtg$_i$.Lev$_j$

For each situation PCE$_\Delta$ there may exist at least one instance $\texttt{ins}_t \in \texttt{INS}$, where $\texttt{Mbr}$ ($\texttt{ins}_t$, $\texttt{Lcn}$).

Axiom 14: $\forall$ PCE$_\Delta$T, may $\exists$ x, where Mbr(x, y) $\in$ {PCE$_\Delta$T| x $\in$ INS, y is Lcn}

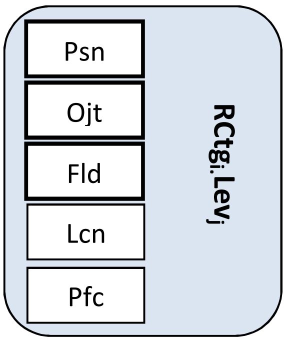


Figure 4.4: $\texttt{RCtg}_\texttt{i}\texttt{.Lev}_j$ of a PCE

All possible occurences of $\texttt{RCtg}_\texttt{i}\texttt{.Lev}_j$, outlined in 1) above, are shown in Figure 4.4 whereas axiomatisation of different $\texttt{Ctg}_\texttt{i}$s, outlined in 2) above, are shown in Axioms 5-14. Boxes with stronger border lines depict their essential presence in any PCE$_\Delta$T.

4.1.6 Combining Occurences of $\texttt{Ctg}_\texttt{i}\texttt{.Lev}_\texttt{j}$ in PCE$_\Delta$T

It is worth reiterating Axiom 6, 8, and 10 in which we state that a PCE$_\Delta$ cannot emerge without the presence of

- ins$_t$ where $\texttt{Mbr}(\texttt{ins}_t, \texttt{Psn})$ is true,
- ins$_u$ where $\texttt{Mbr}(\texttt{ins}_u, \texttt{Ojt})$ is true, and
- ins$_v$ where $\texttt{Mbr}(\texttt{ins}_v, \texttt{Fld})$ is true;

It is, however, acceptable to encounter a PCE$_\Delta$ without any references to any $\texttt{ins}_x$ or ins$_y$ where $\texttt{Mbr}(\texttt{ins}_x, \texttt{Lcn})$ and $\texttt{Mbr}(\texttt{ins}_y, \texttt{Pfc})$ as stated in axioms 12 and 14.

4.1.6.1 Finding Combinations of Root Occurrences $\texttt{RCtg}_\texttt{i}\texttt{.Lev}_\texttt{j}$, in PCE$_\Delta$T

If we were to assume that a PCE$_\Delta$ could be created by combination of any two, three, four or five RCtg$_i$.Lev$_j$ then there are 26 possible ways that RCtg$_i$.Lev$_j$s could make a PCE$_\Delta$ as the calculations below shows. The number of variations when we have combinations without repetition and where order doesn't matter can be calculated according to the formula

$$\binom{n}{r} = n!/r!\,(n-r)!$$

Where 'n' is the number of possible $\mathtt{RCtg_i.Lev_j}$ which is 5, and 'r' is the number of different $\mathtt{RCtg_i.Lev_j}$ selected from 'n'. 'n' is always 5, but 'r' can vary from 2 to 5. For n=5 and r = 2 the number of combinations would be

$$\binom{5}{2} = \frac{5!}{2!\,3!} = 10$$

Likewise, $\binom{5}{3} = 10, \binom{5}{4} = 5, \binom{5}{5} = 1.$ Therefore the total combination would be 10+10+5 +1 = 26. If we were to allow $\mathtt{ins_t}$s of the same $\mathtt{RCtg_i.Lev_j}$ to make a PCE$_\Delta$, then we would have to add $\binom{5}{1} = 5$ to 26 to get a total of 31 combinations.

However, considering Axiom 11 that a PCE$_\Delta$ must have at least one occurrence of three specific $\mathtt{RCtg_i.Lev_j}$s namely, `Psn`, `Ojt` and `Fld`, the total number of possibilities is actually limited to the presence of `Lcn`, `Pfc`, both `Lcn` and `Pfc`, or neither. According to the above formula this is $\binom{2}{0} + \binom{2}{1} + \binom{2}{2}$ = 1 + 2 + 1 = 4.

Therefore, there are always four possibilities of a PCE$_\Delta$. These possibilities are:

(1) {`Psn, Ojt, Fld`},
(2) {`Psn, Ojt, Fld, Lcn`},
(3) { `Psn, Ojt, Fld, Pfc`}, or
(4) {`Psn, Ojt, Fld, Lcn, Pfc`}.

We illustrate (4) in figure 4.5 showing examples of $\mathtt{ins_t}$s of some occurrences of $\mathtt{RCtg_i.Lev_j}$ which make a particular PCE$_\Delta$. In this figure, $\mathtt{RCtg_i.Lev_j}$s are shown at the left and Mbr for each real world instance $\mathtt{ins_t}$ is depicted to their right. Considering Definition 7 and Axiom 3, readers should interpret Figure 4.5 as `Mbr(margaret, Psn)`, `Mbr(heater152, Ojt)`, `Mbr(feverish, Fld)`, `Mbr(room101, Lcn)`, and `Mbr(heaterPreference, Pfc)`.

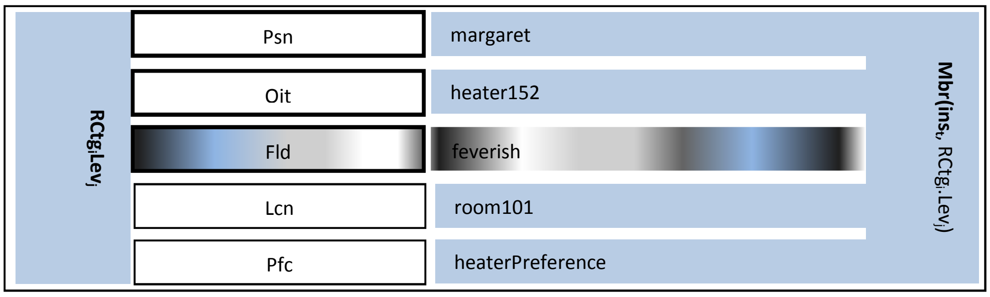


Figure 4.5: $\mathtt{RCtg_i.Lev_j}$ of a PCE with $\mathtt{Mbr(ins_t,\ Ctg_i.Lev_j)}$ examples

4.1.6.2 Illustrating Subsets of $\mathtt{RCtg_i.Lev_j}$, Occurences in $PCE_{\Delta}T$

We illustrate possible subsets of $\mathtt{RCtg_i.Lev_j}$ occurrences in a particular PCE_{Δ}, using the `Fld` occurrence, in order to explain how a detailed abstraction of instances $\mathtt{ins_t}$ and $\mathtt{Mbr(ins_t,\ RCtg_i.Lev_j)}$ work for a particular value of 'i' (i.e. $\mathtt{Mbr(ins_t,\ Fld.Lev_j)}$). In other words, whenever a particular $\mathtt{ins_t}$ is detected, we should be able to determine two things:

1) To which occurrence the $\mathtt{ins_t}$ is a member of (we use the example of $\mathtt{Mbr(ins_t,\ Fld.Lev_j)}$
2) The leaf where $\mathtt{Mbr(ins_t,\ Fld.Lev_1)}$ is true.

However, if there is more than one instance ins_t detected in a particular situation which belongs to the subset of $\mathtt{Fld.Lev_m}$ (`Fld` occurrence at the root level) then we must know, for each detected instance $\mathtt{ins_t}$, which precision (i.e. $\mathtt{Lev_j}$) we need to satisfy when creating $\mathtt{Mbr(ins_t,\ RCtg_i.Lev_j)}$.

In our example of detecting $\mathtt{Mbr(ins_t,\ Fld.Lev_j)}$, we would like to know exactly which $\mathtt{Lev_j}$ is applicable to each particular instance $\mathtt{ins_t}$ detected. In other words, instances $\mathtt{ins_t}$ may belong to any level from $\mathtt{Lev_1}$ (the case of `Mbr`(ins_t, LFld.Lev_j) and $\mathtt{Lev_m}$ (the case of $\mathtt{Mbr(ins_t,\ RFld.Lev_j)}$).

Let us say that we detected instances form `INS`:

- feverish, critical, normal
- diabetes, hypertension, stroke
- residential care, nursing care, continuing care
- NHS, private

The first three can be abstracted into `General Health` category, the second three into `Helath Condition` category, the third three into `Care Home` category, and the last two into `Health` category. However, at the same time, for all of these instances $\mathtt{Mbr(ins_t,\ Fld.Lev_j)}$ must hold. Because our $\mathtt{Ctg_i}$ is actually determined by an occurrence (`Fld`), each of these detected instances must satisfy one of the following: $\mathtt{Mbr(ins_t,\ Fld.Lev_1)}$, $\mathtt{Mbr(ins_t,\ Fld.Lev_2)}$, $\mathtt{Mbr(ins_t,\ Fld.Lev_3)}$, or $\mathtt{Mbr(ins_t,\ Fld.Lev_4)}$. Figure 4.6 shows exactly how the detected instances ins_t are placed correctly within different subsets of the Fld occurrence.

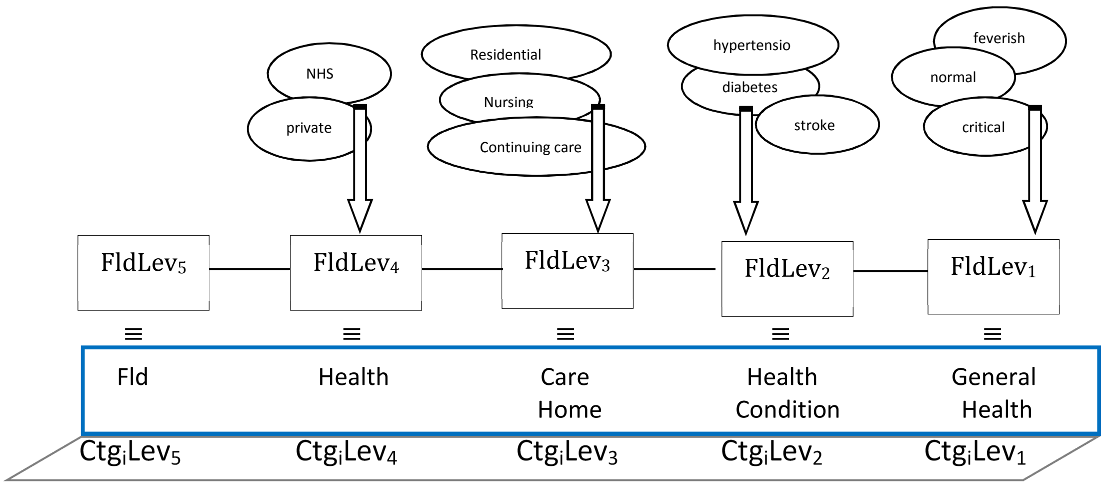


Figure 4.6: An example of a five-level λ $\mathtt{Ctg_i Lev_j}$

The representation of Figure 4.6 in $\mathtt{Mbr(ins_t,\ Ctg_i.Lev_j)}$ format would be

`Mbr(feverish,General health),Mbr(critical,General health),Mbr(normal,General health)`

`Mbr(diabetes,Health Condition), Mbr(hypertension,Health Condition), Mbr(stroke,Health Condition)`

`Mbr(residential care,Care Home),Mbr(nursing care, Care Home),Mbr(continuing care,Care Home)`

`Mbr(NHS, Health), Mbr(Private, Health)`

Where `General health` is $\mathtt{Fld.Lev_1}$, `Health Condition` is $\mathtt{Fld.Lev_2}$, `Care Home` is $\mathtt{Fld.Lev_3}$, and `Health` is $\mathtt{Fld.Lev_4}$.

In other words we have a four-level subset of a $\mathtt{RCtg_i.Lev_j}$, where λ = { $\mathtt{Lev_5}$, $\mathtt{Lev_4}$, $\mathtt{Lev_3}$, $\mathtt{Lev_2}$, $\mathtt{Lev_1}$}. This implies that $\mathtt{Fld.Lev_1}$ is "a subset of" $\mathtt{Fld.Lev_2}$ which is "a subset of" $\mathtt{Fld.Lev_3}$ that is "a subset of" $\mathtt{Fld.Lev_4}$ which is "a subset of" $\mathtt{Fld.Lev_5}$.

In any PCE_Δ therefore, when an $\mathtt{ins_t}$ is detected, we will also know, in addition to $\mathtt{Mbr(ins_t,\ LCtg_i.Lev_j)}$ all other $\mathtt{Ctg_i.Lev_j}$ that $\mathtt{LCtg_i.Lev_j}$ is a "subset of". The successive detection of $\mathtt{Ctg_i.Lev_j}$ (reader is reminded that the term 'successive' is not used pedantically as detection takes place all at once) ends when the last detected $\mathtt{ins_t}$ satisfies $\mathtt{Mbr(ins_t,\ RCtg_i.Lev_j)}$, i.e. any of the following $\mathtt{Mbr}$(ins$_t$, Psn.Lev$_j$), $\mathtt{Mbr(ins_t,\ Ojt_i.Lev_j)}$, $\mathtt{Mbr(ins_t,\ Fld.Lev_j)}$, $\mathtt{Mbr(ins_t,\ Lcn.Lev_j)}$ or $\mathtt{Mbr(ins_t,\ Pfc.Lev_j)}$.

If $\mathtt{ins_t}$ 'feverish' was detected at $\mathtt{LCtg_i.Lev_j}$ in the example shown in Figure 4.6, then we expect the detected information shown in Table 4.1 to be available for further computation:

Expected Information	**Based on**
Mbr(feverish, General Health)	Definition 7, Axiom 2
Lev$_1$ $\prec$ Lev$_2$ $\prec$ Lev$_3$ $\prec$ Lev$_4$ $\prec$ Lev$_5$	Definition 6
Mbr(feverish, Health Condition), Mbr(feverish, Care Home), Mbr(feverish, Health), Mbr(feverish, Fld),	Axiom 4

Table 4.1: An example of situational information received by a PCE

It is worth stressing again that in any particular PCE$_\Delta$T an instance $\mathtt{ins_t}$ of a $\mathtt{LCtg_i.Lev_j}$ cannot be an instance of another leaf at the same level of the same root. To clarify with an example consider an environment in which a user can have different roles and therefore can be of different $\mathtt{LCtg_i.Lev_j}$. In higher education environment, for example, a research student can be a lecturer ($\mathtt{LCtg_i.Lev_j}$ to be lecturer). So, the user has two roles, student and lecturer. Nevertheless, at any particular moment as soon as s/he takes charge of a PCE$_\Delta$, s/he has to indicate which hat s/he has on. So, one and only one of the roles is present at any moment in time. In other words, a detected $\mathtt{LCtg_i.Lev_j}$, in this example, will always have one and only one of $\mathtt{Psn}$ as its $\mathtt{RCtg_i.Lev_j}$.

Axiom 15: $\forall$ ins$_t$, $\exists!$ (x, y) $\in$ {PCE$_\Delta$T | x = LCtg$_i$.Lev$_j$, y = RCtg$_i$.Lev$_j$ }

4.1.7 An Instance Characteristics of a Taxonomical Element

The taxonomical structure PCE$_\Delta$T contains a set of $\mathtt{Mbr(ins_t,\ Ctg_i.Lev_j)}$ which were abstracted from real world instances. However, each PCE$_\Delta$ may contain additional semantics, which may have not been captured when (a) abstracting instances $\mathtt{ins_t}$ into $\mathtt{Ctg_i.Lev_j}$ and (b) securing that for each detected $\mathtt{ins_t}$ the

$\texttt{Mbr}(\texttt{ins}_t,\ \texttt{Ctg}_i.\texttt{Lev}_j)$ is true. Therefore, if we had any additional semantics in PCE_Δ, which is either

1) not captured by initial abstractions from real world instances or
2) too specific to be captured in our initial abstractions because we might decide not to abstract them into any $\texttt{Ctg}_i.\texttt{Lev}_j$

we introduce another element of $PCE_\Delta T$, which cover cases in 1) and 2). These cases will be easily found when detecting main participants in the PCE_Δ.

If detected information in a particular situation Δ, which creates the PCE_Δ shows that it can not be abstracted into any of the occurrences $\texttt{RCtg}_i.\texttt{Lev}_j$ and their subsets, then we should be able to find an element within the taxonomical structure $PCE_\Delta T$ which may accommodate such semantics.

Therefore, we introduce characteristic $\texttt{chr}_q$ of $\texttt{Mbr}(\texttt{ins}_t,\texttt{Ctg}_i.\texttt{Lev}_j)$ as a new taxonomical element to be a description of a particular $\texttt{Mbr}(\texttt{ins}_t,\texttt{Ctg}_i.\texttt{Lev}_j)$. It is natural to expect that these characteristics are initially reserved for the leaves of our taxonomical structure, i.e. we will always need characteristics of $\texttt{LCtg}_i.\texttt{Lev}_j$ with value $\texttt{vlu}_q$ describing $(\texttt{ins}_t,\ \texttt{LCtg}_i.\texttt{Lev}_j)$. However, they may appear as an additional semantics for any other taxonomical element $\texttt{Mbr}(\texttt{ins}_t,\ \texttt{Ctg}_i.\texttt{Lev}_j)$.

Definition 16: An instance characteristic chr_q is a description of a $Mbr(ins_t, Ctg_i.Lev_j)$ with value vlu_q in the format of a triplet $((Mbr(ins_t, Ctg_i.Lev_j), chr_q, vlu_q))$.

4.1.8 Illustrating an Instance Characteristics $\texttt{chr}_q$ of a Taxonomical Element

A particular PCE_Δ has occurred in a health domain. This is known when an $\texttt{ins}_t$ of $\texttt{Fld}$ which is an occurrence of $\texttt{RCtg}_i.\texttt{Lev}_j$ (Definition 13), is detected. As the $\texttt{Mbr}\ (\texttt{inst},\ \texttt{RCtg}_i.\texttt{Lev}_j)$ nor any other $\texttt{Ctg}_i.\texttt{Lev}_j$ of $\texttt{Fld}$ are shown in Figure 4.7, it is not clear how many $\texttt{Ctg}_i.\texttt{Lev}_j$ there are between $\texttt{LCtg}_i.\texttt{Lev}_j$ and $\texttt{RCtg}_i.\texttt{Lev}_j \equiv \texttt{Fld}$. That is why there is '?' in the $\texttt{FldLev}_?$.

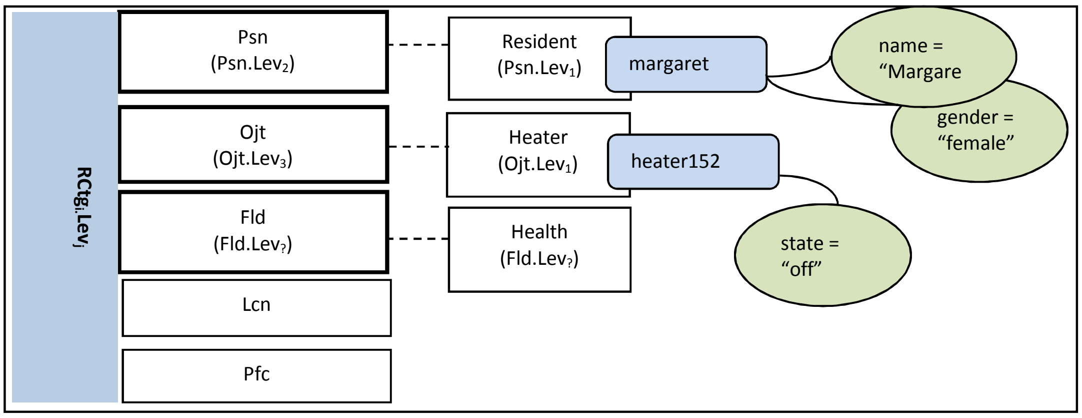


Figure 4.7: Partial graphical representation of an example of a PCE$_{\Delta}$

The representation of what is illustrated in Figure 4.7, and what definitions and axioms they are based on is shown in Table 4.2.

	Representation of Detected Information	**Based on**
a	an instance margaret, Mbr (margaret, Resident), LCtg$_i$.Lev$_j$ $\equiv$ PsnLev$_1$ = Resident $\prec$ RCtg$_i$.Lev$_j$ $\equiv$ PsnLev$_2$ = Psn, are detected.	Definition 7, Axiom 2, 4
b	The name and gender which are chr$_q$ describing 'margaret' become (margaret, name, "Margaret") and (margaret, gender, "female")	Definition 16
c	an instance heater152, Mbr (heater152, Heater), LCtg$_i$.Lev$_j$ $\equiv$ OjtLev$_1$ =Heater $\prec$ OjtLev$_2$ = Allocated Object $\prec$ RCtg$_i$.Lev$_j$ $\equiv$ OjtLev$_3$ = Ojt are detected.	Definition 7, Axiom 2, 4
d	The state chr$_q$ that describes 'heater152' becomes (heater152, state, "off").	Definition 16

Table 4.2: PCE$_{\Delta}$T situational information of Figure 4.7

4.1.9 Relationships $\mathtt{rlp}_r$ in PCE$_{\Delta}$T

In each PCE$_{\Delta}$ we have to allow relationships within the PCE$_{\Delta}$T between taxonomical `Mbr (ins`$_t$`, Ctg`$_i$`.Lev`$_j$`)`. For example, we may have a relation between `Mbr (heater152, Heater)` and `Mbr(margaret, Resident)`, representing the semantics of a relationship `rlp`$_r$ between these two taxonomical elements.

Definition 17: A relationship rlp$_r$ between PCE$_{\Delta}$T elements is a binary relationship between Mbr(ins$_t$, Ctg$_i$.Lev$_j$) and Mbr(ins$_u$, Ctg$_x$Lev$_y$) and denoted as rlp$_r$(Mbr(ins$_t$, Ctg$_i$.Lev$_j$), Mbr(ins$_u$, Ctg$_x$Lev$_y$)). This implies that these two instances ins$_t$ and ins$_u$ which are members of category Ctg$_i$.Lev$_j$ and Ctg$_x$Lev$_y$ are related by rlp$_r$.

We have to think about a set of particular `rlp`$_r$ which should be applied to occurrences of the `RCtg`$_i$`.Lev`$_j$. We illustrate and define essential `rlp`$_r$ through axioms below in order to emphasise that in a particular PCE$_{\Delta}$T there are additional domain and situation-specific semantics, and that there are domain and situation-specific $\mathtt{rlp}_r$ due to the existence of domain and situation-specific occurrences of $\mathtt{RCtg}_i.\mathtt{Lev}_j$.

Consequently characteristics, ($\mathtt{Mbr}(\mathtt{ins}_t,\ \mathtt{Ojt.Lev}_j)$, $\mathtt{chr}_q$, $\mathtt{vlu}_q$) from section 4.4 and relationships $\mathtt{rlp}_r$ are bringing more semantics to the taxonomical elements of $\text{PCE}_\Delta\text{T}$. However, the former is defined solely on $\mathtt{Mbr}(\mathtt{ins}_t,\ \mathtt{Ctg}_i.\mathtt{Lev}_j)$ and latter on $\mathtt{RCtg}_i.\mathtt{Lev}_j$ occurrences.

4.1.9.1 Relationships between Taxonomical Elements at the Root of $\text{PCE}_\Delta\text{T}$

As any PCE is about a specific domain, and given that it is impossible to think of a PCE_Δ without the presence of user (Axiom 6), there is always a $\mathtt{rlp}_r$ between $\mathtt{Mbr}(\mathtt{ins}_t,\ \mathtt{RCtg}_i.\mathtt{Lev}_j)$ and $\mathtt{Mbr}(\mathtt{ins}_u,\ \mathtt{RCtg}_x\mathtt{Lev}_y)$ where $\mathtt{LCtg}_i.\mathtt{Lev}_j \equiv \mathtt{Psn}$ and $\mathtt{RCtg}_x\mathtt{Lev}_y \equiv \mathtt{Fld}$. Therefore, there is always an inherent $\mathtt{rlp}_r$ that exist in any PCE for any PCE_Δ. This inherent relation is “isAssociatedWith”.

> **Axiom 16:** $\forall$ $\text{PCE}_\Delta\text{T}$ if $\exists$ Mbr(ins_t, Psn.Levj) and Mbr(ins_t, Fld.Levj) => $\exists$ rlp_r which denotes isAssociatedWith where isAssociatedWith(Mbr(ins_t, Psn.Levj), Mbr(ins_t, Fld.Levj))

When an occurrence of Lcn plays a role in a particular PCE_Δ, a relationship $\mathtt{rlp}_r$ exists between the user $\mathtt{ins}_t$ ($\mathtt{Mbr}(\mathtt{ins}_t,\ \mathtt{Psn})$) and the location $\mathtt{ins}_u$ ($\mathtt{Mbr}(\mathtt{ins}_v,\ \mathtt{Lcn})$) and/or between an object $\mathtt{ins}_v$ ($\mathtt{Mbr}(\mathtt{ins}_t,\ \mathtt{Ojt})$) and the location $\mathtt{ins}_u$ ($\mathtt{Mbr}(\mathtt{ins}_t,\ \mathtt{Lcn})$).

> **Axiom 17:** If in $\text{PCE}_\Delta\text{T}$ $\exists$ Mbr(ins_t, Psn.Levj) and Mbr(ins_t, Lcn.Levj) => $\exists$ rlp_r which denotes isIn where isIn(Mbr(ins_t, Psn.Levj), Mbr(ins_t, Lcn.Levj))

When there is an occurrence of Lcn in a particular PCE_Δ, a relationship $\mathtt{rlp}_r$ exists between the real world user $\mathtt{Mbr}(\mathtt{ins}_t,\ \mathtt{Psn})$ and the real world location $\mathtt{Mbr}(\mathtt{ins}_v,\ \mathtt{Lcn})$ and/or between a real world object $\mathtt{Mbr}(\mathtt{ins}_t,\ \mathtt{Ojt})$ and $\mathtt{Mbr}(\mathtt{ins}_v,\ \mathtt{Lcn})$.

> **Axiom 18:** If in $\text{PCE}_\Delta\text{T}$ $\exists$ Mbr (ins_t, Ojt.Levj) and Mbr (ins_t, Lcn.Levj) => $\exists$ rlp_r which denotes isCurrentlyIn where isCurrentlyIn (Mbr (ins_t, Ojt.Levj), Mbr (ins_t, Lcn.Levj))

When an occurrence of $\mathtt{RCtg}_i.\mathtt{Lev}_j \equiv \mathtt{Pfc}$ plays a role in delivering a service in a particular PCE_Δ, a relationship $\mathtt{rlp}_r$ exists between the user $\mathtt{ins}_t$ ($\mathtt{Mbr}(\mathtt{ins}_t,\ \mathtt{Psn})$) and the preference $\mathtt{ins}_u$ ($\mathtt{Mbr}(\mathtt{ins}_t,\ \mathtt{Pfc})$).

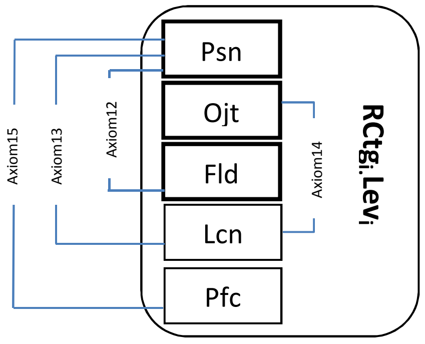


Figure 4.8 Relationship $\mathtt{rlp}_r$ between taxonomical roots

> **Axiom 19:** If in $PCE_{\Delta}T$ ∃ Mbr(ins_t, Psn.Lev_j) and Mbr(ins_t, Pfc.Lev_j) => ∃ rlp_r which denotes hasPreference where hasPreference (Mbr(ins_t, Psn.Lev_j), Mbr(ins_t, Pfc.Lev_j))

The relationship between taxonomical roots are illustrated in Figure 4.8

4.1.9.1.1 Consequences of the Relationship 'hasPreference'

For an occurrence `Pfc`, $\mathtt{RCtg_i.Lev_j} \equiv$ `Pfc`, according to Axiom 19 a $\mathtt{rlp}_r$ of hasPreference must exist between a `Mbr` ($\mathtt{ins_t}$, $\mathtt{Psn.Lev_j}$) and a `Mbr` ($\mathtt{ins_t}$, $\mathtt{Pfc.Lev_j}$). An example for `Mbr` ($\mathtt{ins_t}$, $\mathtt{Pfc.Lev_j}$) can be `Mbr` (`heaterPreference`, $\mathtt{Pfc.Lev_j}$) which shows a real world instance "`heaterPreference`". This instance which is a user preference for some real world objects requires more precision. This will result in a subset of Pfc root category, $\mathtt{Pfc.Lev_m}$, that is the abstraction of all real world instances similar to "`heaterPreference`". We refer to this subset as $\mathtt{Pfc.Lev_{m-1}}$ and named it `Ojt-specific-Pfc`. Therefore $\mathtt{Pfc.Lev_{m-1}}$ = `Ojt-specific-Pfc` which is itself a `Pfc`. By the same token we have defined $\mathtt{Psn.Lev_{m-1}}$ = `Psn-specific-Pfc` which is a `Psn` when there are real world instances that can be abstracted to some preferences for people.

Likewise, $\mathtt{Lcn.Lev_{m-1}}$ = `Lcn-specific-Pfc` which is a `Lcn` are used when there are real world instances that can be abstracted to some preferences for locations.

Therefore, $\mathtt{RCtg_i.Lev_j} \equiv$ `Pfc` must have three subsets $\mathtt{Pfc.Lev_{m-1}}$. In other words in any PCE there exists $\mathtt{Mbr(ins_t, Psn\text{-}specific\text{-}Pfc.Lev_{m-1})}$,

$\mathtt{Mbr(ins_t,Ojt\text{-}specific\text{-}Pfc.Lev_{m-1})}$, and $\mathtt{Mbr(ins_t,Lcn\text{-}specific\text{-}Pfc.Lev_{m-1})}$.

Axiom 20: If in $PCE_{\Delta}T$ $\exists$ rlp_r which denotes hasPreference where hasPreference ($Mbr(ins_t, Psn.Lev_j)$, $Mbr(ins_u, Pfc.Lev_k)$) holds => $\exists$ $Mbr(ins_u, Pfc.Lev_{m-1})$ where ($Pfc.Lev_{m-1}$ = Psn-specific-Pfc) or $\exists$ $Mbr(ins_u, Pfc.Lev_{m-1})$ where ($Pfc.Lev_{m-1}$ = Ojt-specific-Pfc) or $\exists$ $Mbr(ins_u, Pfc.Lev_{m-1})$ where ($Pfc.Lev_{m-1}$ = Lcn-specific-Pfc).

4.1.9.1.2 Consequences of Creating a new Taxonomical Element

The inclusion of the relationship '`hasPreference`' resulted in three new $\mathtt{Mbr\ (ins_u,\ Pfc.Lev_{m-1})}$ which in turn cause the inclusion of three relationships $\mathtt{rlp_r(Mbr(ins_t,\ Ctg_i.Lev_j),\ Mbr(ins_u,\ Ctg_xLev_y))}$. In these relationships $\mathtt{Mbr\ (ins_t,\ Ctg_i.Lev_j)}$ is $\mathtt{Mbr\ (ins_t,\ Pfc.Lev_{m-1})}$ and $\mathtt{Mbr\ (ins_u,\ Ctg_xLev_y)}$ can be either $\mathtt{Mbr\ (ins_u,\ PsnLev_y)}$, $\mathtt{Mbr\ (ins_u,\ OjtLev_y)}$ or $\mathtt{Mbr(ins_u,\ LcnLev_y)}$. These possibilities are shown in the following axioms.

Axiom 21: $\forall$ $PCE_{\Delta}T$ if $\exists$ $Mbr\ (ins_t, Psn.Lev_j)$ and $Mbr\ (ins_u, Pfc.Lev_{m-1})$ where $Pfc.Lev_m$ is $RCtg_i.Lev_j$ => $\exists$ rlp_r which denotes isAbout where isAbout ($Mbr(ins_t, Psn.Lev_j)$, $Mbr(ins_u, Pfc.Lev_{m-1})$)

Axiom 22: $\forall$ $PCE_{\Delta}T$ if $\exists$ $Mbr\ (ins_t, Ojt.Lev_j)$ and $Mbr\ (ins_u, Pfc.Lev_{m-1})$ where $Pfc.Lev_m$ is $RCtg_i.Lev_j$ => $\exists$ rlp_r which denotes isRelatedTo where isRelatedTo ($Mbr(ins_t, Ojt.Lev_j)$, $Mbr(ins_u, Pfc.Lev_{m-1})$)

Axiom 23: $\forall$ $PCE_{\Delta}T$ if $\exists$ $Mbr\ (ins_t, Lcn.Lev_j)$ and $Mbr\ (ins_u, Pfc.Lev_{m-1})$ where $Pfc.Lev_m$ is $RCtg_i.Lev_j$ => $\exists$ rlp_r which denotes isFor where isFor ($Mbr\ (ins_t, Lcn.Lev_j)$, $Mbr\ (ins_u, Pfc.Lev_{m-1})$)

When there exists a relationship $\mathtt{rlp_r(Mbr(ins_t,\ Ctg_i.Lev_j),\ Mbr(ins_u,\ Ctg_xLev_y))}$ then all the subsets of $\mathtt{Mbr(ins_t,\ Ctg_i.Lev_j)}$ and $\mathtt{Mbr(ins_u,\ Ctg_xLev_y)}$ including the leaves will share the $\mathtt{rlp_r}$ such that $\mathtt{rlp_r(Mbr(ins_t,\ Ctg_iLev_h),\ Mbr(ins_u,\ Ctg_xLev_v))}$ holds for $h<j$ and $v<y$.

Axiom 24: $\forall$ $PCE_{\Delta}T$ if $\exists$ $rlp_r(Mbr(ins_t, Ctg_i.Lev_j), Mbr(ins_u, Ctg_xLev_y))$ => $rlp_r(Mbr(ins_v, Ctg_iLev_h)\ , Mbr(ins_w, Ctg_xLev_v))\,|$ where $h<j$, $v<y$.

4.1.10 Summarising $PCE_{\Delta}T$

We have provided in the previous sections (4.1.1 – 4.1.9) seventeen definitions and twenty four axioms. Through these definitions and axioms we have defined what a PCE is, what the role of situations in PCEs that determine services to be delivered to the users of PCEs are. These definitions also set the foundation for what the taxonomical structure and elements of a PCE are. We have summarised these definitions diagrammatically in Figure 4.9.

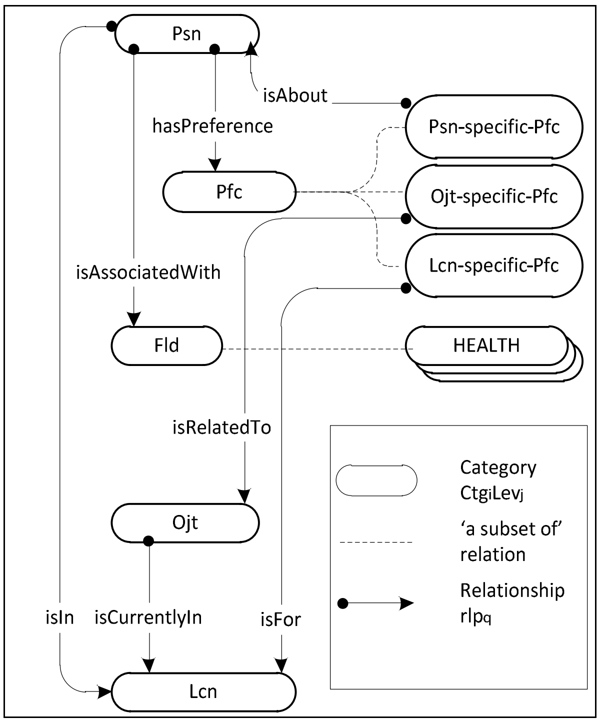


Figure 4.9: Summarisation of generic PCE$_{\Delta}$T

At the left side of Figure 4.9 all five occurrences of $\mathtt{RCtg_i.Lev_j}$ depicted in Figure 4.8 and defined in axioms 12-15 are shown. Out of these occurrences `Pfc` is the only one with defined "sub-set" relationship. According to Axiom 20, when there is an instance of `Pfc` it must be one of its subset, which are `Psn-specific-Pfc`, `Ojt-specific-Pfc`, and `Lcn-specific-Pfc`. One of the features of a PCE is its domain specificity (P18 table 3.3). Domain-specific information in a PCE is represented as subsets of `Fld` (Definition 13, Axiom 9). In Figure 4.9 Health $\mathtt{Ctg_i.Lev_j}$ is shown as an example; it can be any domain of interest.

Relationships, $\mathtt{rlp_r}$ between $\mathtt{RCtg_i.Lev_j}$ defined in axioms 16-19, and the $\mathtt{rlp_r}$ between subsets of `Pfc` and three $\mathtt{RCtg_i.Lev_j}$, namely, `Psn`, `Ojt`, and `Lcn`, axioms 21-23, are also shown in Figure 4.9. For the sake of simplicity and avoiding a cluttered diagram, characteristics, $\mathtt{chr_q}$ of $\mathtt{Ctg_i.Lev_j}$ (Definition 16) are not shown in the diagram.

We would also like to bring to the attention of reader that we could not possibly show the extension of $PCE_{\Delta}T$ for obvious reason that this is solely dependent on different PCE_{Δ}. Nevertheless, examples in Figure 4.5-7 illustrate that the extension of Figure 4.9 may be required.

In the following section we define steps which secure the creation of the generic and extended elements of the $PCE_{\Delta}T$ in order to deliver a situation-specific service.

4.2 Formal Computational Model (FCM) in PCE

The FCM secures the delivery of a situation-specific service(s) in a particular situation PCE_{Δ} by creating a situation-specific taxonomical structure $PCE_{\Delta}T$ for the PCE_{Δ}, and the reasoning upon the $PCE_{\Delta}T$ taxonomical elements in order to deliver a situation-specific service for the PCE_{Δ}.

Formal detailed specification of the computation for a particular PCE_{Δ}, to deliver a service to the user of the PCE is explained in the next section, followed by a section on the representation of the FCM in pseudo code using OWL terminologies, and a separate section on the reasoning upon the $PCE_{\Delta}T$ taxonomical elements in order to deliver a situation-specific service for the PCE_{Δ}.

4.2.1 The FCM with Loops and Steps Towards $PCE_{\Delta}T$

Computations which deliver a domain and situation-specific service in a PCE_{Δ} are divided into three parts as depicted in Figure 4.10. The first part addresses the creation and/or extension of $\mathtt{Ctg_i.Lev_j}$, insertion of real world instances $\mathtt{ins_t}$, adding characteristics $\mathtt{chr_q}$ to $\mathtt{Ctg_i.Lev_j}$ and finally assigning characteristics' value $\mathtt{vlu_q}$ to $\mathtt{chr_q}$ for each $\mathtt{ins_t}$, that is $\mathtt{((Mbr(ins_t,Ctg_i.Lev_j),chr_q,vlu_q))}$.

The second part which is bordered with dashed line addresses the creation of generic relationships (axioms 16-19, and 21-23), $\mathtt{rlp_r(Mbr(ins_t,\ Ctg_i.Lev_j),\ Mbr(ins_u,\ Ctg_xLev_y))}$. The third part deals with the creation of situation-specific extended relationships $\mathtt{rlp_r(Mbr(ins_t,\ Ctg_i.Lev_j),\ Mbr(ins_u,\ Ctg_xLev_y))}$.

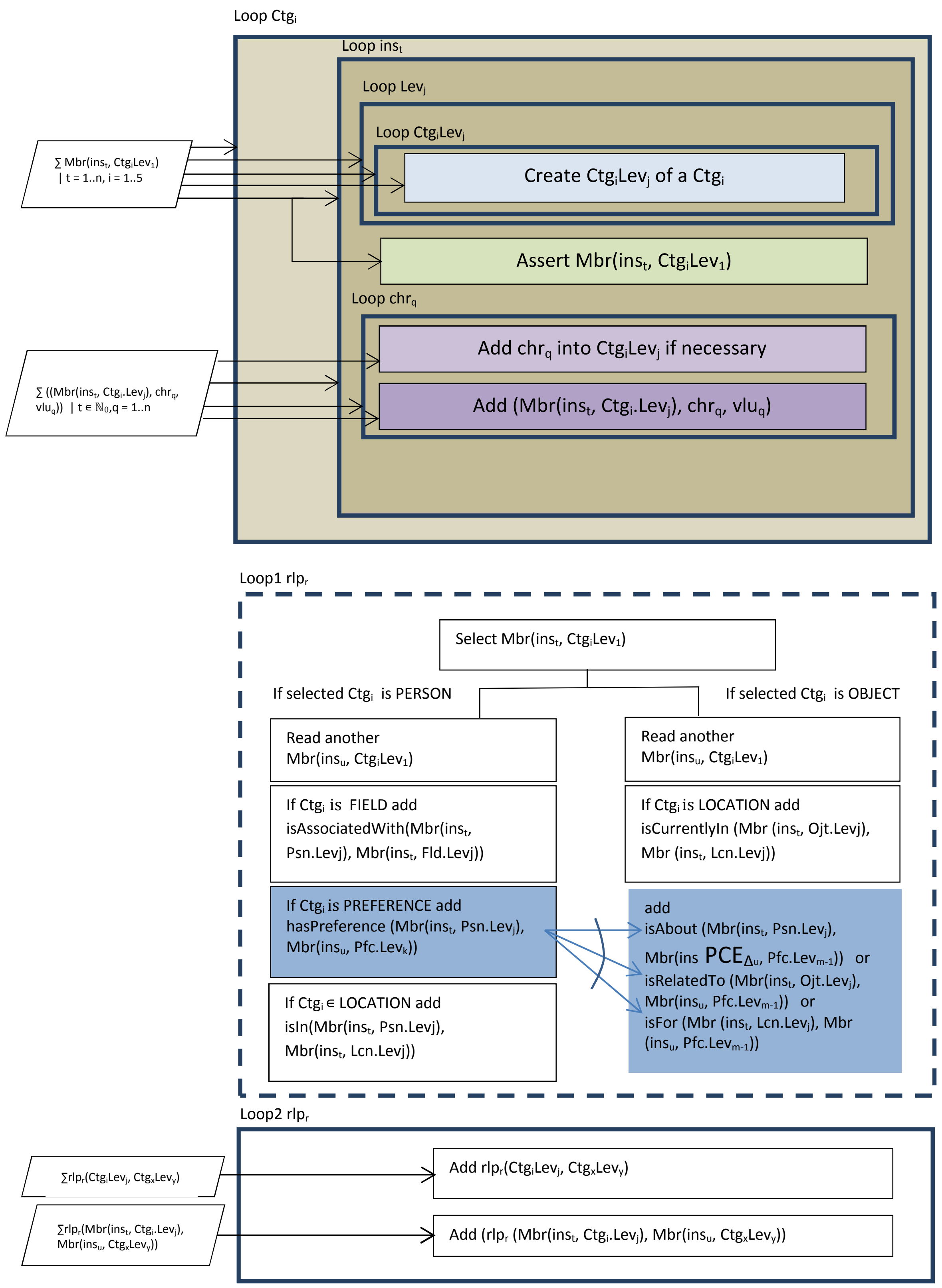

Loop Ctg_i
Loop ins_t
Loop Lev_j
Loop Ctg_iLev_j
Create Ctg_iLev_j of a Ctg_i
Assert $Mbr(ins_t, Ctg_iLev_1)$
Loop chr_q
Add chr_q into Ctg_iLev_j if necessary
Add $(Mbr(ins_t, Ctg_i.Lev_j), chr_q, vlu_q)$
$\sum Mbr(ins_t, Ctg_iLev_1)$ $\mid t = 1..n, i = 1..5$
$\sum ((Mbr(ins_t, Ctg_i.Lev_j), chr_q, vlu_q)) \mid t \in \mathbb{N}_0, q = 1..n$
Loop1 rlp_r
Select $Mbr(ins_t, Ctg_iLev_1)$
If selected Ctg_i is PERSON
If selected Ctg_i is OBJECT
Read another $Mbr(ins_u, Ctg_iLev_1)$
Read another $Mbr(ins_u, Ctg_iLev_1)$
If Ctg_i is FIELD add isAssociatedWith(Mbr(ins_t, Psn.Levj), Mbr(ins_t, Fld.Levj))
If Ctg_i is LOCATION add isCurrentlyIn (Mbr (ins_t, Ojt.Levj), Mbr (ins_t, Lcn.Levj))
If Ctg_i is PREFERENCE add hasPreference (Mbr(ins_t, Psn.Lev_j), Mbr(ins_u, Pfc.Lev_k))
add isAbout (Mbr(ins_t, Psn.Lev_j), Mbr(ins $PCE_{\Delta u}$, Pfc.Lev_{m-1})) or isRelatedTo (Mbr(ins_t, Ojt.Lev_j), Mbr(ins_u, Pfc.Lev_{m-1})) or isFor (Mbr (ins_t, Lcn.Lev_j), Mbr (ins_u, Pfc.Lev_{m-1}))
If $Ctg_i \in$ LOCATION add isIn(Mbr(ins_t, Psn.Levj), Mbr(ins_t, Lcn.Levj))
Loop2 rlp_r
$\sum rlp_r(Ctg_iLev_j, Ctg_xLev_y)$
$\sum rlp_r(Mbr(ins_t, Ctg_i.Lev_j), Mbr(ins_u, Ctg_xLev_y))$
Add $rlp_r(Ctg_iLev_j, Ctg_xLev_y)$
Add $(rlp_r (Mbr(ins_t, Ctg_i.Lev_j), Mbr(ins_u, Ctg_xLev_y))$

Figure 4.10: Formalised Computational Model in PCEs

4.2.1.1 Creation of $\mathtt{Ctg_i.Lev_j}$,insertion of $\mathtt{ins_t}$,addition of $\mathtt{chr_q}$ and $\mathtt{vlu_q}$

The first part of our proposed FCM, which does 1) and 2) from section 4.1.10, addresses the creation and/or extension of $\mathtt{Ctg_i.Lev_j}$, insertion of real world instances, $\mathtt{ins_t}$, adding characteristics $\mathtt{chr_q}$ to $\mathtt{Ctg_i.Lev_j}$ and finally assigning characteristics' value, $\mathtt{vlu_q}$ to $\mathtt{chr_q}$ for all $\mathtt{ins_t,(Mbr(ins_t,Ctg_i.Lev_j),chr_q,vlu_q)}$ when a PCE receives interpreted contextual data form the environment. Part one of FCM goes through a series of inner and outer loops as shown in Figure 4.10. In this section we explain this part of FCM.

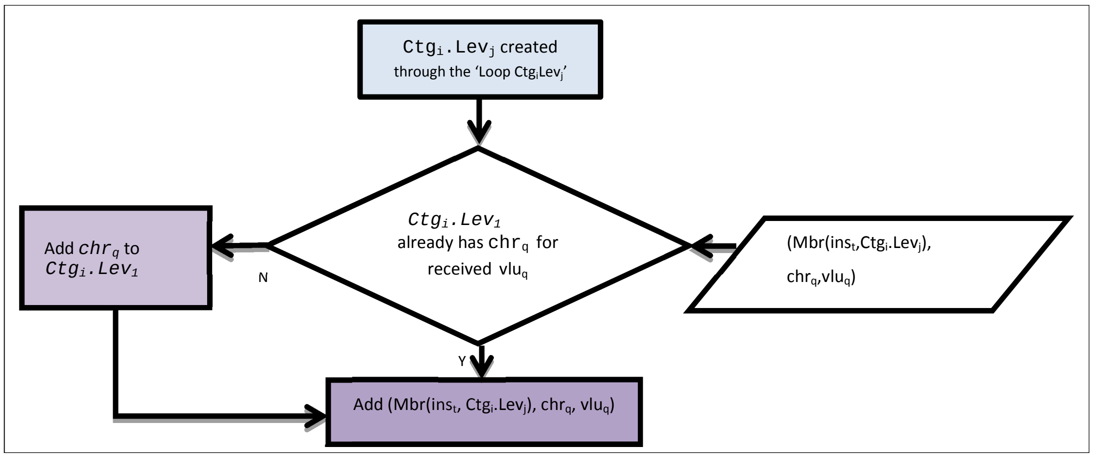
The FCM starts with the most inner loop 'Loop $Ctg_i.Lev_j$', which is within three outer loops, 'Loop Lev_j', 'Loop ins_t' and 'Loop Ctg_i' as shown in Figure 4.10. As we expect, for each cycle of the outer loop the inner loop goes through complete cycles. The loop 'Loop Ctg_i' starts its first iteration with i = 1. Consequently the 'Loop ins_t also starts its first iteration with t = 1, followed by 'Loop Lev_j' with j=2 in the first iteration. 'Loop Lev_j' index starts with 2 because in the statement just before the loop, the leaf category $\mathtt{Ctg_i.Lev_1}$ was read. If the read $\mathtt{Ctg_iLev_2}$ meets the condition statement of the loop, the first $\mathtt{Ctg_i.Lev_j}$ of the $\mathtt{RCtg_i.Lev_j}$, which is either $\mathtt{Psn,\ Ojt,\ Fld,\ Lcn\ or\ Pfc}$, will be created.

Although receiving an interpreted contextual data of a real world instance $\mathtt{ins_t}$ is always accompanied by its leaf category $\mathtt{Ctg_i.Lev_1}$ (Definition 9, Axiom 2), the creation of all $\mathtt{Ctg_i.Lev_j}$ levels that $\mathtt{Ctg_i.Lev_1}$ is a subset of always starts with the creation of the immediate (or first) subset $\mathtt{Ctg_i.Lev_j}$ of any of the five $\mathtt{RCtg_i.Lev_j}$. Consequently, once all its higher level $\mathtt{Ctg_i.Lev_j}$ have been created, a leaf category $\mathtt{Ctg_i.Lev_1}$ will be the last $\mathtt{Ctg_i.Lev_j}$ of a $\mathtt{Ctg_i}$ to be created. This part of FCM is shown in light blue in Figure 4.10.

Once all $\mathtt{Ctg_i.Lev_j}$ have been created for all real world instance $\mathtt{ins_t}$, through the iterations explained above, all $\mathtt{ins_t}$s will be added to the FCM. This part of FCM is shown in light green in Figure 4.10.

When an ins_t for a particular PCE_{Δ} is received, all characteristics, chr_q with their values, vlu_q are also received (Definition 16). For every received chr_q, there must be a place in the corresponding $Ctg_i.Lev_1$ representing the leaf category of the ins_t (Definition 9, Axiom 2), that is $Mbr(ins_t, Ctg_i.Lev_1)$. Considering the fact that there are some intrinsic chr_q applicable to Psn, Ojt, Fld, Lcn or Pfc in all PCEs across domains, and that a $Ctg_i.Lev_j$ that is a subset of $Ctg_i.Lev_k$ inherits all chr_q and rlp_t of $Ctg_i.Lev_k$ (Definition 16), sometimes a received vlu_q has an abstract chr_q already represented in the corresponding $Ctg_i.Lev_1$. In this case just$(Mbr(ins_t, Ctg_i.Lev_j), chr_q, vlu_q)$will be added. Otherwise, if the chr_q to resemble the received vlu_q does not exist, a new chr_q will be added to the $Ctg_i.Lev_1$ first and$(Mbr(ins_t, Ctg_i.Lev_j), chr_q, vlu_q)$will be added next.

Adding $(Mbr(ins_t, Ctg_i.Lev_j), chr_q, vlu_q)$to FCM, whether chr_q already exists or added as an extension to $Ctg_i.Lev_1$, the received value vlu_q will be added for the particular ins_t. This part of FCM is depicted in Figure 4.11.



$Ctg_i.Lev_j$ created through the 'Loop Ctg_iLev_j'
$Ctg_i.Lev_1$ already has chr_q for received vlu_q
Add chr_q to $Ctg_i.Lev_1$
N
Y
$(Mbr(ins_t,Ctg_i.Lev_j), chr_q,vlu_q)$
Add $(Mbr(ins_t, Ctg_i.Lev_j), chr_q, vlu_q)$

Figure 4.11: Adding $(Mbr(ins_t, Ctg_i.Lev_j), chr_q, vlu_q)$ to FCM

Once all $(Mbr(ins_t, Ctg_i.Lev_j), chr_q, vlu_q)$ were added to the FCM, the next round of iteration of the 'Loop ins_t' for the next ins_t of the same Ctg_i will take place. When there is no more real world instance of the same Ctg_i,the FCM will move to the most outer loop which is the 'Loop Ctg_i' to read the next Ctg_i. This will be repeated for each Ctg_i until there are no further Ctg_is.

4.2.1.2 Creation of Generic Relationships $\mathtt{rlp_r}$

PCE$_\Delta$T relationships $\mathtt{rlp_r}$ are divided into two types. Relationships, which are by definition present in any PCE$_\Delta$, and relationships which are doman and situation-specific relationships that their existence depend on the domain and the occurrence of a particular situation Δ. So far as the pre-defined relationships are concerned, when the 'Loop *Ctg$_i$' loop* is over, 'Loop1 rlp$_r$' in Figure 4.10 will be the next step in FCM. This part of the model is bordered with dashed line. Because these $\mathtt{rlp_r}$s are present in all PCE$_\Delta$ across domains, FCM might have it already ready for use, and therefore there is no need for their creation. Otherwise, they have to be created as prescribed in the FCM.

'Loop1 rlp$_r$' box shows that the creation of $\mathtt{rlp_r}$s depends on the membership of $\mathtt{ins_t}$. In other words, $\mathtt{Mbr(ins_t,Ctg_i.Lev_j)}$determines what generic $\mathtt{rlp_r}$ is needed to be created.

4.2.1.3 Creation of Extended Relationships $\mathtt{rlp_r}$

Following the middle part of the FCM in Figure 4.10, domain and situation specific relationships $\mathtt{rlp_r}$ received from PCE for the situation Δ, will need to be represented for further computation so that all semantically significant information about the PCE$_\Delta$ are accounted for. This part of the FCM, which is referred to as 'Loop2 rlp$_r$' in Figure 4.10, is again an iterative process in which two real world instances that have category relation $\mathtt{rlp_t}$ between them are selected and an $\mathtt{rlp_t}$ is added to the model, $\mathtt{rlp_r(Ctg_iLev_j,Ctg_xLev_y)}$. Once the relationship $\mathtt{rlp_r}$ is added, the $\mathtt{rlp_r}$ between the individuals corresponding to the selected instances will be added, $\mathtt{rlp_r(Mbr(ins_t,Ctg_i.Lev_j),Mbr(ins_u,Ctg_xLev_y))}$.

4.2.2 The FCM in Pseudo Code Using OWL Terminologies

In chapter 2 we have explained and justified our choice of SWRL-enabled OWL ontological modelling for knowledge representation and reasoning upon it. To show the pseudo code for the FCM, Owl Ontology Language (OWL) is used. Considering that the FCM will start with detected information from the PCE. So far we have used notions specific to PCE$_\Delta$T, thus there is a need for mapping taxonomical terminologies of this chapter to OWL computation before any code resembling any implementation can be addressed. In the following section mapping PCE$_{\Delta}$T to OWL terminology is show, followed by the pseudo code of the FCM.

4.2.2.1 Mapping PCE$_{\Delta}$T to OWL Terminologies

The mapping provided in this sub-section serves:

- Ease of transformation of the detected information from the environment to OWL computational model;
- Ease of corresponding notions identified in taxonomy PCE$_{\Delta}$T to counterpart concepts in OWL;
- Ease of understanding the formal pseudo code

Table 4.3 summarises the mapping of these terms. On the left side of the table notions used for collecting information about any situation Δ is shown, whereby their counterpart concepts in OWL is listed on the right.

It is worth stressing that although in the Semantic Web literature, terms such as category, concept or type are also used for 'class' or 'instance' and 'individual' are used interchangeably, in this thesis we restrict ourselves to the terms as shown in table 4.3 without any reservations unless stated otherwise.

Environment (User Inputs)			(OWL Terminology)	
	Term	Abbreviation	Term	
Notion	Category	Ctg_i	Base Class	Concept
	Level	Lev	Order	
	Category level	$Ctg_i.Lev_j$	Subclass	
	Leaf category	$LCtg_i.Lev_j \equiv Ctg_i.Lev_1$	Subclass	
	Root category	$RCtg_i.Lev_j \equiv Ctg_i.Lev_m$	Base Class	
	Category Relationship	Rlp	Object property	
	Category Instance characteristic	Chr	Data type property	
	Category Instance characteristic value	Vlu	Range	
	Real world Instance	Ins	Individual	
	Category membership	Mbr		
Taxonomy $PCE_\Delta T$	Psn		PERSON	**OWL Ontology**
	Ojt		OBJECT	
	Fld		FIELD	
	Lcn		LOCATION	
	Pfc		PREFERENCE	
	Psn-specific-Pfc		PERSON-SPECIFIC-PREFERENCE	
	Ojt-specific-Pfc		OBJECT-SPECIFIC-PREFERENCE	
	Lcn-specific-Pfc		LOCATION- SPECIFIC-PREFERENCE	

Table 4.3: Mapping $PCE_\Delta T$ to OWL terminologies

4.2.2.2 The FCM in Pseudo Code

In this section the pseudo code for FCM in PCE is given. The code mirrors figure 4.10 and uses OWL terminology summarised in Table 4.3.

```
Input
$\sum Mbr(ins_t, Ctg_i.Lev_1) \mid t = 1..n, i = 1..5$
$\sum (ins_t, cha_q, vlu_q) \mid t \in \mathbb{N}_0, q = 1..n$
$\sum rlp_r(Ctg_i.Lev_j, Ctg_x Lev_y)$
$\sum rlp_r(ins_t, ins_u)$
Begin
For i = 1..5 Loop                                                    **Loop $Ctg_i$
        For t = 1..v Loop                                            **Loop $ins_t$
                Read $Mbr(ins_t, Ctg_i.Lev_1)$
                For j= 2..n Loop                                     **Loop $Lev_j$
                        Read $Ctg_i.Lev_j$
                        If $Ctg_i.Lev_j$ = Psn Then
                                Create subclass $Ctg_i.Lev_{J-1}$ of PERSON
                                For k = j-2..1 Loop                  **Loop1 $Ctg_i.Lev_j$
                                        Create subclass $Ctg_i Lev_k$ of $Ctg_i Lev_{k+1}$
                                        k = k-1
                                End Loop1 $Ctg_i.Lev_j$ subclass
                        Else If $Ctg_i.Lev_j$ = Ojt Then
                         Create subclass $Ctg_i.Lev_{J-2}$of OBJECT if it doesn't exist(if ∄)
                         For k = j-3..1 Loop                         **Loop2 $Ctg_i.Lev_j$
                                Create subclass $Ctg_i Lev_k$ of $Ctg_i Lev_{k+1}$ if ∄
                                k = k-1
                         End Loop2 $Ctg_i.Lev_j$ subclass
                        Else If $Ctg_i.Lev_j$ = Fld Then
                                If $Ctg_i.Lev_{J-1}$ = Health Then
                                        Create subclass $Ctg_i.Lev_{j-2}$ of HEALTH if ∄
                                        For k = j-3..1 Loop          **Loop3 $Ctg_i.Lev_j$
                                                Create subclass $Ctg_i Lev_k$ of $Ctg_i Lev_{k+1}$ if ∄
                                                k = k-1
                                        End Loop3 $Ctg_i.Lev_j$
                                Else If $Ctg_i.Lev_{j-1}$ = Education Then
                                        Create subclass $Ctg_i.Lev_{j-2}$ of EDUCATION
                                        For k = j-3..1 Loop          **Loop4 $Ctg_i.Lev_j$
                                            Create subclass $Ctg_i Lev_k$ of $Ctg_i Lev_{k+1}$ if ∄
                                            k = k-1
                                        End Loop4 $Ctg_i.Lev_j$
                                Else If $Ctg_i.Lev_{j-1}$ = Manufacturing Then
                                        Create subclass $Ctg_i.Lev_{j-2}$ of MANUFACTURING if ∄
```

There can be as many number of domains as necessary. The enumerated list of domains is thorough to cater for any possible environment.

```
                    For k = j-3..1 Loop                 **Loop5 $Ctg_i.Lev_j$
                        Create subclass $Ctg_iLev_k$ of $Ctg_iLev_{k+1}$ if $\nexists$
                        k = k-1
                    End Loop5 $Ctg_i.Lev_j$
                Else If $Ctg_i.Lev_j$ = Lcn Then
                    Create subclass $Ctg_i.Lev_{j-1}$ of LOCATION
                    For k = j-2..1 Loop                 **Loop6 $Ctg_i.Lev_j$
                        Create subclass $Ctg_iLev_k$ of $Ctg_iLev_{k+1}$ if $\nexists$
                        k = k-1
                    End Loop6 $Ctg_i.Lev_j$ subclass
                Else If $Ctg_i.Lev_j$ = Pfc Then
                    If $Ctg_i.Lev_{j-1}$ = Psn-specific-Pfc Then
                      Create subclass $Ctg_i.Lev_{j-2}$ of
                            PERSON_SPECIFIC_PREFERENCE if $\nexists$
                      For k=j-3..1 Loop                 **Loop7 $Ctg_i.Lev_j$
                            Create subclass $Ctg_iLev_k$ of $Ctg_iLev_{k+1}$ if $\nexists$
                     k = k-1
                      End Loop7 $Ctg_i.Lev_j$ subclass
                    Else If $Ctg_i.Lev_{j-1}$ = Ojt-specific-Pfc Then
                      Create subclass $Ctg_i.Lev_{j-2}$ of
                             OBJECT_SPECIFIC_PREFERENCE if $\nexists$
                      For k=j-3..1 Loop                 **Loop8 $Ctg_i.Lev_j$
                            Create subclass $Ctg_iLev_k$ of $Ctg_iLev_{k+1}$ if $\nexists$
                       k = k-1
                      End Loop8 $Ctg_i.Lev_j$ subclass
                    Else $Ctg_i.Lev_{j-1}$ = Lcn-specific-Pfc  Then
                          Create subclass $Ctg_i.Lev_{j-2}$ of
                             LOCATION_SPECIFIC_PREFERENCE if $\nexists$
                     For k =j-3..1 Loop                 **Loop9 $Ctg_i.Lev_j$
                            Create Subclass $Ctg_iLev_k$ of $Ctg_iLev_{k+1}$ if$\nexists$
                     k = k-1
                      End Loop9 $Ctg_i.Lev_j$
                Else
            j = j + 1
        End Loop $Lev_j$
            Assert Individual $ins_t$ into Subclass $Ctg_i.Lev_1$

            Select Subclass $Ctg_i.Lev_1$
            For q = 1..n Loop                       ** Loop $chr_q$
              Read ($ins_t$, $chr_q$, $vlu_q$)
              If $\exists(chr_q, Ctg_i.Lev_1) | chr_q$ is characteristic of $Ctg_i.Lev_1$ Then
                    Add($chr_q$, $vlu_q$) to Individual $ins_t$
                Else
                        Add Datatype Property $chr_q$ into Subclass $Ctg_i.Lev_1$
                        Add ($chr_q$, $vlu_q$) to Individual $ins_t$
                q = q +1
            End Loop datatype property
        t = t+1
    End Loop $ins_t$
    i = i +1
End Loop $Ctg_i$

For t = 1..w Loop                                              ** Loop individual
    Select individual $ins_t$ which is ∊ Ctgi.Lev1
    If Ctgi.Lev1 ⊑ PERSON Then
        For u = 1..x Loop                                      ** Loop1 $rlp_r$
            Select ($ins_u$, $Ctg_x Lev_1$)
            If $Ctg_x Lev_1$ ⊑FIELD Then
                Add Object Property isAssociatedWith($ins_t$, $ins_u$)
            Else If $Ctg_x Lev_1$ ⊑ PREFERENCE Then
                Add Object Property hasPreference($ins_t$, $ins_u$)
            Else If $Ctg_x Lev_1$ ⊑ LOCATION Then
                Add Object Property isIn($ins_t$, $ins_u$)
            u = u +1
        End Loop1 object property
    Else If $Ctg_i.Lev_1$ ⊑OBJECT Then
        For u = 1..x Loop                                       ** Loop2 $rlp_r$

            Select ($ins_u$, Ctgi.Lev1)
            If $Ctg_i.Lev_1$ ⊑ LOCATION Then
                Add Object Property isCurrentlyIn($ins_t$, $ins_u$)
            u = u + 1
        End Loop2 object property
    t = t + 1
End Loop individual
For i = 1..5 Loop                                              ** Loop  domain
    For  x = 1..5 Loop                                         ** Loop range
        For  r = 1..n Loop                                     ** Loop3 $rlp_r$
            Read $rlp_r$($Ctg_i.Lev_1$, $Ctg_x Lev_1$)
                If $Ctg_i.Lev_1$ ⊑ PERSON Then
                    If $rlp_r$ ∊{isAssociatedWith,isIn,hasPreference}Then
                      Nothing
                    Else
                      Create Object Property $rlp_r$($Ctg_i.Lev_1$,$Ctg_x Lev_1$)
```

```
                        Else If $Ctg_i.Lev_1 \sqsubseteq$ OBJECT Then
                            If $rlp_r$ = isCurrentlyIn Then
                                Nothing
                            Else
                                Create Object Property $rlp_r(Ctg_i.Lev_1,Ctg_xLev_1)$
                        Else If $Ctg_i.Lev_1 \sqsubseteq$ PREFERENCE Then
                            If $rlp_r \in$ {isAbout, isRelatedTo, isFor} Then
                                Nothing
                            Else
                                Create Object Property $rlp_r(Ctg_i.Lev_1,Ctg_xLev_1)$
                        Else
                            Create Object Property $rlp_r(Ctg_i.Lev_1, Ctg_xLev_1)$
                    Read $rlp_r (ins_t, ins_u)$
                    Add Object Property $rlp_r(ins_t, ins_u)$
                    r = r + 1
                End Loop3 $rlp_r$
                x = x + 1
        End Loop range
        i = i + 1
End Loop domain
```

4.2.3 The FCM and Delivering a Situation-specific Service in a PCE

As explained in the previous sections, the power of the proposed FCM is creating a semantically rich taxonomical structure, without which we cannot secure the delivery of a situation-specific service. However, the FCM cannot fully specify the exact computation of services to be delivered, because services are domain and situation specific. $PCE_{\Delta}T$, ensured by the FCM, is semantically rich that may trigger automatically reasoning for the delivered service. Therefore $PCE_{\Delta}T$ can have additional rules to trigger the situation-specific services.

We have explained in Chapter 2 that our SE solution in PCEs is influenced by the SW technologies, particularly OWL ontology. Considering that the W3C recommended ontology language OWL is based on DL, any rule used in the FCM must be based on DL. Given that DL is based on propositional logic, the propositions that define a DL argument must be found in the taxonomical structure $PCE_{\Delta}T$. For example, "if Margaret is in her room and she is laying on the floor, her caregiver must be alerted" is an argument. In this argument "Mrgaret is in her room", and "she(Margaret) in laying on the floor" are two propositions that make the premises of the argument. The conclusion of the argument is that “her caregiver must be alerted”. The semantics of these three propositions in the premises and the conclusion must be found in the PCE$_{\Delta}$T.

If the delivery of a service in a PCE depends on the conditions set for the activation of that service, then all the conditions for a particular situation must be present in the PCE$_{\Delta}$T, which entails that the taxonomical structure is correct. When the conditions are met, then the service expected for that situation will be delivered. Therefore, there is a need for a logical “argument”, in which premises are the conditions that have to be “true” for the “conclusion” of the argument to be also true.

The PCE$_{\Delta}$T that is created for a situation is fundamentally setting the premises of the “argument”. In other words, each instance $\mathtt{Mbr(ins_t,\ Ctg_i.Lev_j)}$, instance characteristic ($(\mathtt{Mbr(ins_t,\ Ctg_i.Lev_j),\ chr_q,\ vlu_q})$), and relationship between instances $\mathtt{rlp_r(Mbr(ins_t,\ Ctg_i.Lev_j),\ Mbr(ins_u,\ Ctg_xLev_y))}$ represented in situation-specific PCE$_{\Delta}$T is a proposition of the premises of the argument that defines the service for the situation, as the conclusion of the argument. The conclusion itself, whether a single proposition or a multiple, needs to be represented in the situation-specific PCE$_{\Delta}$T.

Although each proposition of the premises of the argument is represented inside the situation-specific PCE$_{\Delta}$T, and the PCE$_{\Delta}$T itself is a situation-specific model, the propositions are not collectively represented as a $Ctg_i.Lev_j$ representing the situation. The propositions represented in the PCE$_{\Delta}$T are provided by the PCE as situational information, but the “conclusion” has to be computed.

Within the PCE$_{\Delta}$T, the only inference that can take place is to infer an instance $\mathtt{Mbr\ (ins_t,\ LCtg_i.Lev_j)}$ to be an instance for all available values of j (Axiom 4). Therefore, considering the existence of multiple relationships between different $\mathtt{Ctg_i.Lev_j}$ imposed by the situation-specific “argument”, it is not possible to represent it in the generic taxonomical structure PCE$_{\Delta}$T, nor can it be defined in the extended taxonomical structure $PCE_{\Delta}T$. A situation-specific “rule” has to augment the $PCE_{\Delta}T$ to address the delivery of the service for the situation.

4.3 Summary

In this chapter, we have defined a formalised computational model for delivering services in PCEs. We have defined formal terms used in the taxonomical structure $PCE_{\Delta}T$ of the FCM. The FCM consists of $PCE_{\Delta}T$ for a particular PCE_{Δ}, and reasoning upon it in order to deliver a situation-specific service for the user of PCE. We have defined the FCM in boxes as technology independent steps, and in pseudo code influenced by SWT.

Chapter 5
Evaluation of the Proposed Model by Implementation

In this chapter, a real world scenario in healthcare domain is presented to demonstrate how the proposed formal computational model in PCE works. The model allows computations around any real world situation that takes place in a PCE. We will show through a scenario the creation of a situation PCE_{Δ} from which point in time the FCM creates the domain and situation-specific taxonomical structure $PCE_{\Delta}T$ to deliver a service to the user of the PCE for that particular PCE_{Δ}. The example scenario will illustrate the reasoning upon the taxonomical elements of the $PCE_{\Delta}T$ through the the FCM computations in order to deliver an expected service to the user for the particular PCE_{Δ}. We show, step by step, how the PCE_{Δ}can be modelled as a $PCE_{\Delta}T$ and how to compute the domain and situation-specific service. We exemplify the cyber-physical objects that exist and participate in the situation, who the user of the PCE in the particular PCE_{Δ} is, and the service(s) that are expected by the user to be delivered in that PCE_{Δ}.

5.1 Setting the Scene

To illustrate the FCM we need to set the scene. The rationale behind using healthcare domain for the running example is briefly explained first, followed by describing a particular PCE environment within the healthcare domain. A real life scenario that the example is based on is stated in the third subsection, to be followed by a recapitulation of the terms used in Chapter 4 To facilitate better connection between the rest of this chapter and the definitions and axioms provided in Chapter 4. Finally, the software architecture, part of which is the computational model, is addressed at the end of this section.

5.1.1 Healthcare Domain

The healthcare domain gives some of the most successful examples where the application of pervasive computing has materialised (Arnich et al., 2010) (Coronato, 2010) (Bardram, 2007) (Romero, et al., 2011), (Rolim, et al. 2011), (Varshne, 2007), (Zhang et al., 2011). The issue of having enormous number of devices with variable communication and computational power embedded into our everyday life environments, thus providing various types of PCEs, has become almost common in healthcare. In support of the technological advances, new software solutions also have been developed which support deliveries of health services, remote patient monitoring, remote management of diseases, self-care systems, and patient tele-monitoring. These have resulted the consequent claim that pervasive healthcare has become a scientific discipline (Bardram, 2008). This means that we are now able to turn our traditional general practitioner's surgeries, clinical interventions, patient monitoring and public health protection into e-health services, delivered at any time, in any place with the involvement of empowered patients interested in self-management of their health. In the following sections our software solution which would guarantee delivery of services is demonstrated through an example of a situation in a healthcare environment.

Despite the fact that security is a major issue in PCEs (Campbell et al 2002), people might be willing to compromise and give up considerable amount of their privacy for the sake of medical treatment (Bohn et al 2004). As the computing boundaries are extended through PCEs and include physical spaces, people who are interacting with the devices might not be aware of the amount of information about them that are being collected, exchanged and processed. Hence the issue of privacy and security arises. In health care domain, nevertheless, if provision of personal health information reassure people of their helath, and timely medical treatment when required, they might be more prepared to compromise their privacy. It is no wonder why by far the number of applications built for the healthcare domain is more than of that in any other domain, and why we have chosen this domain for our example scenario.

5.1.2 SeCH Environment

Self **C**are **H**ome, SeCH, is a physical environment, inhabitants of which are people who need constant care. These can be people who suffer from chronic illnesses and conditions which require medical attention, or senior citizens who require constant or occasional support. SeCH is equipped with sensors which detect the whereabouts of its residents and monitor their activities and physiological functions. When prospective residents are admitted to SeCH, they will be allocated a room and their belongings will be tagged. They will also receive specific items such as sensorised garments and hand-held communicators. Other cyber-physical objects such as clothes, furniture, appliances and similar will be available in SeCH. Constant monitoring facilities in SeCH were attractive to its residents; therefore they all agreed in advance that their health status and activities would be monitored. Residents also stated their preferences in terms of using and agreeing upon the facilities available in SeCH, which means that the software system which supports SeCH is less intrusive and more personalised to residents' own needs. Devices in SeCH are either embedded with computational capabilities, or are attached to actuators, which can trigger the delivery of SeCH services. SeCH is therefore, a PCE where sensors, devices, and actuators are connected through a wireless network to a gateway as shown in Figure 5.1. The system architecture of SeCH is beyond the scope of this thesis.

Examples of services delivered in SeCH are:

1) *Recommending* residents to wear their coat, to take due medication or to stop a current social/physical activity;
2) *Informing* residents of any changes to their daily routine due to the change of their circumstances, such as sending a new schedule for daily geriatric exercises or modifying their prescription medicine;

3) *Activating a device* within SeCH automatically, such as switching a heater on in a resident's room;
4) *Issuing an alarm* for the medical staff on duty because urgent medical attention is needed, after detecting anomalies in a resident's health status, such as a sudden fall in blood sugar level.

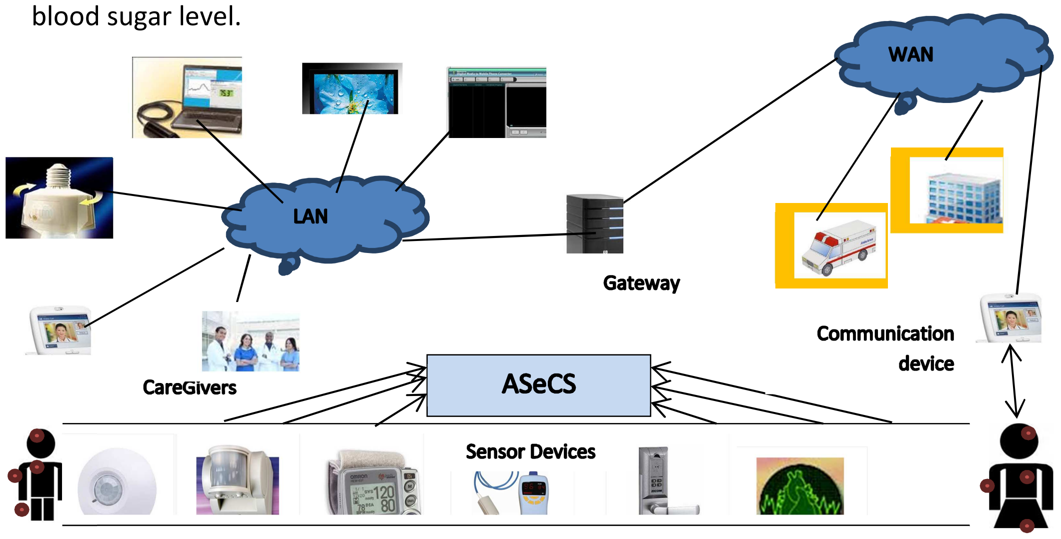


Figure 5.1 SeCH environment as an example of a PCE

We hasten to add that these four types of services are merely examples that we have chosen for the SeCH and therefore, reader should treat these as typical non-exclusive examples of services in a PCE. Different environments in the same domain may serve their users differently, hence services may vary for different PCEs. This is to say that, services delivered in any PCE are domain specific (Definition 13 , Axiom 9 in Chapter 4). This specificity indicates that these services cannot be represented in advance in any formal computational model for PCEs. This is why PCE$_\Delta$T can have additional rules to trigger the situation-specific services.

In SeCH, the formal model will provide a computational foundation that is capable of deciding what specific service of type 1) – 4) above, given a situation, should be delivered. In the following section a domain-specific scenario projecting a precise situation within SeCH environment demanding a particular service is explained. This scenario is the basis of this chapter's running example.

5.1.3 The Scenario

Margaret, John, Peter and Paul are residents of SeCH. Their morning routine starts with having a shower, followed by breakfast, and taking morning medicine. Then, they have free time to take part in a physical and social activity suitable for their health condition and preferences. Every day a balance exercise class for senior residents takes place in the ‘Function’ room. Attendance is compulsory for geriatric residents, but other residents attend if they wish. Margaret, usually takes part in the ‘walking-for-all’ activity in the adjacent park. This morning, however, she did not feel well and decided to go to her bedroom and read the daily paper. Margaret like all other residents of SeCH is being monitored so as to be attended to whenever there is a change in her health situation. The sensorised garment Margaret is wearing shows that she is feverish. When she was admitted in SeCH, Margaret has indicated that she would prefer to have her allocated heater in her room to be turned on, if it is off, when she feels cold. One of the contextual information that is produced in SeCH is to inform whether an allocated room is cold, normal or hot considering the body temperature of the person the room is allocated to, provided the person is currently inside the room. This sensor device indicates that the room is cold for Margaret.

As mentioned in the previous section, “recommending”, “informing”, “activating a device”, or “issuing an alarm” is the action taken in SeCH for any particular situation. Considering the new situation created when Margaret being feverish is detected, and given that she would expect the heater in her room to be turned on in the situation she is in, the expected service to be delivered is “Activating a device”. The computation that the formal model prepares the ground for, is therefore expected to trigger an actuator within SeCH that turns on the heater in Margaret’s room.

In the following section we summarise the formal model process from the moment a situation is created until the end of the computation that triggers an expected service for the user. Without going into intricacies, we encapsulate key terms introduced in the previous chapter in a paragraph or two to set the scene for more elaborations of the proposed model.

5.1.4 Recapitulation of the FCM Terminologies

The purpose of this section is recapitulation of the terms used in Chapter 4, vis-à-vis the FCM, to facilitate the description of how the domain and situation-specific PCE_{Δ}, detailed above, is translated into a detailed specification of the situation in $PCE_{\Delta}T$. The $PCE_{\Delta}T$ will lend itself to the reasoning required to deliver a service to the user of the PCE; i.e. trigger an actuator within SeCH that turns on the heater in Margaret's room.

The detection of PCE_{Δ} in the PCE takes place when there is a change in the SeCH environment, in our case Margaret being feverish, is outside the scope of this thesis. This thesis is about FCM and the computations it does once a PCE receives interpreted situational information. The creation of $PCE_{\Delta}T$ and reasoning upon its taxonomical elements for a particular PCE_{Δ} is what this thesis is after, thus the detection of PCE_{Δ} is not within the remit of the thesis. When a PCE_{Δ} is detected we know exactly which and in what locations cyber-physical objects participate in the PCE_{Δ}, who and where the user of the PCE is, what the preferences of the user are, and which services that are expected by the user will be delivered in that PCE_{Δ}. We also know some domain-specific information such as Margaret's general health. Knowing all these at this particular moment triggers the existence of a particular situation in the PCE, PCE_{Δ} (Chapter 4, Definition 1).

A participant in PCE_{Δ}, such as Margaret, is referred to as instance $\texttt{ins}_t$. Instances are members of categories(Chapter 4, Definition 5) and represented as $\texttt{Mbr(ins}_t\texttt{, Ctg}_i\texttt{.Lev}_j\texttt{)}$ where $\texttt{Ctg}_i\texttt{.Lev}_j$ represents the category that $\texttt{ins}_t$ is a member of (Chapter 4, Definition 7). For example, in `Mbr(margaret,Resident)`, `Resident` is a category represented as $\texttt{Psn.Lev}_1$ or in `Mbr(feverish, General Health)` `General Health` is a category represented as $\texttt{Fld.Lev}_1$. The category membership $\texttt{Mbr(ins}_t\texttt{, Ctg}_i\texttt{.Lev}_j\texttt{)}$of real world participants $\texttt{ins}_t$ of the PCE_{Δ} necessitates a $PCE_{\Delta}T$ (Chapter 4, Definition 8) in which all $\texttt{Ctg}_i\texttt{.Lev}_j$ are modelled.

Therefore, for the given scenario and its particular PCE_{Δ} a $PCE_{\Delta}T$ is abstracted to be reasoned upon its taxonomical elements in order to deliver an expected service; i.e.

triggering an actuator within SeCH that turns on the heater in Margaret's room. Characteristics $\texttt{Chq}_r$ and its value $\texttt{vlu}_r$ that describe category memberships (Chapter 4, Definition 16), and relationships $\texttt{rlp}_q$ between taxonomical elements (Chapter 4, Definition 17) are other aspects of $PCE_{\Delta}T$ contributing towards the computation of PCE_{Δ}. The $\texttt{Chq}_r$ is used in $\texttt{(Mbr(ins}_t\texttt{, Ctg}_i\texttt{.Lev}_j\texttt{),chq}_r\texttt{,vlu}_r\texttt{)}$ format, for example, `(Mbr(margaret, Resident),“gender”,“female”)` which shows `margaret` is a member of the category “`Resident`” and has a gender characteristic which has the value `female`, or `(Mbr(heater152,Heater),status,“off”)`which shows `heater152` is a member of the category `Heater` and has `status` characteristic which has the value `off`.

The FCM computation to achieve $PCE_{\Delta}T$ is essential for the FCM to reason upon its elements to deliver a service to the user of the PCE_{Δ}. Computations from the moment situational information of a PCE_{Δ} is received by the PCE was shown in Figure 4.1 in Chapter 4. Here, in Figure 5.2 we present the application of the diagram to the example scenario. a leading to delivery of a service prepares the foundation for the computational model which is instrumental to deciding what domain and situation specific service(s) is (are) to be delivered to the user of the situation created by a change in the PCE environment as shown in Figure 5.2.

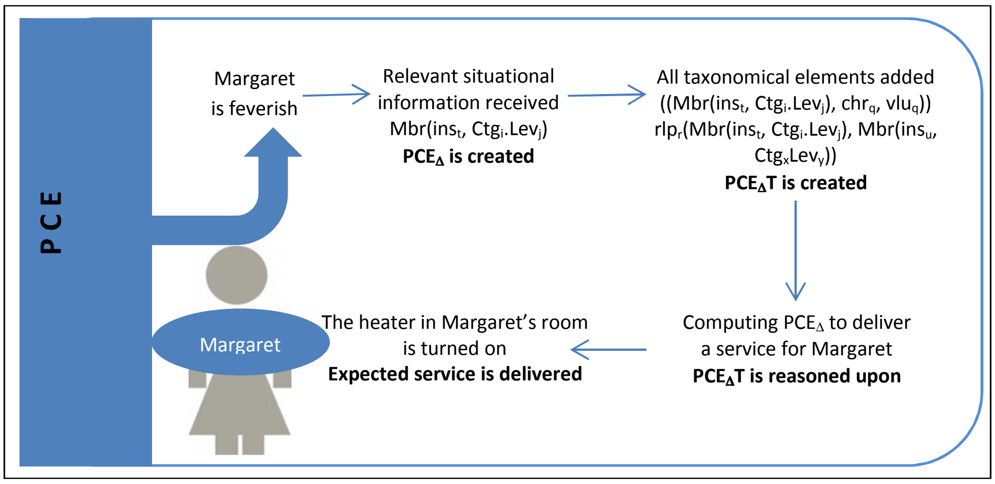

PCE
Margaret is feverish
Relevant situational information received
Mbr(ins$_t$, Ctg$_i$.Lev$_j$)
PCE$_\Delta$ is created
All taxonomical elements added
((Mbr(ins$_t$, Ctg$_i$.Lev$_j$), chr$_q$, vlu$_q$))
rlp$_r$(Mbr(ins$_t$, Ctg$_i$.Lev$_j$), Mbr(ins$_u$, Ctg$_x$Lev$_y$))
PCE$_\Delta$T is created
Computing PCE$_\Delta$ to deliver a service for Margaret
PCE$_\Delta$T is reasoned upon
The heater in Margaret's room is turned on
Expected service is delivered
Margaret

Figure 5.2: FCM computations for the example scenario PCE_{Δ}

5.1.5 ASeCS Software Architecture

Cyber-physical objects of SeCH that participate in Δ create particular "situational information" PCE_{Δ} which is abstracted to $PCE_{\Delta}T$. As stressed earlier, how these objects are networked and managed, and in general the mechanism of the system architecture is not the concern of this thesis. However, the cyber-physical objects supply software applications with domain and situation-specific semantics which secures the delivery of services for a particular PCE_{Δ} within SeCH.

Although the software architecture is not domain specific and is reusable for different environments, in our running example we refer to this architecture as **A**ssistive **Se**lf **C**are **S**ystem (ASeCS) just because the scenario takes place in a care home environment, SeCH.

Figure 5.2 illustrates the ASeCS software architecture. It is component based and layered and each layer has its own purpose and role in the overall ASeCH architectural style.

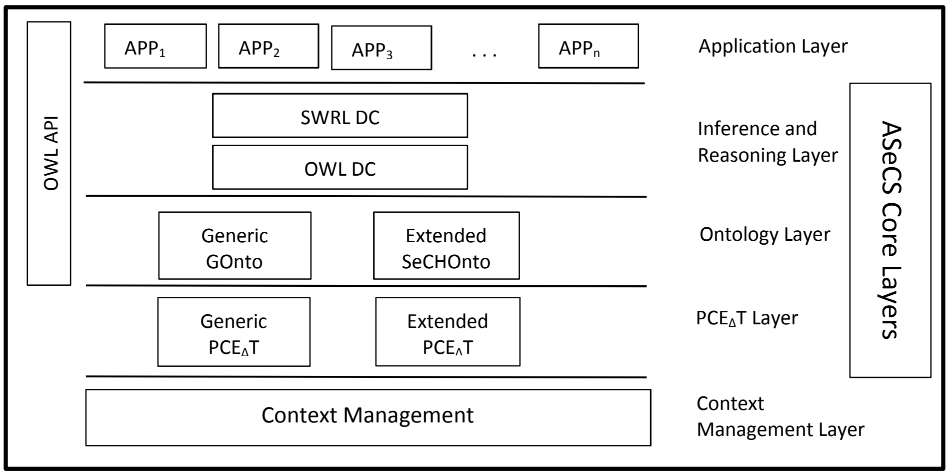


Figure 5.3: ASeCS software architecture

Context Management Layer (CML) and *Application Layer (AL)* have specific roles compared to other ASeCS layers. The ASeCS software architecture hosts software applications that primarily support SeCH and trigger the delivery of its services. Therefore a set of various software applications and their interfaces are stored within the AL. They are all are able to communicate with software components stored in the lower ASeCS layers and interpret and manipulate any type of input or user interaction we may have in various situations in SeCH.

However, the CML has a completely different role. The availability of information about PCE_{Δ} or "situational information" is essential to provide timely and appropriate service to the user. Therefore, acquiring contextual data from the environment is an indispensable part of any PCE (Schmidt et al., 1999), (Henricksen and Indulska, 2006), (Sabagh et al., 2011). Sensorised garments, tagged heater, persistent data repositories, and software programs which integrate various devices into a PCE, are examples of cyber-physical devices in SeCH which provide some form of contextual data to the ASeCS architecture. For PCE_{Δ} in SeCH, information on who the user is (Margaret), where she is, is the room she is in cold given her body temperature, is the heater in her room on or off, would she prefer the heater in her room to be turned on if it is off when she is feverish, are some of the "situational information" examples necessary to acquire in order to define the situation in SeCH to deliver a service to fulfil Margaret's expectation. Consequently, such contextual data have to be managed, i.e. captured and interpreted (Day 2000), (Day, 2001), (Strang and Linhoff-Popien, 2004), (Bettini et al., 2010) and therefore the CML stores, represents and manages data received from the Cyber-Physical Objects and **prepares the "situational information"** for the ASeCS upper layers. This is in line with many other similar solutions which require interpreting the meaning of the collected contextual data, and which has been exercised in context aware software applications for more than a decade ((Gu et al., 2004), (Davis et al., 2005), (Gua et al., 2005), (Ellenber et al., 2011), (Sang et al., 2003), (Wu et al., 2007). For example, the detection whether Margaret is "feverish" or not is the responsibility of the CML, and should be given to the upper ASeCS layers as a part of particular **"situational information".** Consequently, the CML makes sure that sensed data is qualified with some significant semantics for further computation; i.e. interpreted.

Computationally significant semantics provided by the CML is managed in the ASeCS core layers, which comprises $PCE_{\Delta}T$, Ontology, and Inference and Reasoning Layers as depicted in Figure 5.3.

It is important to note that the ASeCS core layers are essential for exploiting the **"situational information"** generated by the CML and delivering services within the SeCH through the applications App_n.

The $PCE_\Delta T$ layer stores taxonomies of PCE_Δ which may be generic, i.e. may contain enough taxonomical elements which could be used for describing any situation PCE_Δ in SeCH. However, possible extensions of the generic situational information in PCE_Δ may be needed for two important reasons: the situational information generated by the CML may require to create more $PCE_\Delta T$ elements in order to secure the delivery of services in SeCH and the generic $PCE_\Delta T$ might not be sufficient to accommodate the specificity of the semantics in this particular domain (SeCH). We illustrate the former in the paragraph below.

All real world instances that participate in Δ and create PCE_Δ are accommodated in the taxonomical structure $PCE_\Delta T$. The $PCE_\Delta T$ layer is responsible for arranging and organising all detected $\texttt{Mbr}(\texttt{ins}_t,\texttt{Ctg}_i.\texttt{Lev}_j)$ participating in the PCE_Δ. In Chapter 4 section 4.3 we have axiomatised (Axiom 6, 8, 10, 12, and 14) the category membership such that for $\texttt{Mbr}(x,y)$ where x is a real world instance, y is $\texttt{Psn}$, $\texttt{Ojt}$, $\texttt{Fld}$, $\texttt{Pfc}$, or $\texttt{Lcn}$. The Generic $PCE_\Delta T$ of this layer allows representation of general instances. For example, any real person in the SeCH without any specificity could be presented as an instance of $\texttt{Psn}$, i.e $\texttt{Mbr}(x,\texttt{Psn})$. However, Margaret is a specific inhabitant of SeCH. She is a $\texttt{Resident}$, where the scenario says "Margaret, … are residents of SeCH". This requires the Generic $PCE_\Delta T$ to be extended to allow specific $\texttt{Ctg}_i.\texttt{Lev}_j$ Resident as an extension to $\texttt{Psn}$.

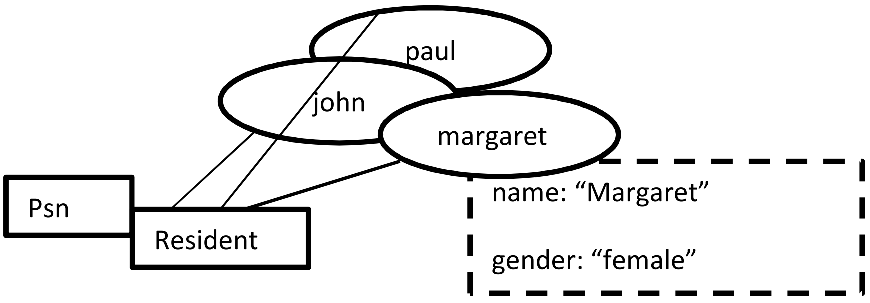


Figure 5.4: Part of PCE_Δ showing instance "$\texttt{margaret}$" and its category "$\texttt{Resident}$"

Figure 5.4 shows (`Mbr(margaret,Resident)`as an extension of `Mbr(margaret,Psn).`

The Ontology Layer has a similar role to the *The PCE$_\Delta$T Layer.* The only difference is that the taxonomy of the situational Information from the *PCE$_\Delta$T Layer* is transferred into OWL classes and properties. Similar to the PCE$_\Delta$T Layer the Ontology Layer hosts a generic OWL ontology. However, the generic ontology GOnto can be extended ONLY by adding the specificity of SeCH and therefore the extended ontology must be called SeCHOnto. The generic ontology GOnto represents "bare-minimum" of OWL concepts applicable to all PCEs depicted in Figure 4.9 in Chapter 4. The development of GOnto was explained thoroughly in Chapter 4 and summarised in Figure 4.8 where a generic taxonomical model for any PCE$_\Delta$ is given. Extension of GOnto to achieve SeCHOnto is discussed in the subsequent sections.

The inference and Reasoning Layer provides an essential functionality for delivering services in SeCH. OWL ontologies are based on Description Logic and therefore inference on the concepts within GOnto or SeCHOnto is feasible using DL reasoning mechanism of OWL. However, when there is a need for reasoning about a complex semantic involving several concepts, OWL falls short and SWRL rules, which are also based on DL, have to be used. This is why the Ontology Layer is complemented with the Inference/Reasoning Layer to cover for the reasoning aspect of the ASeCS architecture.

Finally, the applications from the Application Layer are able to communicate with Ontology and Inference/Reasoning layers, through OWL-API. It enables that software applications in ASeCS "know" SeCH inhabitants' precise location, their current activities, and present physiological vital sign measurements, and they can "react" in order to assist residents in their everyday lives. In other words, "reacting" means delivery of personalised service(s) to the user of the PCE$_\Delta$ in SeCH.

5.2 Illustration of the FCM in a PCE

The previous section has set the scene for a specific user in a particular situation Δ in SeCH. However, as stated in Chapter 4, FCM creates abstraction of situational information received by the PCE. We start this section with explaining what and how the situational information for the creation of $PCE_{\Delta}T$ is established. Then, the parts of the FCM as explained in section 4.2 in the previous chapter will be illustrated.

The illustration of the FCM requires deployment of technologies. The background research in section 2.4 and the overview of the FCM in section 3.4.3 has highlighted that the use of SWTs particularly SWRL enabled OWL ontology is the way forward, if we desire to create a new era of SE solutions based on the semantics and understanding of our computational environments across domains. That said, by knowing that we will use the SWT stack, FCM and situational information should be expressed in vocabulary of and following terms of the opted-for technology to secure the implementations of the FCM. Therefore, considering the ASeCS architecture (Figure 5.2) in which CML is responsible for the provision of interpreted contextual data , we need to use competency question (CQ) which is another OWL ontology terminology to establish situational information PCE_{Δ} for a particular situation Δ.

Identification of interpreted contextual data to achieve PCE_{Δ} is not within the remit of FCM and it cannot work without that. In the following section we briefly explain the role of CQs and how it leads to the creation of $PCE_{\Delta}T$. However, reader is reminded that the mechanism through which “words” of the CQ are automatically translated into machine understandable terms for FCM to work, is outside the scope of this thesis.

5.2.1 Formulating the Competency Question (CQ)

The previous section has set the scene for a specific user in a particular situation Δ in SeCH. Designing a domain-specific computational model pertinent to the situation requires clarification and verification of data the computational model needs. Some, for example (Chen et al., 2004b) use ‘use case scenario’ to verify the appropriateness of their ontological vocabularies structure. However, we use ‘competency question’ as used in the realm of Semantic Web and explained in Chapter 2 to verify input data to the computational model.

We have explained in Chapter 2 what the role of CQ is in realization of Semantic Web technologies. In SeCH, we associate the role of CQ with three issues

1) the detection of real world instances and the creation of $PCE_{\Delta}T$ ($PCE_{\Delta}T$ Layer)
2) a selection and extension of GOnto classes important for creating SeCHOnto (Ontology Layer), and
3) the reasoning process we perform upon SeCHOnto concepts in order to deliver a service.

Therefore, a correct CQ is of upmost importance for our process of extending GOnto to SeCHOnto. We show how our CQ helps to perform 1), 2) and 3) above.

The scenario from 5.3 gives a clear CQ:

CQ: *Which device(s) should be activated when there are some feverish individuals inside their assigned room, who would prefer the heater in their room (if it is cold) to be turned on (if it is off).*

The above CQ can be rephrased as follows to illustrate necessary information needed to be collected for a particular PCE_{Δ}. As any PCE_{Δ} is focused on one and only one user (Chapter 4, Axiom 6), in the re-phrased format, the CQ will reflect a real PCE_{Δ} in SeCH.

Re-phrased CQ: *Margaret is a resident in SeCH. (As soon as a new PCE_{Δ} is created, information such as Margaret’s gender, her name and her assigned room in SeCH become available at once.) SeCH is a care home in the UK and therefore it follows the UK health policy and procedures. It is detected that Margaret is feverish. Her current location is “Room101” which is a private location (bedroom) inside SeCH. This location (Room101) is “cold” now considering Margaret’s body temperature. A heater, “heater152”, is present in Room101. The status of this heater, which belongs to Margaret, is “off”. Margaret has indicated when she was admitted to SeCH some object preferences. One of these preferences is that she prefers the heater in her room to be turned on, if it is off, when she is feverish.*

Therefore, the new situation (Margaret beeing feverish) creates a new PCE_{Δ}, as shown in Figure 5.2. As soon as this piece of information (Margaret is feverish) is sensed, a collection of other semantically significant pieces of information relevant to the CQ becomes available to the computational model. We must give more information about a particular PCE$_{\Delta}$ in SeCH. This is essential for two reasons. We must know which exact inputs we give to the ASeCS application, and which output we expect from it (which is usually the result of delivering a service).

	Competency Question Segments
A	*Margaret is a resident in SeCH.*
B	*Margaret's name, gender and her assigned room are examples of available information from*
C	*SeCH is in the UK and therefore it follows the UK health policy and procedures.*
D	*SeCH is a care home.*
E	*Margaret is feverish.*
F	*Her current location is "Room 101"*
G	*which is a private physical location (bedroom) inside SeCH.*
H	*This location (Room101) is "cold" (considering Margaret's body temperature).*
I	*A heater "heater152" is present in Room101.*
J	*The status of this heater is "off".*
K	*This heater belongs to Margaret.*
L	*Margaret has indicated when she was admitted to SeCH some object preferences.*
M	*One of these preferences is the heater in her room to be turned on, if it is off, when she is*

Table 5.1, Segments of the running example competency question

The breakdown of the CQ into 13 segments (A to M) is shown in Table 5.1. The purpose of this breakdown is to demonstrate how each piece of information, a word, a phrase or a complete sentence in the CQ can be mapped to PCE$_{\Delta}$T elements.

The following section will show how the situational information PCE$_{\Delta}$ of the example scenario received by the PCE is computed by the FCM to create PCE$_{\Delta}$T and to reason upon it to deliver a service to the user (Margaret).

5.2.2 Illustration of FCM Loops and Steps

In this section steps explained in section 4.2.1 about the development of FCM is explained. In the following subsections we will show how the words that appear within the CQ (section 5.2.1) - in the form of nouns, verbs, proverbs and adjectives - found their space within PCE$_{\Delta}$T.

How PCE_{Δ} is abstracted to generic taxonomical structure, how generic taxonomical structure is extended to $PCE_{\Delta}T$, and how $PCE_{\Delta}T$ paves the way to reason to deliver a situation-specific service, is also explained in this section.

5.2.2.1 Illustration of Creation of $ctg_i.Lev_j$, Insertion of ins_t, and addition of chr_q,vlu_q

Here, we would like to go through all the statements regarding the creation of a new $ctg_i.Lev_j$, $Mbr(ins_t,ctg_i.Lev_j)$, and$(Mbr(ins_t,Ctg_i.Lev_j), chr_q,vlu_q)$listed in Table5.1.

Statement A: *Margaret is a resident in SeCH.*

According to Definition 11-15 detailed in section 4.3 (Chapter 4) the only occurrences of root category are Psn, Fld, Ojt, Lcn, and Pfc. Every real world instance, ins_t, is therefore an instance of one of these root categories. Considering that, ins_t is identified along its $ctg_i.Lev_j$ (Definition 7), when the $ctg_i.Lev_j$ of an instance is given in a CQ, for example "resident" in the first sentence of the CQ in Table 5.1, its root category is assumed. Therefore, for the sentence *"Margaret is a resident in SeCH"* we know that Resident is of root category Psn and given Axiom 2, we can conclude that Resident is $Psn.Lev_1$. The ins_t of $Psn.Lev_1$ according to the statement is Margaret with capital "M". There is a subtle concern here. "Margaret" is actually the "name" of the real world person and it is not the instance itself. To differentiate between these we refer to the instance ins_t as margaret with lowercase "m" and the name of the person as Margaret with capital letter "M". The first sentence of the CQ should have read "*margaret is a resident in SeCH"*, but merely because of linguistic concerns we left it with capital M. The contextual representation of this, based on Definition 7, is therefore Mbr(margaret,Resident).To summarise, for the statement *Margaret is a resident in SeCH,* the following is represented in the taxonomical structure $PCE_{\Delta}T$:

Mbr(margaret,Resident) where Resident is $Psn.Lev_1$.

Statement B: *Margaret's name, gender, and her assigned room are some examples of available information from CML.*

For every $Mbr(ins_t,Ctg_i.Lev_j)$ in the $PCE_{\Delta}T$ all available instance characteristics $\texttt{chr}_q$ are represented according to Definition 16. Therefore for `Mbr(margaret,Resident)`the "`name`","`gender`", and "`assignedRoom`" characteristics are represented in the PCE$_\Delta$T as :

```
(Mbr(margaret,Resident),name,"Margaret")
(Mbr(margaret,Resident),gender,"female")
(Mbr(margaret,Resident),assignedRoom,"Room101")
```

Statement A and B are depicted in blue in Figure 5.5. "`margaret`", "`john`" and "`paul`" are different ins$_t$ of `Resident`. "`margaret`" oval is bolded to indicate the $\texttt{ins}_t$ of concern for the particular PCE$_\Delta$. Characteristics $\texttt{chr}_q$ of "`margaret`" and their values are shown inside the broken-line-border box.

Statement C: *SeCH is in the UK and therefore it follows the UK health policy and procedures.*

The provision of services in any PCE according to Definition 2 and 3 is domain-specific. The specificity of any domain is represented in the PCE$_\Delta$T under the `Fld` root category. Statement "C" indicates the environment is within the health domain. The representations of the "Health" $\texttt{Ctg}_i\texttt{.Lev}_j$ and an $\texttt{ins}_t$ of it in the PCE$_\Delta$T similar to that of statement A are:

`Mbr(UK-health,Health)` where `Health is` $\texttt{Fld.Lev}_3$.

We would like to attract the reader's attention to the level that `Health` belongs to in the PCE$_\Delta$T. The $\texttt{Ctg}_i\texttt{.Lev}_j$ for `Resident` in statement "A" was defined as $\texttt{Psn.Lev}_1$ as the precision of abstraction required for instance "`margaret`". However, different precisions are required for different $\texttt{ins}_t$ of `Fld` occurrence. The higher the precision required the lower the level of $\texttt{Ctg}_i\texttt{.Lev}_j$. Defining `Health` at level 3 ($\texttt{Fld.Lev}_3$) indicates that for the particular PCE$_\Delta$ specified in the CQ there are two more precise levels ($\texttt{Fld.Lev}_2$ and $\texttt{Fld.Lev}_1$)which are more specific than the `Health`. Lack of any characteristics of `UK-health` in the CQ is a good indication that it belongs to a high-level of abstraction, i.e. no detailed precision required.

Statement D: *SeCH is a care home.*

Similar to `UK-health` instance, no characteristics are given for SeCH instance in the CQ. The only extra feature available for SeCH is that it is a subset of `UK-`

health. This explains why Care Home is $Fld.Lev_2$. These representations in the $PCE_{\Delta}T$ is:

Mbr(SeCH,Care Home) where Care Home is $Fld.Lev_2$.

Statement E: *Margaret is feverish.*

As shown in Figure 5.5 real world instances "feverish", "normal", "critical" are abstracted as "General Health". The ins_t which is available as a result of the particular situation PCE_{Δ} of the CQ is "feverish". As there are some characteristics available for "feverish", General Health ought to be at the lowest level of abstraction (high precision). This representations in the $PCE_{\Delta}T$ is:

Mbr(feverish, General Health) where General Health is $Fld.Lev_1$.

In Figure 5.5, Fld and all its $Ctg_i.Lev_j$ are shown in amber.

Statement F and G: Her current location is "Room 101 which is a private physical location (bedroom) inside SeCH.

The same explanation about the formation of different $Ctg_i.Lev_j$ of Fld root category is applicable here for Lcn. All $Ctg_i.Lev_j$, of Lcn with some instances are shown in green in Figure 5.5; their representations in the $PCE_{\Delta}T$ are:

Mbr(insideSeCH,PhysicalLocation) where Physical Location is $Lcn.Lev_2$.
Mbr(room101,Private) where Private is $Lcn.Lev_1$.

Statement H: *This location (Room101) is "cold" considering Margaret's body temperature.*

The $Ctg_i.Lev_j$ of Private is $Lcn.Lev_1$ which suggest that there are some characteristics available for the instance room101 of this $Ctg_i.Lev_j$. The implicit indication of the CQ is that the location or room temperature is cold. Therefore, the roomTemperature characteristic of the instance needs to be represented in the $PCE_{\Delta}T$:

(Mbr(room101, Private),roomTemperature,"cold").

Statement I and J: *A heater "heater152" is present in Room101. The status of this heater is "off".*

These statements are quite similar to statements A and B except that here the $\texttt{ins}_t$ is of root category `Ojt`. Considering that even the structure of all $\texttt{Ctg}_i.\texttt{Lev}_j$ of both occurences of `Psn` and `Ojt` are similar, there is no need of any further explanation for this part of the CQ. Their representations in the $PCE_{\Delta}T$ are:

```
Mbr(heater152,Heater),
(Mbr(heater152, Heater),type,“heater”),
(Mbr(heater152, Heater),status,“off”).
```

Statement L: *Margaret has indicated when she was admitted to SeCH some object preferences.*

Some of the real world instances that can be abstracted to a $\texttt{Ctg}_i.\texttt{Lev}_j$ of the `Pfc` root category are shown in Figure 5.5 in purple. The $\texttt{Ctg}_i.\texttt{Lev}_j$ is named `Ojt-specific-Pfc`. In this particular PCE_{Δ} the bolded `heaterPreference` is the available instance of this $\texttt{Ctg}_i.\texttt{Lev}_j$. The representations in the $PCE_{\Delta}T$ are `Mbr(heaterPreference,Ojt-specific-Pfc)` where `Ojt-specific-Pfc is` $\texttt{Pfc.Lev}_1$.

Statement M: *One of these preferences is the heater in her room to be turned on, if it is off, when she is feverish.*

As we have seen earlier, since `heaterPreference` is a member of a leaf $\texttt{Ctg}_i.\texttt{Lev}_j$ (that is $\texttt{Pfc.Lev}_1$) some characteristics of the ins_t should also be available. These characteristics are presented in the $PCE_{\Delta}T$ as:

```
(Mbr(heaterPreference,Ojt-specific-Pfc),objectType,“heater”)
(Mbr(heaterPreference,Ojt-specific-Pfc),objectNewStatus,“on”).
```

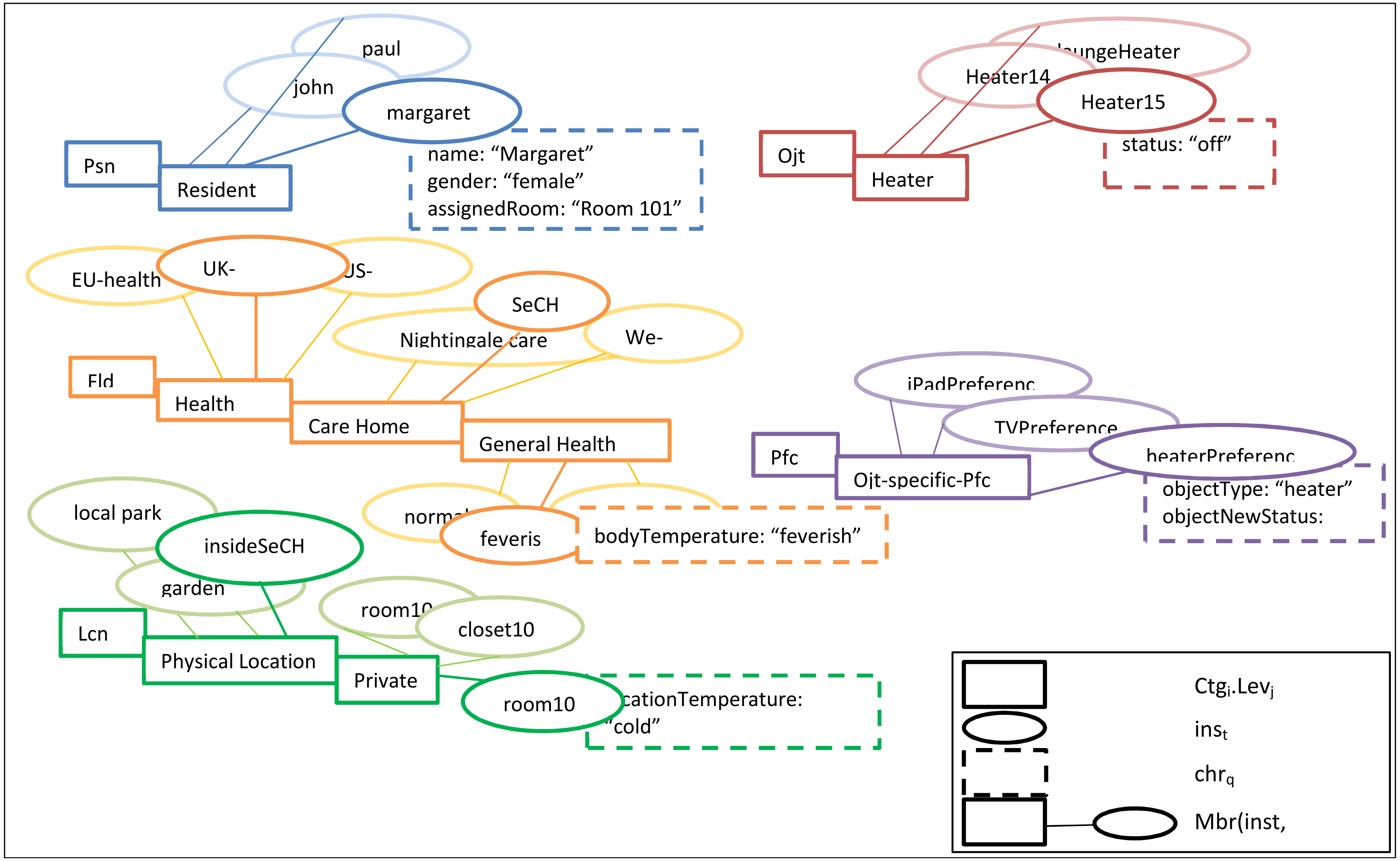


Figure 5.5: PCE$_\Delta$T representation of instances and `Ctgi.Levj` characteristics of the scenario CQ

5.2.2.2 Illustration of Generic Relationships rlp$_r$,

Here, we would like to go through all the statements regarding the creation of generic relationships `rlp`$_r$`(Mbr(ins`$_t$`,Ctg`$_i$`.Lev`$_j$`),Mbr(ins`$_u$`,Ctg`$_x$`Lev`$_y$`))` listed in Table5.1.

Statement E: *Margaret is feverish.*

We have already created `Mbr(feverish,General Health)` in the previous section. However, considering that there is always a relationship "`isAssociatedWith`" between `Mbr(ins`$_t$`,Psn.Lev`$_j$`)` and `Mbr(ins`$_t$`,Fld.Lev`$_k$`)` according to Axiom 16, the following is also established: `isAssociatedWith(Mbr(margaret,Resident),Mbr(feverish, General Health)).`

Statement F and G: Her current location is "Room 101 which is a private physical location (bedroom) inside SeCH.

We have already created `Mbr(insideSeCH,PhysicalLocation)`, `Mbr(room101,Private)` and `Mbr(room101,Private)` in the PCE$_\Delta$T. However, once `Mbr(room101,Private)`is represented in the PCE$_\Delta$T then a relationship between `Mbr(room101,Private)`and a previously represented `Mbr(margaret,Resident)`can be established according to Definition 17 as `isIn(Mbr(margaret,Resident),Mbr(room101,Private)).`

Statement I and J: *A heater "heater152" is present in Room101. The status of this heater is "off".*

We have already created `Mbr(heater152,Heater)`and added `(Mbr(heater152, Heater),type,"heater"), (Mbr(heater152, Heater),status,"off")` in the PCE$_\Delta$T. However according to Axiom 18 we need to add a relationship FCM prescribe between the object and the location it is in. This is illustrated as `isCurrentlyIn(Mbr(heater152,Heater),Mbr(room101,Private))`

Statement L: *Margaret has indicated when she was admitted to SeCH some object preferences.*

We have already created `Mbr(heaterPreference,Ojt-specific-Pfc).`

Given Axiom 20, there is always a relationship hasPreference between `Mbr(ins`$_t$`, Psn.Lev`$_j$`)` and `Mbr(ins`$_u$`,Pfc.Lev`$_k$`)`, therefore we also have the following in the taxonomical structure:

```
hasPreference(Mbr(margaret,Resident),Mbr(heaterPreferenc,
Ojt-specific-Pfc)).
```

Statement M: *One of these preferences is the heater in her room to be turned on, if it is off, when she is feverish.*

We have already added the following two characteristic values,

```
(Mbr(heaterPreference,Ojt-specific-Pfc),objectType,“heater”)
(Mbr(heaterPreference,Ojt-specific-Pfc),objectNewStatus,“on”).
```

However, given Axiom 22, there is always a relationship `isRelatedTo` between `Mbr(ins`$_t$`, Ojt.Lev`$_j$`)` and `Mbr(ins`$_u$`, Pfc.Lev`$_k$`)`, therefore we also have to add the following in the taxonomical structure:

```
isRelatedTo(Mbr(heaterPreference,Ojt-specific-
Pfc),Mbr(heater152,Heater)).
```

5.2.2.3 Illustration of Extended Relationships `rlp`$_r$,

Here, we would like to go through any statement regarding the creation of new relationships `rlp`$_r$`(Mbr(ins`$_t$`,Ctg`$_i$`.Lev`$_j$`),Mbr(ins`$_u$`,Ctg`$_x$`Lev`$_y$`))` listed in Table5.1 that extends PCE$_\Delta$T .

Statement K: *This heater belongs to Margaret.*

We have already created `Mbr(heater152,Heater)`and `Mbr(margaret,Resident)`in section 5.2.2.1. The relationship *“belongs to” is* not currently available in the taxonomical structure and therefore a new rlp$_r$ has to be added to the PCE$_\Delta$T as an extension. This new relationship is

`belongsTo(Mbr(ins`$_t$`,Heater),Mbr(ins`$_u$`,Resident)).` Once the relationship is available the following will be added according to Definition 17:

```
belongsTo(Mbr(heater152,Heater),Mbr(margaret,Resident))
```

5.2.2.4 Summarising all PCE$_\Delta$T Elements for the Running Scenario

The content of the PCE$_\Delta$T as explained above in section 5.2.2.1, 5.2.2.2, and 5.2.2.3 is summarised in Table 5.2 in the order FCM prescribes and were followed above.

	Contextual Data
1	`Mbr(margaret,Resident)`
2	`(Mbr(margaret,Resident),name,“Margaret”)`
3	`(Mbr(margaret,Resident),gender,“female”)`
4	`(Mbr(margaret,Resident),assignedRoom,“Room101”)`
5	`Mbr(UK-health, Health)`
6	`Mbr(SeCH, Care Home)`
7	`Mbr(feverish,General Health)`
8	`Mbr(heater152,Heater)`
9	`(Mbr(heater152,Heater),status,“off”)`
10	`(Mbr(heater152,Heater),type,“heater”)`
11	`Mbr(insideSeCH,Physical Location)`
12	`Mbr(room101,Private)`
13	`Mbr(heaterPreference,Ojt-specific-Pfc)`
14	`(Mbr(heaterPreference,Ojt-specific-Pfc),objectType, “heater”)`
15	`(Mbr(heaterPreference,Ojt-specific-Pfc), objectNewStatus,“on”)`
16	`(Mbr(room101, Private),roomTemperature,“cold”)`
17	`isAssociatedWith(Mbr(margaret,Resident),Mbr(feverish,General Health))`
18	`isIn(Mbr(margaret,Resident),Mbr(room101,Private))`
19	`isCurrentlyIn(Mbr(heater152,Heater), Mbr(room101,Private))`
20	`hasPreference(Mbr(margaret,Resident), Mbr(heaterPreference,Ojt-specific-Pfc))`
21	`isRelatedTo(Mbr(heaterPreference,Ojt-specific-Pfc),Mbr(heater152, Heater))`
22	`belongsTo(Mbr(heater152,Heater), Mbr(margaret,Resident))`

Table 5.2 Content of the PCE$_\Delta$T of the running example

In the previous section we explained the derivation of different elements of PCE$_\Delta$T from statements of Table 5.1, whereas in Table 5.2 the content of PCE$_\Delta$T for the CQ of Table 5.1 is listed. For ease of reference, the summary of mapping between these two tables, i.e. CQ statements to the corresponding FCM representation, is also provide in Table 5.3.

	Mapping of Competency Questions to their Corresponding Contextual Data												
CQ	A	B	C	D	E	F	G	H	I	J	K	L	M
PCE$_\Delta$T	1	2,3,4	5	6	7,,17	18	11,12	16	8,10	9	22	13,20	14,15,21

Table 5.3: Mapping situational information of Table 5.1 to PCE$_\Delta$T taxonomical element of Table 5.2

5.2.3 Summarising the Creation of PCE$_{\Delta}$T

As we have illustrated above, the creation of a PCE$_{\Delta}$T encounters the following

- $\mathtt{Ctg_i.Lev_j}$ and rlp$_r$ are already available in the taxonomical structure that FCM is initially consist of;
- the generic model has to be extended to cater for any situation-specific $\mathtt{Ctg_i.Lev_j}$ for the precision required;
- the generic model has to be extended to cater for any situation-specific $\mathtt{rlp_r}$ for the precision required.

In the following subsections we will go through the above for the running example.

5.2.3.1 Using Existing $\mathtt{Ctg_i.Lev_j}$ and $\mathtt{rlp_r}$

Apart from the root $\mathtt{Ctg_i.Lev_j}$ the only other $\mathtt{Ctg_i.Lev_j}$ available in the generic taxonomy are `Ojt-specific-Pfc` and `Health` as shown in Figure 5.6.

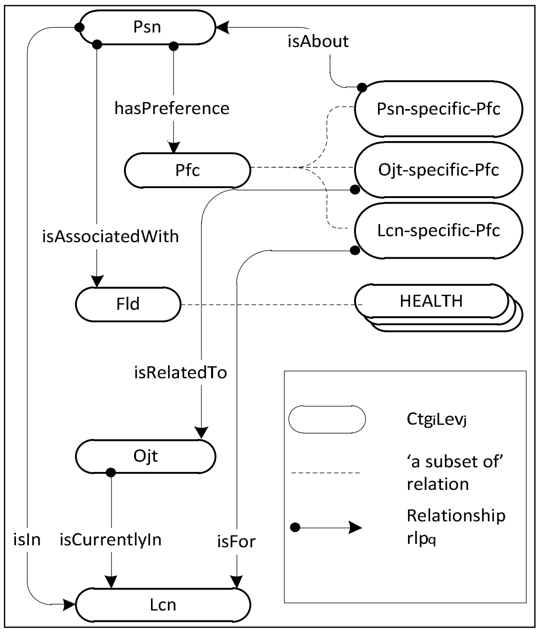


Figure 5.6: The generic PCE$_{\Delta}$T

These are based on Definition 11, 12, 13, 14, 15 and Axiom 6, 8, 10, 12, 14 and 20. Similarly, based on Axiom 16, 17, 18, 19 and 22, there are already five existing relationships rlp$_r$, which are `hasPreference, isAssociatedWith, isIn, isCurrentlyn` and `isRelatedTo`. The existance of these $\mathtt{Ctg_i.Lev_j}$ and $\mathtt{rlp_r}$ means that creation of situation-specific extensions to the PCE$_{\Delta}$T will not be delayed for the creation of the currently available elements of the PCE$_{\Delta}$T.

5.2.3.2 Extension of of PCE$_\Delta$T with Situation-specific $\texttt{Ctg}_\texttt{i}\texttt{.Lev}_\texttt{j}$ in SeCH

The generic taxonomy has to be extended to cater for any necessary $\texttt{Ctg}_\texttt{i}\texttt{.Lev}_\texttt{j}$. As shown earlier in Figure 5.5, there are quite a few $\texttt{Ctg}_\texttt{i}\texttt{.Lev}_\texttt{j}$ that will have to be added to the taxonomy to cater for all necessary precisions for $\texttt{ins}_\texttt{t}$ of the PCE$_\Delta$. Extensions of `Psn` to include `Resident` and of `Ojt` to include `Heater` are only required one level down. This means that the root category `Psn` will be $\texttt{Psn.Lev}_2$ and its extension `Resident` will be $\texttt{Psn.Lev}_1$. Likewise, the root category `Ojt` will be $\texttt{Ojt.Lev}_2$ and its extension `Heater` will be $\texttt{Ojt.Lev}_1$.

The extension of the `Lcn` root category happens twice, so structurally it is deeper than `Psn` and `Ojt` root category in this particular PCE$_\Delta$T. In this occasion `Lcn` will be $\texttt{Lcn.Lev}_3$ and its first extension `Physical Location` will be $\texttt{Lcn.Lev}_2$, and its second extension `Private` will be $\texttt{Lcn.Lev}_1$.

The `Fld` root category already has some enumerated $\texttt{Ctg}_\texttt{i}\texttt{.Lev}_\texttt{j}$ for variety of domains such as health, education, and manufacturing. In the running example, the domain is health therefore if `Fld` is $\texttt{Fld.Lev}_\texttt{m}$, `Health` would be $\texttt{Fld.Lev}_\texttt{m-1}$. Any extension in this part of PCE$_\Delta$T has to be an extension of `Health`. At the bottom of the extensions is `General Health` which will be $\texttt{Fld.Lev}_1$. The `Care Home` extension would be $\texttt{Fld.Lev}_2$ which is the immediate subset of `Health`. Therefore `Health` would be $\texttt{Fld.Lev}_3$ and the root would be $\texttt{Fld.Lev}_4$. So, structurally `Fld` is deeper than any other root category in this particular PCE$_\Delta$T. These extensions are depicted in Figure 5.7.

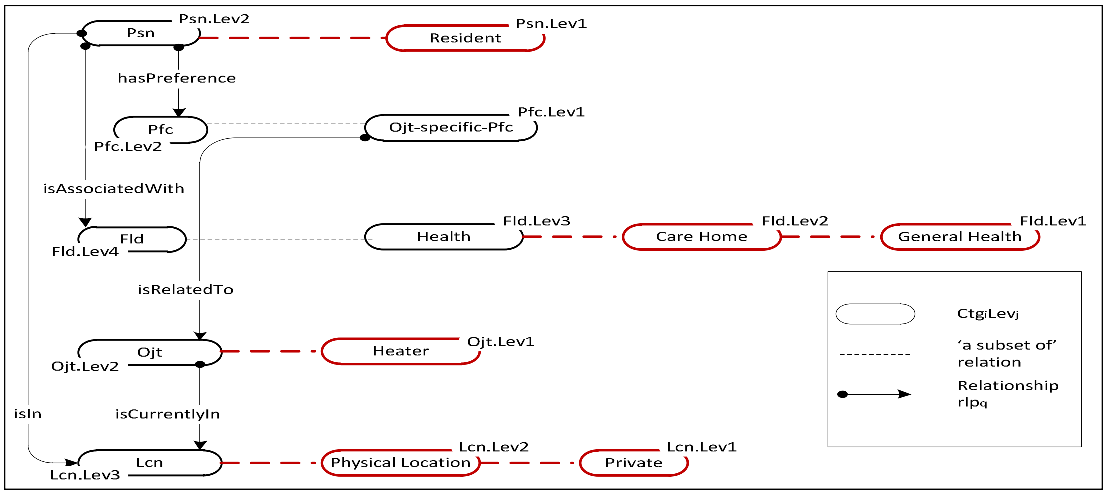


Figure 5.7 Extension of generic PCE$_\Delta$T to accommodate new $\texttt{Ctg}_\texttt{i}\texttt{.Lev}_\texttt{j}$

5.2.3.3 Extension of PCE$_\Delta$T with Situation-specific rlp$_r$ in SeCH

The only necessary relationship `rlp`$_r$ that is situation-specific and needs to be added to the taxonomy is `belongsTo(Mbr(heater152,Heater), Mbr(margaret,Resident))`. This is shown in Figure 5.8 in red.

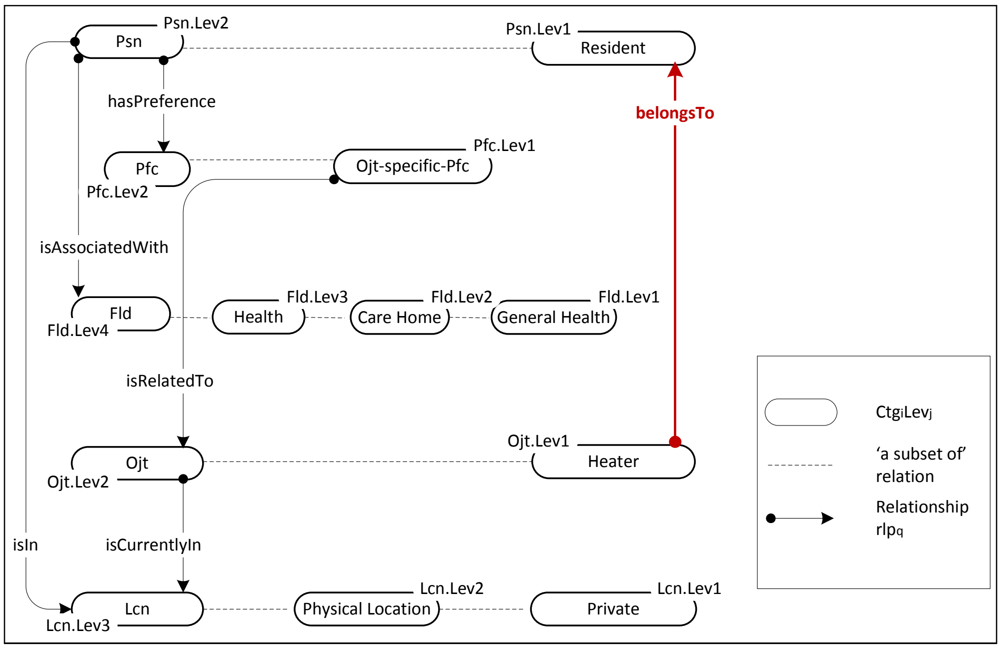


Figure 5.8 Extension of generic PCE$_\Delta$T to accommodate the new `rlp`$_r$

In the following section the transformation of the taxonomical notions to OWL ontological concepts are explained.

5.2.4 Illustration of Mapping PCE$_\Delta$T to OWL Ontological Concepts

As discussed in Chapter 2, OWL ontologies are modelled around four concepts. These concepts are individual, class, object property, and data type property. Variety of features that OWL supports for each one of these concepts is not the concern of this thesis. What is, however, important is to establish a mapping between each and every notion of the PCE$_\Delta$T and a sister OWL ontological model. Every piece of information represented in PCE$_\Delta$T must be reflected in the ontological model which in turn provides concepts necessary for delivering a service for a particular PCE$_\Delta$ (Definition 3).

The counterpart of `Ctg`$_i$`.Lev`$_j$ of PCE$_\Delta$T in ontology is "class". Although classes can have different relationships with each other in an OWL ontology, here we are only interested in the *is-a* (or subsumption) relationship, and when relationships is defined through object properties.

Every $\mathtt{ins}_t$ of PCE$_\Delta$T has some characteristics $\mathtt{chr}_q$. These are defined as data type properties. The domain of a data type property is the class which is the mapping of the $\mathtt{Ctg}_i.\mathtt{Lev}_j$ of the $\mathtt{Mbr}(\mathtt{ins}_t,\mathtt{Ctg}_i.\mathtt{Lev}_j)$. The range value of data type properties can be of different types, but we restrict to “string” type.

Instances $\mathtt{ins}_t$ in PCE$_\Delta$T becomes assertion of “individuals” in OWL ontology. When an individual is asserted, the class it is a member of has to be stated. Once an individual is asserted, its data type properties’ value can also be asserted.

Similar to relationship $\mathtt{rlp}_r$ in PCE$_\Delta$T that is a relationship between two $\mathtt{ins}_t$, object properties in Owl ontology are also relationships between individuals. The object property is defined by its domain and range, which by definition are classes. When individuals are asserted if there are any relationships between them, object properties will be asserted.

For ease of reference the mapping of key constructs of PCE$_\Delta$T and OWL ontology is given in Table 5.4.

	Key Constructs of PCE$_\Delta$T and OWL ontology			
PCE$_\Delta$T	$\mathrm{Ctg}_i.\mathrm{Lev}_j$	Relationship rlp_r	Characteristic chr_q	Real world instance ins_t
OWL ontology	Class	Object Property	Data Type Property	Individual

Table 5.4: Mapping of key elements of PCE$_\Delta$T and constructs OWL ontology

Following the above mapping guideline, the generic PCE$_\Delta$T shown in Figure 5.6 is represented as shown in Figure 5.9 for the generic OWL ontology. We name this generic ontology GOnto. The name of classes in GOnto, unlike in PCE$_\Delta$T, have been deliberately chosen from real terms in spoken English. The reason for this rational was to make these as close to terms used in a competency question as possible. Consequently, this will lend itself to easier writing, and interpretation of rules which are based on the ontological concepts and are governing the delivery of services to users of PCEs.

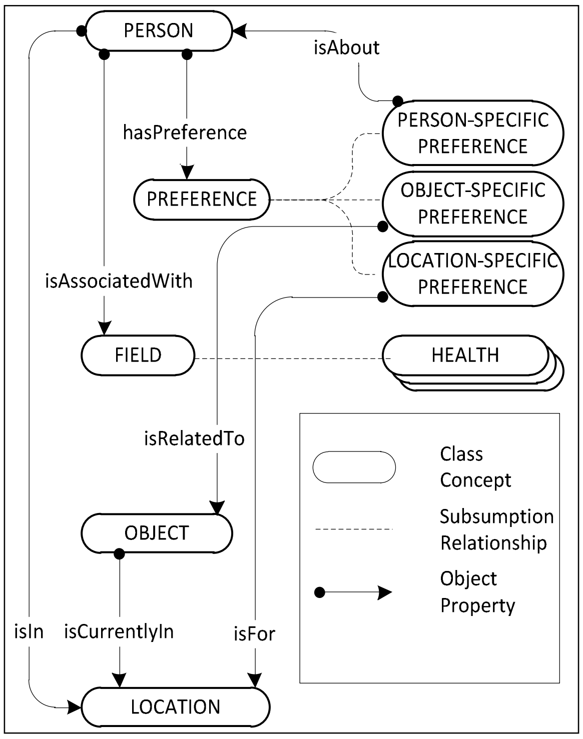


Figure 5.9: The generic GOnto OWL ontology

The object and data type properties name remain as in the PCE$_\Delta$T as they are already close to the speaking language.

In the following section the development of the extended ontology based on given GOnto, and the CQ reflected in the PCE$_\Delta$T is discussed.

The steps for the transformation of PCE$_\Delta$T to OWL ontological model is summarised in Table 5.5. These steps are based on definitions and axioms, and the processing procedure stated in Chapter 4. As a result of these steps an extended ontology based on GOnto is developed. The availability of concepts in GOnto does not mean that every one of them have to be used. The PCE$_\Delta$ determines which ones need to be used. The extended ontology, named SeCHOnto, will provide all the concepts necessary for the delivery of services. The SeCHOnto ontological model is given in Figure 5.10. In this figure, the extended concepts representing the particular situation PCE$_\Delta$ are depicted in red.

Contextual Data	Based on	Sub-procedure	Location in the Processing
Resident is Psn.Lev$_1$ Mbr(Margaret,Resident)	Axiom 1, 2, 5 Definition 7, 8, 11	Create subclass Resident of PERSON Assert margaret into Subclass Resident	Create subclass Ctg$_i$.Lev$_{J-1}$ of PERSON Assert Individual ins$_t$ into Subclass Ctg$_i$.Lev$_1$
(Mbr(margaret,Resident),name,“Margaret”)	Definition 16	Add (name, “Margaret”) to Margaret	If $\exists$(chr$_q$,Ctg$_i$.Lev$_1$)|chr$_q$ is characteristic of Ctg$_i$.Lev$_1$ Then Add(chr$_q$, vlu$_q$) to Individual ins$_t$
(Mbr(margaret,Resident),gender,“female”)		Add (gender, “female”) to Margaret	
(Mbr(margaret,Resident), assignedRoom, “Room101”)		Add Datatype Property assignedRoom into Subclass Resident Add (assignedRoom, “Room101”) to Margaret	Add Datatype Property chr$_q$ into Subclass Ctg$_i$.Lev$_1$ Add(chr$_q$, vlu$_q$) to Individual ins$_t$
Health is Fld.Lev$_3$ Mbr(UK-health, Health)	Axiom 1	Create subclass Health of FIELD	If Ctg$_i$.Lev$_{J-1}$ = Health Then Create subclass Ctg$_i$.Lev$_{j-2}$ of HEALTH
Care Home is Fld.Lev$_2$ Mbr(SeCH, Care Home)		Create subclass Care Home of FIELD	Create subclass Ctg$_i$Lev$_k$ of Field if $\nexists$
General Health is Fld.Lev$_1$ Mbr(feverish, General Health)	Axiom 1, 2, 9 Definition 7, 8, 13	Create subclass General Health of FIELD Assert feverish into Subclass Resident	
Heater is Ojt.Lev$_1$ Mbr(heater152, Heater)	Axiom 1, 2, 7 Definition 7, 8, 12	Create subclass Heater of OBJECT Assert heater152 into Subclass Heater	Create subclass Ctg$_i$.Lev$_{J-2}$of OBJECT if $\nexists$ Assert Individual ins$_t$ into
(Mbr(heater152,Heater),status,“off”)	Definition 16	Add (status, “off”) to heater152	If $\exists$(chr$_q$,Ctg$_i$.Lev$_1$)|chr$_q$ is characteristic of Ctg$_i$.Lev$_1$ Then Add(chr$_q$, vlu$_q$) to Individual ins$_t$
(Mbr(heater152,Heater),type,“heater”)		Add (type,“heater”) to heater152	
Physical Location is Lcn.Lev$_2$ Mbr(insideSeCH,Physical Location)	Axiom 1, 2, 13 Definition 7, 8, 15	Create subclass Physical Location of LOCATION	Create subclass Ctg$_i$.Lev$_{J-1}$ of LOCATION
Private is Lcn.Lev$_1$ Mbr(room101,Private)		Create subclass Private of LOCATION Assert room101 into Subclass Private	Create subclass Ctg$_i$.Lev$_{J-1}$ of LOCATION Assert Individual ins$_t$ into

Contextual Data	Based on		Sub-procedure	Location in the Processing
Ojt-specific-Pfc is Pfc.Lev$_1$ Mbr(heaterPreference,Ojt-specific-Pfc)	Axiom 1, 2, 11 Definition 7, 8, 15		Create subclass Object-specific-Preference of PREFERENCE Assert heaterPreference into Subclass Object-specific-Preference	Create subclass Ctg$_i$.Lev$_{J-1}$ of PREFERENCE Assert Individual ins$_t$ into Subclass Ctg$_i$.Lev$_1$
(Mbr(heaterPreference,Ojt-specific-Pfc), objectType,“heater”	Definition 16		Add Datatype Property objectType into Subclass Object-specific-Preference Add (objectType, “heater”) to	Add Datatype Property chr$_q$ into Subclass Ctg$_i$.Lev$_1$
(Mbr(heaterPreference,Ojt-specific-Pfc), objectStatus, “on”)			Add Datatype Property objectStatus into Subclass Object-specific-Preference Add (objectNewStatus, “on”) to	Add (chr$_q$, vlu$_q$) to Individual ins$_t$
(Mbr(room101, Private), roomTemperature, “cold”)			Add Datatype Property roomTemperature into Subclass Private Add (roomTemperature, “cold”) to	
isAssociatedWith(Mbr(margaret,Resident), Mbr(feverish, General Health))	Definition 17	Axiom 16	Add Object Property isAssociatedWith between margaret and feverish	Add Object Property isAssociatedWith (ins$_t$, ins$_u$)
isIn(Mbr(margaret,Resident), Mbr(room101,Private))		Axiom 17	Add Object Property isIn between margaret and room101	Add Object Property isIn (ins$_t$, ins$_u$)
isCurrentlyIn(Mbr(heater152, Heater), Mbr(room101,Private))		Axiom 18	Add Object Property isCurrentlyIn between heater152 and room101	Add Object Property isCurrentlyIn (ins$_t$, ins$_u$)
hasPreference(Mbr(margaret,Resident), Mbr(heaterPreference,Ojt-specific-Pfc))		Axiom 19, 20	Add Object Property hasPreference between margaret and heaterPreference	Add Object Property hasPreference (ins$_t$, ins$_u$)
isRelatedTo(Mbr(heaterPreference,Ojt-specific-Pfc), Mbr(heater152, Heater))		Axiom 22	Add Object Property isRelatedTo between heaterPreference and heater152	Add Object Property isRelatedTo (ins$_t$, ins$_u$)
belongsTo(Mbr(heater152,Heater), Mbr(margaret,Resident))		Extension	create Object Property belongsTo between heater152 and margaret	Create Object Property belongsTo (Ctg$_i$.Lev$_1$, Ctg$_x$Lev$_1$) Add Object Property belongsTo

Table 5.5 Transformation of detected contextual data to ontological concepts

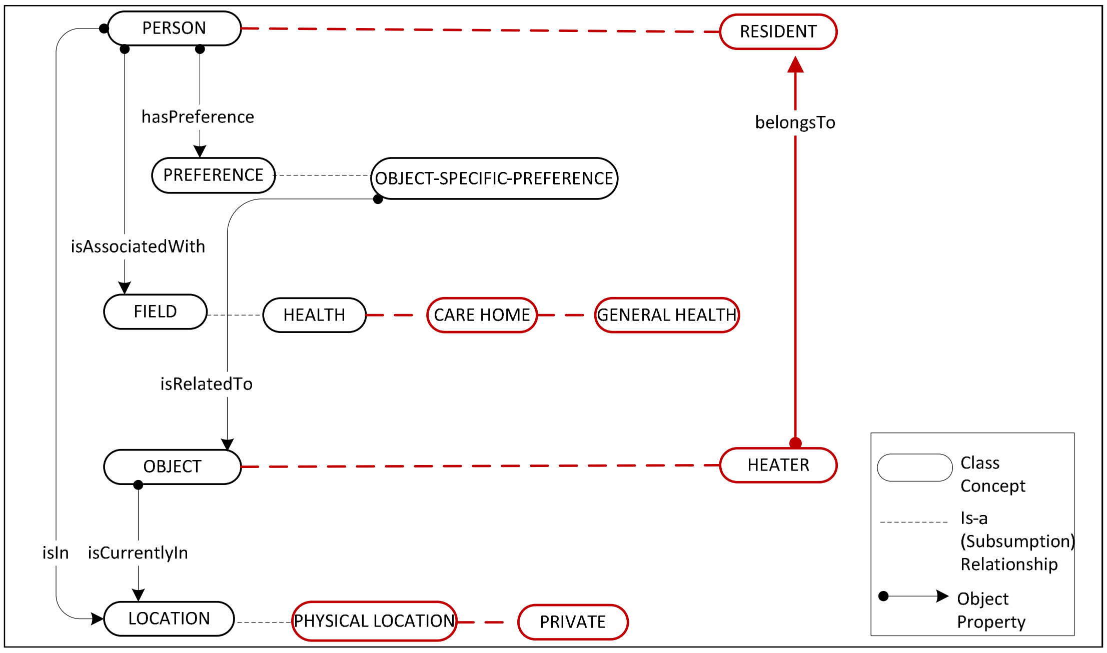


Figure 5.10: The SeCHOnto OWL ontology

5.2.5 Illustration of Delivering a Service in a PCE$_\Delta$

If we were to prepare a SWRL rule, which could support the CQ stated in section 5.5, then Table 5.6 illustrates the mapping between the text in the CQ and the atoms of that SWRL rule. In other words, the second column in Table 5.6 gives us directly the syntax and semantics of the SWRL rule ensuring the delivery of the service that *the heater in Margaret's room should be turned on*; `ToBeTurnedOnObject(?h)`.

Preparation of Table 5.5 from Table 5.1 goes through a filtering process. Segments of the CQ which are not situation-specific but rather structural statements shall not have any place in the SWRL rule. After all, SWRL rule is about a specific situation Δ. This is why statements C, D and G which are structural statements without any corresponding SWRL rule atoms are faded out in Table 5.6.

The SWRL rule corresponding to the CQ is:

```
General_Health(?gh),Heater(?h),LOCATION(?l),OBJECT-
SPECIFIC-PREFERENCE(?osp),Resident(?r),belongsTo(?h,?r),
hasPreference(?r,?osp),isAssociatedWith(?r,?gh),
isCurrentlyIn(?h,?l),isIn(?r,?l),isRelatedTo(?osp,?h),
assignedRoom(?r,?ln),bodyTemperature(?gh,"feverish"),
locationName(?l,?ln),objectNewStatus(?osp,"on"),
roomTemperature(?l,"cold"),status(?h,"off")->
ToBeTurnedOnObject(?h)
```

The SWRL rule has in its body a set of OWL restrictions which we explain in their order of appearance in Table 5.6.

The CQ begins with the statement *"Margaret is a resident in SeCH"*. This justifies the presence of a `Resident` class inside the body (antecedent) of the rule as a unary predicate, `Resident(?r)`. As the rule is general and should be applied to any individual of the Resident class, the argument of the atom is not an actual value such as "`margaret`" and therefore a variable `?r` is used instead.

	Competency Question Statement	SWRL Rule Atoms
A	*Margaret is a resident in SeCH.*	`Resident(?r)`
B	*Margaret's name, gender and her assigned room are examples of available information from SeCH.*	`assignedRoom(?r,?ln)`
C	*SeCH is in the UK and therefore it follows the UK health policy and procedures.*	
D	*SeCH is a care home.*	
E	*Margaret is feverish.*	`General_Health(?gh)` `bodyTemperature(?gh, "feverish")` `isAssociatedWith(?r,?gh)`
F	*Her current location is "Room 101"*	`LOCATION(?l)` `isIn(?r,?l)`
G	*which is a private physical location (bedroom) inside SeCH.*	
H	*This location (Room101) is "cold" (considering Margaret's body temperature).*	`roomTemperature(?l,"cold")`
I	*A heater "heater152" is present in Room101.*	`Heater(?h)`
J	*The status of this heater is "off".*	`status(?h,"off")`
K	*This heater belongs to Margaret.*	`belongsTo(?h,?r)`
L	*Margaret has indicated when she was admitted to SeCH some object preferences.*	`OBJECT-SPECIFIC-PREFERENCE(?osp)` `hasPreference(?r,?osp)`
M	*One of these preferences is the heater in her room to be turned on, if it is off, when she is feverish.*	`isRelatedTo(?osp,?h)` `isCurrentlyIn(?h,?l)` `objectNewStatus(?osp,"on")`

Table 5.6: Mapping the CQ semantics with the atoms of SWRL rule

There are varieties of information available about `?r` as soon a PCE$_{\Delta}$ is created. To name a few, the individual's `name`, `gender`, `d.o.b.` are some examples. However, despite their availability some of them are not semantically useful information as far as the CQ is concerned. In other words, whether the person is male or female, young or old, the SWRL rule holds. There is, never the less, one piece of information about a resident ?r which is important as far as the CQ is concerned. This is the resident's assigned room. When residents are registered in SeCH their bedroom which is their assigned room is known. This, therefore, justifies the presence of the binary predicate assignedRoom(?r,?ln)in the body of the rule. This predicate is a data type property which has Resident as its defined domain. Its range value is of type string, which means the variable ?ln can only take string values.

The CQ states that the device activation takes place when there is a change in SeCH; i.e. Margaret is feverish. We have seen before the justification for the FIELD hierarchy. General Health is the bottom level class of FIELD. When the Context Management detects the change, it produces the information about the feverishness of the person. Consequently an individual "feverish" of class General Health is created, as shown in Figure 5.5. This therefore verifies the presence of General_Health(?gh). Instead of "feverish" a general variable "gh" is used for this unary predicate. The General_Health class has a bodyTemperature data type property which has string range value. In this particular case the range is "feverish", that is bodyTemperature(?gh, "feverish"). An individual of FIELD must always be associated with a PERSON, as discussed before. In OWL ontology, it is always an object property that links two individuals together. In this case the object property isAssociatedWith links an individual of Resident (as the domain) to an individual of General Health (as the range). Object properties are always shown in binary predicate, isAssociatedWith(?r, ?gh).

Next statement in the CQ is "*Her current location is "Room 101".* The inevitability of having a unary predicate, similar to Resident(?r) for LOCATION is evident, hence the atom LOCATION(?l). It is important to note that ?l is an individual of the class LOCATION unlike the variable ?ln in assignedRoom(?r,?ln)that is a string type for the name of a location. Similar to object property

isAssociatedWith, there is a need to link the LOCATION and Resident individuals. This is the reason for the inclusion of isIn(?r,?l) in the SWRL rule.
The statement *"This location is 'cold'"* bring about the atom roomTemperature(?l,"cold") in which roomTemperature is a data type property of LOCATION class which has string range value.

The next statement is *"A heater "heater152" is present in Room101".* This gives enough ground to have Heater as a class predicate inside the body of the rule. The following statement, *"The status of this heater is 'off'."* requires the predicate status(?h,"off"). Considering that OBJECT class has status data type property and that Heater is a subclass of OBJECT, it is fine to have ?h individual which is of type Heater as the domain argument.

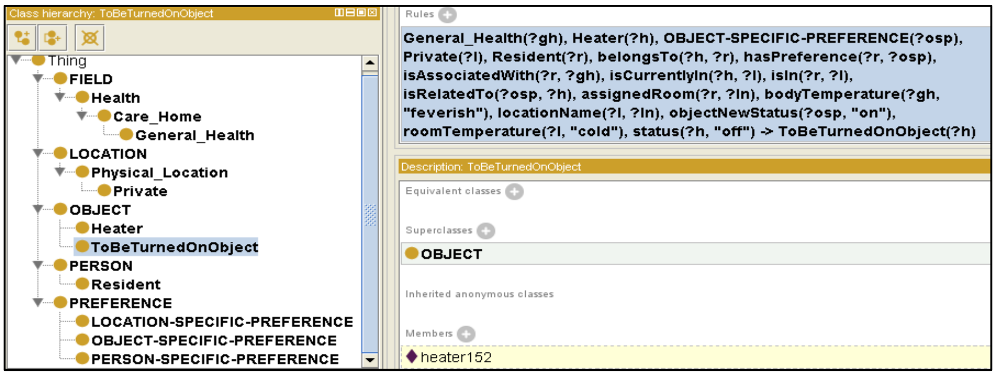
The CQ imposes that the heater has to belong to Margaret to be activated, where it says *"This heater belongs to Margaret."*. This requires an object property with Heater as its domain and Resident as its range. This is precisely why we have belongsTo (?h,?r) as an atom in the SWRL rule.
"Margaret has indicated … some object preferences." clearly justifies OBJECT-SPECIFIC-PREFERENCE (?osp). As before, to relate the individual ?osp with individual ?r a binary object property predicate is used, hasPreference(?r, ?osp). The last statement specifies what type of object this preference is referring to where it says *"One of these preferences is the heater…"*. The object property isRelatedTo which has OBJECT-SPECIFIC-PREFERENCE as its domain and OBJECT as its range is used, hence isRelatedTo(?osp,?h).
Further down the CQ adds that *"…the heater in her room to be turned on…".* Considering that earlier we have used the variable ?l for the location of Margaret, now the same variable has to be used in the rule for the location of the heater object. As before, for this purpos isCurrentlyIn which is another object property, is used. That is why isCurrentlyIn(?h,?l)is part of the rule.
The preference is to turn the heater "on". So this information is available as a string value for a data type property of ?osp individual of OBJECT-SPECIFIC-

PREFERENCE class. This is shown as `objectNewStatus(?osp,"on")` which is a binary predicate among the premises of the SWRL rule.

So far we have covered only the atoms of the body of the rule. All the atoms of the body must be true for the rule to trigger the action specified in the head (consequent) of the rule. When all the atoms of the body of the rule corresponding to the CQ are true, then the specific heater in Margaret's room that is currently off has to be turned on (item iii in section 5.3). The physical process of turning the heater on is not the responsibility of ASeCS. What is its responsibility is to deliver the service that triggers an actuator (item 3 in section 5.3). For this purpose, we have extended the `OBJECT` class horizontally to have a subclass of `OBJECT` called `ToBeTurnedOnObject`. This class will accommodate all individual objects which have to be turned on. In this particular CQ when all the conditions are met, the heater `?h` which is a member of the class `Heater` will be also a member of the class `ToBeTurnedOnObject`. This is why `ToBeTurnedOnObject(?h)` by itself defines the head of the SWRL rule.

5.2.6 Running the Rule

Once all the necessary GOnto concepts (including classes, object property, data type property, and individuals) have been identified, and where necessary the GOnto is extended to have all necessary concepts required for the SWRL rule, a reasoner engine can run the rule to reason upon the assertions to infer new knowledge about already existing individuals. We used the built-in Pellet reasoner (Pellet, 2004) in Protégé 4.0 (Protégé, 2009) ontology editor to run the SWRL rule. We show the final result of running the rule in Figure 5.11.



```
General_Health(?gh), Heater(?h), OBJECT-SPECIFIC-PREFERENCE(?osp),
Private(?l), Resident(?r), belongsTo(?h, ?r), hasPreference(?r, ?osp),
isAssociatedWith(?r, ?gh), isCurrentlyIn(?h, ?l), isIn(?r, ?l),
isRelatedTo(?osp, ?h), assignedRoom(?r, ?ln), bodyTemperature(?gh,
"feverish"), locationName(?l, ?ln), objectNewStatus(?osp, "on"),
roomTemperature(?l, "cold"), status(?h, "off") -> ToBeTurnedOnObject(?h)
```

Figure 5.11: The result of running the SWRL rule for the running example CQ

The screen shots of steps that the FCM is being applied for the running example through a Java application is shown in Appendix A, and the software application with GOnto is supplied in Appendix B.

5.3 Summary

In this chapter, we have set the scene for a successful implementation of the FCM. As PCE systems are domain-specific, the evaluation had to be done within a specific domain. Considering the increasing demand for PCE systems in healthcare domain we have chosen a care home environment and stated an example scenario. The scenario is simple but comprehensive to address all elements of $PCE_{\Delta}T$.

An example of a software application of ASeCS architecture was developed using Java technologies. We have shown through a CQ the situational information of the example scenario that have to be detected for the computational model to be able to reason about the situation. This is illustrated through several interactions using SWRL enabled OWL ontology.

Chapter 6
Evaluation and Reflection

In this chapter, we evaluate and reflect on achievements against objectives set in Chapter 1; explaining how the FCM addresses concerns of Chapter 1 and views in Chapter 3. We will discuss design decisions in modelling the Generic $PCE_{\Delta}T$. Problems encountered during implementation are also pointed out in the chapter.

The aim of this research was to specify a formal computational model that can represent an abstraction of computationally significant semantics of situations in domain-specific PCEs. In this chapter therefore, we also elaborate on other achievments in our journey towards the creation of the FCM.

6.1 EVALUATION

6.1.1 Meeting Objectives

In Chapter 2 we have reviewed some examples of solutions in creating intelligent applications in PCEs, with the intention that they could help us with our vision of PCEs outlined in Chapter 1. In section 3.2 and 3.3 we have analysed particularly shortcomings of the current software technologies in PCEs, and the limitations of context-aware applications respectively. The review of related work and our earlier experiences (Shojanoori, et al. 2008, 2009) lend themselves to the conclusion that we could not use existing software solutions to address problems and shortcomings in pervasive computing. In the following subsections the four objectives of the thesis are evaluated.

6.1.1.1. Outlining Problems in PCE Research and their Shortcomings

The first objective of the research was to analyse and summarise the problems in and shortcomings of pervasive computing, and assess the way they have been addressed in SE in the last decade. We started our background reading from the time when the vision of ubiquity of computing emerged (section 2.1.1), which was in the mid ‘70s. The vision challenged the HCI and AI communities by asking them to focus on users (section 2.1.2) and make computing devices, which have become mobile and wireless, more aware of their environments (section 2.1.3). The gradual recognition of ubiquitous computing (section 2.1.4) which led to the emergence of pervasive computing in 2001 (section 2.2.1) created numerous and different perceptions of PCEs. The examples are: users should be in control of non-intrusive PCEs; PCEs should provide services to users anywhere and at anytime; preferences of users should be observed in PCEs; PCEs should be context-aware; PCE devices should be able to share knowledge and to reason about the environment; PCE applications should operate in highly dynamic environments and many more (section 2.2.2). Despite various perceptions of PCEs, context-awareness has been a defacto necessity of such environments. We anticipated, therefore, that without correct contextual information we could not secure the delivery of the service to the users in PCEs, thus a thorough analysis of context-aware applications was necessary in order to understand their extent of usefulness for PCEs (section 2.3). However, the available research on context awareness was rather disappointing. We have highlighted the lack of common consensus on what exactly context may mean and have become aware that context-aware software applications and context modelling in such applications did not guarantee the delivery of the expected outcome in PCEs. Imperfection of context information, inflexibility of context models, domination of localisation-aware systems, the lack of high-level abstraction and consequently the lack of context reasoning, were identified as stumbling issues in the realisation of PCEs, if the traditional context awareness is being used.

6.1.1.2. Agreeing on Common Characteristics of and Situations in PCEs

The second objective of the research was to create a list of common characteristics of PCEs which may systemise our perception on what PCEs are and what we expect from them. These common characteristics should also be useful in the FCM, which in turn will help us to define and create a situation in a PCE which will deliver an expected service. The common characteristics of PCEs are derived in Chapter 3 from both expectations we may have from PCEs and limitations of current context aware software solutions in pervasive computing. Therefore in Chapter 3, we focus on expectations from pervasive computing and focused on issues such as: a very short time span of information, which describes a situation in PCEs, problems with 'location-aware only' mobile applications that are perceived as context-aware applications, context-aware applications with fixed hardware infrastructure with no regard to the abstraction of situational knowledge in PCEs and consequently are incapable of supporting scalability which is a key issue in PCEs. We have also shown the lack of support for inference and reasoning in context aware applications; a necessity in PCEs to infer new knowledge and reason about situations to deliver services expected by their users.

We advocate that PCEs are user-centric, have dynamically defined inputs, require flexible interfaces, implicit interaction between users and pervasive computational devices, and might not depend on historical information accumulated in PCEs at all because they focus on a particular situation in PCE. Consequently, traditional contexts are not sufficient to define and manipulate PCE and therefore we should augment or replace it with situations in order to deliver expected services to the users of PCEs. The purpose of PCEs has become more clear: this is a non-autonomous environment, which empowers its users, without overloading them with information, behaves as a proactive environment and, at the same time, unobtrusive with minimal distraction to their users. The common characteristics of PCEs might be sufficient to manage the description of semantics of situations we may encounter in PCEs. However, a formal computational model is needed in order to define which of its elements we must have, and which computational steps we must perform in order to secure the existence of computations, that deliver a service in PCEs.

6.1.1.3. Defining the FCM

The third objective of the research is to define an FCM which will allow representation of computationally significant semantics of any situation in PCEs for the purpose of delivering situation-specific services. In Chapter 4 we have provided 17 definitions and 24 axioms about notions and contributing elements of the formal computational model for defining situations in PCEs, and reason about them to deliver services to their users.

We started our journey towards the creation of formal computational model by defining what PCEs, situation encountered in them and services, which are expected to be delivered to PCE users, are. Consequently, we require that each situation PCE_{Δ} comprises a finite number of **real world instances**, and for all $\mathtt{ins}_{\mathtt{t}}$ that share the same features, we have defined **category** $\mathtt{Ctg}_{\mathtt{i}}$ as an important abstraction in the PCE. When more precision is required for representation of the semantics of an $\mathtt{ins}_{\mathtt{t}}$, lower levels of abstractions are needed. Category $\mathtt{Ctg}_{\mathtt{i}}$ should therefore be seen as having subsets which allow for various levels of abstraction. We must know exactly which subset of category $\mathtt{Ctgi}$ the $\mathtt{ins}_{\mathtt{t}}$ belongs to. To differentiate between different subsets of a $\mathtt{Ctg}_{\mathtt{i}}$, each of them is qualified with a **level** $\mathtt{Lev}_{\mathtt{j}}$ (Definition 6). When an $\mathtt{ins}_{\mathtt{t}}$ is detected, it is always at the **"leaf"** level $\mathtt{LCtg}_{\mathtt{i}}\mathtt{.Lev}_{\mathtt{j}}$ (Definition 9) of its **"root"** category $\mathtt{RCtg}_{\mathtt{i}}\mathtt{.Lev}_{\mathtt{j}}$ (Definition 10).

Creating a situation-specific taxonomical structure $PCE_{\Delta}T$ for a PCE_{Δ} is an important step towards a FCM. $PCE_{\Delta}T$, as the **taxonomical structure** of the real world participants in PCE_{Δ} is described through memberships of instances within categories $\mathtt{Mbr(ins_t,Ctg_i.Lev_j)}$ and we have defined (Definition 11-15) five **$\mathtt{RCtg_i.Lev_j}$ occurrences** in $PCE_{\Delta}T$, namely **$\mathtt{Psn}$** (for person as users have a central role in PCEs and existence of one without user is not possible, P8 and P11 in Table 3.3), **$\mathtt{Fld}$** (for field, encompassing all possible abstractions of $\mathtt{ins}_{\mathtt{t}}$s, of domain-specific information in any PCE_{Δ}, P18 in Table 3.3), **$\mathtt{Ojt}$**(for object, encompassing all possible abstractions of $\mathtt{ins}_{\mathtt{t}}$s of cyber and physical objects in any PCE_{Δ}, P1, P2 and P3 in Table 3.3), **Pfc** (for preferences, encompassing all possible abstractions of ins_ts of preferences of users in any PCE_{Δ}, P12 in Table 3.3), and **Lcn**(for location, encompassing all possible abstractions of ins_ts of physical or cyber locations in any PCE_{Δ} , P1, P9 and P19).

If detected information in a particular PCE_{Δ} cannot be abstracted into any of the occurrences $RCtg_i.Lev_j$ and their subsets, then we should be able to find an element within the $PCE_{\Delta}T$ which may accommodate such semantics. An instance **characteristic** chr_q which is a description of a $Mbr(ins_t,Ctg_i.Lev_j)$ with value vlu_q and represented in a triplet does exactly that (Definition 16).
We have also defined binary **relationship** rlp_r between $PCE_{\Delta}T$ elements (Definition 17) to allow relationships within the $PCE_{\Delta}T$ between $Mbr(ins_t,Ctg_i.Lev_j)$. We have summarised these definitions and axioms of section 4.1 in Figure 4.9.

The formal computational model (FCM) is presented in section 4.2. Loops and steps towards the creation of $PCE_{\Delta}T$ to deliver a domain and situation-specific service in a PCE_{Δ} are divided into three parts and depicted in Figure 4.10. The first part addresses the extension of $Ctg_i.Lev_j$, insertion of ins_t, adding chr_q to $Ctg_i.Lev_j$ and finally assigning vlu_q to chr_q for each ins_t. The second part addresses the creation of generic relationships, $rlp_r(Mbr(ins_t, Ctg_i.Lev_j), Mbr(ins_u, Ctg_xLev_y))$. The third part addresses the creation of situation-specific extended relationships. The FCM shown in Figure 4.10 is a programming language-independent model which has the flexibility of accommodating any changes to the generic $PCE_{\Delta}T$ in terms of adding new occurrences $RCtg_i.Lev_j$, new rlp_r or chr_q.

Knowing that we will use the SW technology stack as the result of the background research (section 2.4), the FCM should also be expressed in vocabulary and following the terms of the suitable technology to secure the implementations of formalised computations. Both vocabulary and terms of OWL ontology language have influenced the format and the content of the FCM shown in pseudo code in section 4.2.2.2. Unlike boxes of Figure 4.10, which represent a generic formal computational model, the FCM pseudo code is tailored to the proposed model prescribed in section 4.1.

To complete the journey towards creation of the FCM, here we would like to reiterate that the power of the FCM is creating a semantically rich $PCE_{\Delta}T$, without which we cannot secure the delivery of a situation-specific service in PCE. However, the FCM cannot fully specify the exact computation of services to be delivered, because services are domain and situation-specific. $PCE_{\Delta}T$, ensured by the FCM, is semantically rich that may trigger automatically reasoning for the delivered service. Therefore, $PCE_{\Delta}T$ can have additional rules to trigger the situation-specific services.

6.1.1.4. Illustrating the FCM

The fourth objective of the research was to illustrate and implement the proposed FCM in a domain of interest, using SWRL enabled OWL ontology. We have set the scene for the application of the FCM in section 5.1. by choosing remote patient monitoring from the **healthcare domain** (section 5.1.1). We introduce **healthcare environment of Self-care homes (SeCH)**, with residents who require constant or occasional support. SeCH is equipped with sensors, which detect the whereabouts of its residents and monitor their activities and physiological functions (Figure 5.1). Issuing a health related recommendation to residents (users), informing them of any changes that concern them, activating devices around them automatically, raising an alarm for the medical staffs on duty, are some examples of the services delivered in SeCH.

The **example scenario** that is relatively humble is about Margaret who is a resident in SeCH and like all other residents is being monitored so as to be attended to whenever there is a change in her health situation.

Illustration of $PCE_{\Delta}T$ creation according to the scenario is given in section 5.2, in which we have explained how the situational information for the creation of $PCE_{\Delta}T$ is established through the use of CQ. We have formulated the CQ and segmented it in Table 5.1 and showen how it is used to detect $\mathtt{ins}_{t}$s and subsequently to create $PCE_{\Delta}T$. The breakdown of the CQ in Table 5.1 demonstrates how each piece of information, a word, a phrase or a complete sentence in the CQ can be mapped to a formal element of PCE$_\Delta$T as prescribed in three parts of Figure 4.10.

Part 1 of Figure 4.10: In section 5.2.2.1, we went through all the segments of the CQ and subsequently created new $\mathtt{ctg_i.Lev_j}$, $\mathtt{Mbr(ins_t,ctg_i.Lev_j)}$, and $\mathtt{(Mbr(ins_t,Ctg_i.Lev_j),chr_q,vlu_q)}$ and represented them in formal specification in terms of a formal element of PCE$_\Delta$T.

Part 2 of Figure 4.10: Then in section 5.2.2.2, we went through all the segments of the CQ and created generic relationships $\mathtt{rlp_r(Mbr(ins_t,Ctg_i.Lev_j),Mbr(ins_u,Ctg_xLev_y))}$.

Part 3 of Figure 4.10: In section 5.2.2.3 we finally went through all the segments of the CQ and created domain and situation-speciic relationships of the PCE$_\Delta$T.

The formal content of PCE$_\Delta$T is summarised in Table 5.2, and a mapping between them and the situational information of Table 5.1 is presented in Table 5.3.

The use of existing generic $\mathtt{Ctg_i.Lev_j}$, and generic $\mathtt{rlp_r}$ for the example scenario is depicted in Figure 5.6, extension of PCE$_\Delta$T with situation-specific $\mathtt{Ctg_i.Lev_j}$ is shown in Figure 5.7, and extension of PCE$_\Delta$T with situation-specific rlp$_r$ in illustrated in Figure 5.8. The generic GOnto ontology and the extended situation-specific SeCHOnto ontology mapping the PCE$_\Delta$T taxonomical elements for the example scenario situation are summarised in Figure 5.9 and 5.10 respectively.

Transformation of PCE$_\Delta$T taxonomical elements to OWL ontology entities (or concepts) had to be done to perform the implementation of Margaaret's situation outlined in the CQ. OWL ontologies are modelled around four concepts: individual, class, object property, and data type property (section 2.4.1). Every piece of information represented in PCE$_\Delta$T is mapped and represented in SeCHOnto, which in turn provides semantics necessary for delivering the service Margaret expects for her situation. For example, the counterpart of $\mathtt{Ctg_i.Lev_j}$ of PCE$_\Delta$T in ontology is "class". Although classes can have different relationships with each other in an OWL ontology, we are only interested in the *is-a* (or subsumption) relationship. Transformation of situational information to OWL ontological concepts is provided in Table 5.5. Every $\mathtt{ins}_t$ of $PCE_{\Delta}T$ has some characteristics $\mathtt{chr}_q$. These are defined as data type properties in OWL. The domain of a data type property is the class which is the mapping of the $\mathtt{Ctg}_i.\mathtt{Lev}_j$ of the $\mathtt{Mbr}(\mathtt{ins}_t,\mathtt{Ctg}_i.\mathtt{Lev}_j)$. The range value of data type properties can be of different types, but we restrict to “string” type. Instances $\mathtt{ins}_t$ in $PCE_{\Delta}T$ have become assertion of “individuals” in OWL ontology. When an individual is asserted, the class it is a member of has to be stated. Once an individual is asserted, its data type properties’ value can also be asserted. Similar to $\mathtt{rlp}_r$ relationship between two $\mathtt{ins}_t$ in $PCE_{\Delta}T$, object properties in Owl ontology are also relationships between individuals. The object property is defined by its domain and range, which by definition are classes. Their counterparts in $PCE_{\Delta}T$ are the two $\mathtt{Ctg}_i.\mathtt{Lev}_j$ that the $\mathtt{ins}_t$ at both sides of the corresponding $\mathtt{rlp}_r$ are member of.

Illustration of the reasoning about the situation PCE_{Δ} of the example scenario to deliver a service to the user of the PCE in that situation is explained in section 5.2.5, in which the CQ semantics of Table 5.1 is mapped to the premises of a SWRL rule in Table 5.6. Appendix A contains the screen shots of the implementation of the example scenario through a Java application, that communicates with the ontology and inference and reasoning layers of the architecture through OWL API communication channel.

6.1.2 Impact of Semantic Web Technologies

6.1.2.1 The Impact of SWT on the FCM

In Chapter 3, we have clearly stated that we will use the SWT stack for creating computations according to the FCM. This demands the use of technology dependent vocabulary and terms within the FCM. Therefore, SWT must have influenced the format and the content of the FCM and we have known that from the start of this research. However, this is not a major issue because the power of the proposed FCM is in its ability to grasp the semantics of PCEs, specify its conceptualisation, and perform reasoning upon it in order to deliver services. Furthermore, we need a technology which will enable us to deploy the computations from the FCM. Therefore, however influential SWT is in the delivery of the FCM is, SWT is a technology of choice, because it happens to respond fully to our needs.

The other impact of SWT on the FCM was attributed to the OWL ontology language. The syntax and semantics of the language did not have any influence on our research until section 4.2 where the FCM is formally specified using the "if-then-else" pseudo code. In other words, the proposed FCM described through a set of loops for creating categories and asserting instances in Figure 4.10 is not OWL-dependent. They guide any PCE designer to implement the FCM using any other languages and technologies rather than the recommended W3C technologies. However, the FCM pseudo code, which is nothing more than the translation of the semantics from the set of lopps of Figure 4.10 into OWL terms is OWL specific. Some readers may argue that the FCM in Figure 4.10 is completely technology specific, but it will be very difficult today to work differently. It is impossible to find any SE solution today which is NOT dependent on technology and which is so "generic", that actually any available software technology can be used to implement it.

Finally, the selection of situational information for the creation of $PCE_{\Delta}T$ is established through the use of CQs which is, strictly speaking, an OWL term. We have shown how dividing CQ into segments will eventually form the building blocks of $PCE_{\Delta}T$ systemise receiving and formalising situational information by PCEs. This dependency on OWL when mirroring the semantics of QC in $PCE_{\Delta}T$ is extremely important, because it can pave the way of high level of automation based on exact user preferences, which is very important in PCEs. If OWL and its terminology have secured such an outcome, then being OWL dependent when defining the FCM is not unreasonable. In contrary, the power and expressivity of OWL have exceeded our expectations and however difficult it was to find exactly what we should have in the FCM, OWL has helped us to successfully structure and evaluate the FCM.

6.1.2.2 The Impact of SWT Stack on the FCM

The SWT stack is extremely rich and offers choices of technology components, which can be used in understanding and interpreting the Web. Our choice of using SWRL enabled OWL ontologies was based on both

- the previous experiences of using them as a SE solution across various problem domains, and
- the relative maturity of OWL and SWRL.

Standardisation in software in general takes very long and we have experienced considerable discrepancies between different implementations because of the complexity of software standard. One of the best examples is a painful standardisation of SQL and numerous SQL dialects which still exist in applications using SQL. OWL, however, emerged in 2004 as the W3C recommended SW language (W3C, 2004b) rather quickly. The language is constantly being improved, but even the initial version was stable and convincing enough to be chosen for the deployment of the FCM.

Where is OWL in the FCM?

The loops of the FCM (Figure 4.10) are almost OWL and SWRL free because we did not want that the FCM to be programming language specific. However, the FCM was developed with the use of SWT in mind. The diagram and description of section 4.2.1 could be easily adapted to any new language should W3C decided to introduce and replace OWL or SWRL in the SWT stack. However, considering the existing SWT stack, and knowing that we wanted to implement the FCM using SWT, we had no choice but to use syntax and semantics of OWL in pseudo code in section 4.2.2.

6.1.2.3 The applicability of the FCM across problem domains

The FCM is applicable across domains because of several factors. Firstly, concepts which model problem specific semantics are not instantiated in the FCM. The occurences of $\texttt{RCtg}_{\texttt{i}}\texttt{.Lev}_{\texttt{j}}$ in the Generic $\text{PCE}_{\Delta}\text{T}$ have been carefully chosen that they are applicable to all PCEs and are reflecting common characteristics of PCEs

(Table 3.3). This commonality of $\mathtt{RCtg_i.Lev_j}$ ensures that the reusability of the formal model in different environments across domains.

Secondly, SE solutions use conceptualised knowledge and only through implementation and running of application programmes generated from the model, they become problem specific. This SE principle is observed in the FCM because the conceptualisation of domain and situation-specific PCE_Δ follows the steps shown in Figure 4.10, which has no specificity about any problem domain.

Thirdly, PCEs we advocate are always associated with a conceptualisation of a situation PCE_Δ within them. It is always the PCE_Δ, part of which is the user of the PCE, that manages the behaviour of the PCE. Therefore, if we have a sufficient mechanism in the FCM to handle any situation in PCE, then the FCM can be used across domains. Considering the generic $PCE_\Delta T$ (paragraph 4.1.10) that is common among all PCEs, and the Figure 4.10 steps to extend it with situation-specific $\mathtt{Ctg_i.Lev_j}$ to create extended $PCE_\Delta T$, the FCM is applicable across problem domains.

6.2 Reflection

6.2.1 The FCM and the Generic Taxonomical Structure

6.2.1.1 What Influenced the Generic Taxonomical Structure

The constraints defined in the FCM are summarised in Figures 4.9 and 4.10. We would like to note that we had to strike a balance between the re-usability of the FCM and its semantic expression. Too many constraints within the generic taxonomical structure $PCE_\Delta T$ would minimise the reusability of the FCM across domains and would not be used across various situations in PCEs. It is true that a model without constraints is too general to be “decidable”, but too many constraints might also weaken inference mechanisms or even prevent from extending the generic taxonomical structure if needed. This is particularly important for the FCM, as it delivers situation-specific services on an ad-hoc basis (i.e. generated dynamically), and its concepts in such conceptualisation are usually not known in advance. Therefore, the generic taxonomical structure $PCE_\Delta T$ should have “just enough constraints” to enhance its semantics. Adding more concepts to the vertical taxonomical hierarchy, to remedy the lack of constraints, is not an option we advocate. The power of constraints in computing in general is rarely replaceable by the expansion of hierarchical structures, including vertical hierarchies, because they are simply different mechanisms of expressing semantics in computational models. They may be interchangeable, but not universally.

When looking at the axiomatisation of the taxonomical structure $PCE_{\Delta}T$, the reader would notice that we rarely used pre-enumerations, which has been a widely acceptable practice in SE. We have only pre-enumerated the `Pfc` root category, which represent preferences of the user. We have found that `Pfc` semantics is applicable to all situations in PCEs. We agree that pre-enumeration helps to control hierarchical levels in the generic taxonomical structure and its extensions, making it easier to manipulate the PCE semantics and eliminate excessive inference. Nevertheless, excessive pre-enumeration is dangerous in PCEs if we cannot predict the semantics of the situation in PCEs and manipulate them though reasoning.

We have also not included 'time' in $PCE_{\Delta}T$. PCEs, as described in Table 3.3, deal with one situation at a time and are not responsible for storing historical information. We do not advocate the use of persistence as in (Paganelli and Giuli, 2007), where a relational database is part of the ontology manager component. We have already specified that, considering P15, P16, and P17 in Table 3.3, situational information acquired for a particular moment might not be relevant for the next. Thus 'time' has no role in $PCE_{\Delta}T$ as a contributing factor, unlike examples from (Chen et al., 2004b) and (Stevenson et al., 2009).

6.2.1.2 Choice of Root Categories $RCtg_i.Lev_j$ in $PCE_{\Delta}T$

The proposed generic taxonomical structure $PCE_{\Delta}T$ has divided root categories $\mathtt{RCtg_i.Lev_j}$ into five occurrences: `Psn` (Person), `Ojt` (Object), `Fld` (Field), `Pfc` (preference), and `Lcn` (Location). These five $\mathtt{RCtg_i.Lev_j}$ are natural result of the five groups of characteristics of PCEs, as evident in Table 3.3. For "PCE and devices" and "PCE and computational/communication setting" groups in Table 3.3., we have chosen to use the root category `Ojt`, which stores all possible abstractions of

$\texttt{ins}_t$s of cyber and physical objects in any PCE_Δ. For the group “PCE and its users” (Table 3.3) `Psn`, and `Pfc` store all possible abstractions of $\texttt{ins}_t$s of preferences of users in any PCE_Δ. For “PCE and its performance” (Table 3.3) the category `Fld` stores all possible abstractions of $\texttt{ins}_t$s of domain-specific information in any PCE_Δ, and for “PCE and its situation” (Table 3.3) the category `Lcn` stores all possible abstractions of $\texttt{ins}_t$s of physical or cyber locations in any PCE_Δ. We remind the reader that in Definition 2 of situation in PCEs in Section 4.1.1, we note that location is not the only contributing factor in defining a situation. However, an intrinsic attribute of an object or person is their physical or cyber location which might be semantically important in a PCE_Δ. This was convincing enough merit for location to deserve the separate root category `Lcn` in the generic taxonomical structure. However, it is worth stressing that situation is represented by $PCE_\Delta T$, and not just an instance of $\texttt{RCtg}_i.\texttt{Lev}_j$ occurrence `Lcn.`

We would also like to add that occurrence `Fld` has some distinctive semantics comparing it with other occurrences of $\texttt{RCtg}_i.\texttt{Lev}_j$. It allows `Fld` to be different from the other occurrences of $\texttt{RCtg}_i.\texttt{Lev}_j$. P18 in Table 3.3 indicates that PCEs are domain-specific. This characteristic requires some generic information about the domain, which we depict with `Fld,` to be available irrespective of situations. One of the options is to allow programmers, who are using the proposed FCM to have a generic semantics of the domain, applicable to all application programs generated from the FCM, to be included in the ontological OWL model associated with the implementation of computations.

6.2.1.3 Natural Growing of the $PCE_\Delta T$

Any extension of the generic $PCE_\Delta T$ is natural, as it is domain and situation-specific. Real world instances create $PCE_\Delta T$ and therefore specialisation of the conceptualisation is entirely based on real situations. The selection of root categories $\texttt{RCtg}_i.\texttt{Lev}_j$ as explained in the previous section, and the generic relationships $\texttt{rlp}_r$ between them was also based on natural development of situations in PCEs. For example, for an occurrence `Lcn`, $\texttt{RCtg}_i.\texttt{Lev}_j \equiv \texttt{Lcn}$, a $\texttt{rlp}_r$ of `isIn` or `isCurrentlyIn` or both must exist. However, the location that a user is in or an object is at, is naturally either a cyber or physical location. Therefore, $\texttt{RCtg}_{\texttt{i}}\texttt{.Lev}_{\texttt{j}} \equiv \texttt{Lcn = Ctg}_{\texttt{i}}\texttt{.Lev}_{\texttt{m}}$ must have two $\texttt{Ctg}_{\texttt{i}}\texttt{.Lev}_{\texttt{j}}$, namely `Cyber-Lcn` and `Physical-Lcn`. In the current FCM these categories can be added as extension, but we believe they could have been included in the generic $PCE_{\Delta}T$. In other words the following axiom should have been included in section 4.1.

Axiom: If $\exists(x, y) \in \{PCE_{\Delta}T(x, y) \mid x \in INS\ , y \in RCtgi.Levj\ \equiv Lcn = Ctgi.Levm\} \Rightarrow$
$\exists(x, z) \in \{PCE_{\Delta}T(x, z) \mid z \in CTG.\lambda \equiv Cyber\text{-}Lcn = Ctgi.Levm_{-1}\} \cup$
$\{PCE_{\Delta}T(x, z) \mid z \in CTG.\lambda \equiv Physical\text{-}Lcn = Ctgi.Levm_{-1}\}$

One of the difficulties Chen et al. (2004b) encountered in developing ontologies was the use of “terms”. Although they have adopted common terms used in ‘consensus’ ontologies, such as Friend-Of-A-Friend ontology (Brickley and Miller 2003), they still had the difficulty of using terms in their SOUPA, and therefore their solution may not provide the right answer to software applications in some PCE that have adopted a different set of vocabularies. In our proposal the structure of the extended $PCE_{\Delta}T$ and the terms used are domain and situation-specific. The FCM complies with the SE practice of conceptualisation of a situation as high-level abstraction, and natural implementation of the situation as problem domain specific solution. In other words, names used for subsets of any $\texttt{RCtg}_{\texttt{i}}\texttt{.Lev}_{\texttt{j}}$ is based on situations and therefore we eliminated the problem of “terms” and “vocabulary” in our OWL ontologies.

Therefore, the FCM is not a fixed pre-defined model, like the one offered by Chen et al. (2003a) in their fixed context broker infrastructure CoBrA. In their ontology-based architecture, the ontology provides a set of ’terms’ for ’describing’ contextual data. It also allows common understanding of ’terms’ between distributed agents and reasoning, in order to derive additional knowledge about the pervasive space. It is also interesting to note, that Chen et al (2004b) developed a set of ‘core’ and ‘extension’ vocabulary ontologies. The fundamental difference between their model and the FCM is that in our proposal the $PCE_{\Delta}T$ in any situation $PCE_{\Delta}$is dynamically extended from the generic $PCE_{\Delta}T$ based on PCE_{Δ}, but in SOUPA all extended ontologies are prepared by application developers in advance.

6.2.1.4 Disjoint Extension of $Ctg_i.Lev_j$

It can be argued that, similar to the horizontal subsets of Ctg_i, $Ctg_i.Lev_j$, we may also have several vertical subsets at each Lev_j of a Ctg_i as shown in Figure 6.1.

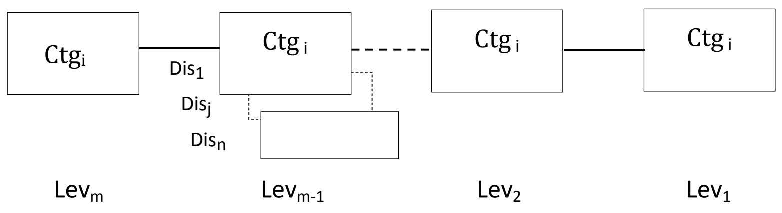


Figure 6.1:Vertical extension of $Ctg_i.Lev_j$

However, If we represent disjoint categories of each Lev_j of a Ctg_i as Dis_k, our argument is that although the "a subset of" relation still holds, the $Ctg_i.Lev_j.Dis_k$ and $Ctg_i.Lev_j.Dis_{k+1}$ do not have any other relation apart from the "a subset of" relation. Therefore, it is inconsequential to know whether it is Dis_k or Dis_{k+1} of the level Lev_j of the Ctg_i that the leaf category of Ctg_i is a classification of for a particular PCE_{Δ}. As long as the full path from the leaf to the root category is preserved, there is no need for any additional information, because it is irrelevant. Therefore, for any particular PCE_{Δ} it is sufficient and valid to claim that every category can be represented only by $Ctg_i.Lev_j$, where i, j $\in \mathbb{N}_0$. We would also like to stress that once we have agreed on the generic ontology in the implementation of the FCM, it cannot be vertically extended; i.e. no new disjoint concepts can be added to it. The reader should be reminded that we generate $PCE_{\Delta}T$ dynamically for every PCE_{Δ}. Otherwise, a complete pre-defined $PCE_{\Delta}T$ supporting all PCE_{Δ}s would be necessary. Nevertheless, a pre-defined $PCE_{\Delta}T$ contradicts the characteristics we have set for PCEs, therefore as far as we are concerned there is no other option but dynamic generation of $PCE_{\Delta}T$ for PCEs.

6.2.1.5 Depth of $Ctg_i.Lev_j$ Extension

The situational information determines how the generic $PCE_{\Delta}T$ should be extended to represent the knowledge about the situation, and to allow reasoning upon it to deliver situation-specific services to the user of the PCE. A real world instance $\mathtt{ins}_t$ is grasped by the PCE together with the category it belongs to, $\mathtt{Mbr(ins}_t\mathtt{,Ctg}_i\mathtt{.Lev}_1\mathtt{)}$, and $\mathtt{Mbr(ins}_t\mathtt{,Ctg}_i\mathtt{.Lev}_j\mathtt{)}$ for the particular 'i' and all available values of 'j' (Axiom 4. Section 4.1.4). Where exactly $\mathtt{Ctg}_i\mathtt{.Lev}_1$ is placed in the $PCE_\Delta T$ of a particular PCE_Δ depends entirely on the PCE_Δ itself. In other words, PCE_Δ determines the precision required for the particular $\mathtt{ins}_t$ in PCE_Δ. The greater the precision for the PCE_Δ, the deeper the position of $\mathtt{Ctg}_i\mathtt{.Lev}_1$. Therefore, the depth of $\mathtt{Ctg}_i\mathtt{.Lev}_1$ is situation dependent. This means that a particular $\mathtt{ins}_t$ may have different precision in various situations, but in any case the representation $\mathtt{Mbr(ins}_t\mathtt{,Ctg}_i\mathtt{.Lev}_1\mathtt{)}$ is valid. What varies is the distance between the leaf $\mathtt{Mbr(ins}_t\mathtt{,Ctg}_i\mathtt{.Lev}_1\mathtt{)}$ and its root $\mathtt{RCtg}_i\mathtt{.Lev}_j$, where $\mathtt{j=1..n}$. In other words the value of $\mathtt{j}$ in $\mathtt{RCtg}_i\mathtt{.Lev}_j$ can be 1 when the root and leaf are the same, and $\mathtt{n>1}$ when there is a distance between the root and leaf; the greater the $\mathtt{n}$, the further the distance.

6.2.2 Role of OWL in Defining THE FCM

We have provided a guideline to map each notion of taxonomical elements of $PCE_\Delta T$ to standard OWL concept in Table 4.3. As explained in 6.1.2.1 and 6.1.2.2, in spite of knowing that SWT will be used for implementing the computations generated from the proposed FCM, the definitions of the FCM terms are independent of OWL and any other programming languages of the SWT stack. In reality the FCM is independent from any other formal computing languages. Therefore, the abbreviations used for $PCE_\Delta T$ notions are absolutely arbitrary. However, the 'meaning' attached to them reflects our vision and philosophy of PCEs and summary of PCE characteristics from Table 3.3. Mapping these terms to OWL concepts was easy as both are based on hierarchical structures. The mapping in Table 4.3 is firm for the FCM. However, as mentioned in 6.2.1.1, the FCM and the generic $PCE_\Delta T$ can be modelled differently. Modifications to generic $PCE_\Delta T$ might require mappings and therefore, Table 4.3 is not 'the' table required for transformation of any FCM in PCEs to OWL.

6.2.3 Implementation

6.2.3.1 SWRL enabled OWL Ontology

The choice of OWL sublanguages within the SWT stack is impressive. Out of the three sublanguages: OWL Lite, OWL DL and OWL Full, we have used OWL DL. Considering that the purpose of the computation in PCEs is to reason upon taxonomical elements of $PCE_{\Delta}T$ to deliver a service to the user of a PCE, the variant of OWL to be chosen should support SWRL as the reasoning language used in the SWT stack. This requirement automatically dismissed OWL Full, which was not a suitable candidate because of its unrestricted expressivity. OWL Full allows restrictions to be defined at the "meta level" and therefore types, such as classes and individuals, are not separated from each other. Elements of $PCE_{\Delta}T$, on the contrary, are clearly separated from each other and therefore OWL Full could not be suitable for the mapping outlined in paragraph 6.2.2.

OWL Lite was also disqualified because it would not support decidable computation. OWL DL, which supports DL, as a decidable fragment of first order logic that SWRL is also based on, was obviously a preferred language which can be mapped to hierarchies of $PCE_{\Delta}T$.

Although Protégé 4 is user-friendly and the most commonly-used open source ontology editor, it is still an incomplete and somehow unstable tool. The obvious example is the "tab" provided in Protégé 4, for editing SWRL rules. There is a limit to the number of atoms one can employ in each rule; if the number exceeds the limit, a number of the "consequences" in the rule, would be literally omitted from the rule's syntax. Furthermore, editing case-sensitive SWRL built-in functions, used in our earlier experiments, are also problematic. For example, although we observed the camel-casing convention for naming built-in functions when we wrote functions in SWRL rules, Protégé editor changed the format and consequently it did not realise it as a built-in function.

With regards to the development of a software application in IDEs, such as NetBeans, using SWRL enabled OWL ontology, we would like to emphasise that establishing the communication between the application and ontology and the Pellet reasoner was not a direct task. The use of OWL-API interface to link the Application Layer of ASeCS (Figure 5.3) with Inference and Reasoning Layers, and Ontology Layers was not straightforward. There is still a lack of supporting online documentation to guide developers how to create applications based on SWRL enabled OWL ontologies.

Although in real life situations all SWRL rules are usually defined in advance and stored with ontologies, such as GOnto in our case, we have also experienced how SWRL rule can be defined, created and executed through the ASeCS Application Layer at run time once SeCHOnto has been created. Readers should notice that the result of the inference and reasoning of a situation PCE_{Δ} is just for the moment, when the situation occurs, and as soon as another change in PCE_{Δ} is detected, the result of the reasoning related to the previous moment has to be deleted. This is because the inference/reasoning of a particular situation PCE_{Δ} might not be exactly a correct contextual information or suitable for another situation in the PCE. The behaviour of Protégé editor towards changes of situation in our solution depends on how the OWL ontology file is accessed. If changes are modified locally (within the editor), the update is handled automatically without any need for user interaction. However, when an OWL file is accessed by an application through OWL API it is "assumed" that ontology has changed outside of Protégé, therefore the programmer needs to confirm whether the ontology is to be reloaded. Otherwise, the changes of situation would be disregarded.

This feature of Protégé supports our view of PCEs that each semantics of PCE_{Δ} should be treated separately. This means that each time a change is detected in a PCE, applications generated from the FCM must reload GOnto and disregard situation-specific SeCHOnto once the reasoning about the situation which delivered a service is done. Nevertheless, accessing GOnto from outside of Protégé to extend it to SeCHOnto requires somewhat frustrating interaction with confirmation dialog boxes, because the current version of Protégé does not support its OWL file to be handled by applications automatically.

6.2.3.2 Mapping CQ to the FCM

The verification of ontologies, in terms of addressing if they really represent what they are expected to, is usually carried out by Semantic Web designers through CQs. However, we used a CQ in a situation PCE_{Δ}, to establish the situational information necessary for a reasoning mechanism to to deliver a service to the user in the situation PCE_{Δ} . Interpretation of the CQ and its conversion into SWRL rule is within the realm of natural language processing, and well outside the scope of this thesis. We have, however, rephrased, formulated, and stratified the example scenario CQ to segments, each of which qualify as a distinctive situational information. These segments, mapped to formal representation of the $PCE_{\Delta}T$ as shown in Table 5.2, collectively form the premises (or atoms) of the SWRL rule. The natural transformation of the CQ to the creation of a situation to be reasoned upon to deliver a service, facilitates computation with minimal resources and hence the applicability of the FCM to be implemented on handheld mobile devices.

6.2.3.3 Mapping Situation-specific CQ to SWRL Rule

In Chapter 5, we have focused on the extension of GOnto to represent all the premises of the SWRL rule related to the particular PCE_{Δ} of the example scenario and its CQ. The consequent of running the rule, which is "to turn the heater on", `ToBeTurnedOnObject(?h)`, needs to be also represented in the ontology. This means that `Heater(?h)`which is currently "off", `status(?h,"off"),` will change its status from "off" to "on" as a result of the specified action. Reader should bear in mind that all the computation is for a particular situation and the status of a device cannot be "off" and "on" simultaneously. Hence, the need for the separate class `ToBeTurnedOnObject.` Pre-enumeration of `OBJECT` or `PERSON` to have classes such as `ToBeTurnedOnObject, ToBeTurnedOffObject, ToBeAlertedPerson, ToBeMonitoredPerson` is an alternative that can be looked into in future work. Although we have shown that only one SWRL rule is used in Chapter 5 to reason about a situation PCE_{Δ}, we have used several rules in our previous experiences (Shojanoori et al., 2010; 2012; Shojanoori and Juric, 2013). Executing several rules simultaneously, even when we were rule chaining, was not expensive in terms of resource usage and software application response time.

6.2.3.4 Role of Traditional Computing in Realisation of PCEs

In Chapter 1 we posed the question of whether we can integrate skills and experiences of the traditional computing with computations generated from the FCM, in order to enable inference and reasoning mechanisms of PCEs. Such systems are expected to support knowledge representation of situations, and to reason upon it to deliver situation-specific services to users of PCEs. The use of Semantic Web technologies through IDEs such as NetBeans, as illustrated in Chapter 5 (screen shots available in Appendix A and application programme available in Appendix B) shows that we are able to extend the same mechanism of manipulating the semantics of the Web towards any other form of computations in SE, which does not have to be related the Internet. However, we have to bear in mind that traditional software technologies, including Java technologies, cannot manage PCEs without reference to SWT. Pervasive software applications also cannot rely only on procedural or object-oriented programming languages alone to address requirements of PCEs. Managing them though heavy and elaborative knowledge base systems and making them dependent on constantly growing persistence would not satisfy a fraction of expectations we have from PCEs.

We are now in a position to debate or answer some of the question in Chapter 1. The most intriguing one would be to establish exactly where “computing” starts in our PCEs? Where do we start “computing” if we must deliver services, according to the computations derived from the proposed FCM? Is it when we start executing SWRL? Is it when we extend the generic taxonomical structure in order to accommodate specificity of a situation in PCE? Is it when we start defining constraints in the taxonomical structure? All three of these may be associated solely with reasoning mechanisms and inference, but at the same time they could legitimately correspond to computations in various software applications. Considering that the implementation of the proposed FCM in software applications which deliver services in PCEs is feasible, and their running is commercially viable, the question of where exactly we compute in PCE might be immaterial? Furthermore, we would not like sideline Java computational power, or marginalise it, by moving the complexity of computational Java code required in PCE into OWL ontologies and SWRL rules. However, we demonstrated that the FCM delivers exactly what we need in PCEs with SWT. If the FCM implementation with SWRL enabled OWL ontologies gives a viable result, and then we should look at the proposed architectural solution given through ASeCS and assume that new types of computations, stored within its architectural core components cannot be ignored.

6.3 Contribution

The proposed FCM in Figure 4.10 is independent of any software technology or programming language. Although we are confident of the applicability of the FCM across domains, our intention was to prove that the proposed FCM can work and be implemented in a particular PCE. That said, while preserving the fundamentals of the FCM, the generic taxonomical structure of $PCE_{\Delta}T$ could be altered by introducing more semantics (axioms) should we be in a situation to revisited it.

The proposed FCM gives instructions on how and which situational information we need to have. In other words, it is the situation that manages the PCE. The domain specificity of a PCE indicates which devices and contextual information we may have in a PCE, but it is the situation that specifies which one of them will be “selected” for the situation. The Generic $PCE_{\Delta}T$ guarantees the creation of any situation and includes its extension if required.

The FCM allows grows of $PCE_{\Delta}T$ exactly according to expectations a user may have in a situation PCE_{Δ}. This means that the FCM supports the “extend as you go” policy because its $PCE_{\Delta}T$, that makes use of the powerful “is-a”(a subset of) relationship, is the taxonomical structure of $\mathtt{Mbr(ins_t,Ctg_i.Lev_j)}$as real world participants in a PCE_{Δ}. Considering that the participation of $\mathtt{Mbr(ins_t,Ctg_i.Lev_j)}$is determined by situations, $PCE_{\Delta}T$ grows naturally the way we, as users of PCE, want it. Therefore, the FCM guarantees that in PCEs users are and stay really in control.

Considering that $PCE_{\Delta}T$ grows naturally, based on PCE_{Δ}, the computational model representing the semantics of the situation and infers new knowledge and reason upon situation is always correct. This entails that the precision of abstraction is arbitrary and depends on the situation. This humble, but efficient way of knowledge representation secures expected delivery of services to users of PCEs.

We have to add also the way we address unintentionally one of the major challenges to the information systems community is information overloading. The challenges presented by big data inspired the Semantic Web vision. Even though PCE is in a different league compared to SW, the concept of information overloading applies to it more than ever. The amount of information in PCEs is too much to be bearable by any human brain and to some extent by computing processors, if a timely response is desired. Limiting the computation about any situation to the situational information that contributes to the situation in PCE is the solution supported by the FCM. This improves inference and reasoning processes and generates a response to the situation in a timely manner.

6.4 Research Conclusion and Future Work

This research was a demonstration of the fact that we can use SWT for other forms of computation than they were originally designed for. We believe this research and its result will give a pause of thought to the pervasive and ubiquitous computing community to address the true nature of PCEs. We have achieved all the objectives set at the beginning. It was a demanding but pleasantly rewarding body of research. We are pleased with the results of our research and sincerely hope its contribution to knowledge paves the way for more powerful, user-centric PCEs, the way we have defined them.

We have shown the role of situation in PCEs and created a common set of PCE characteristics that helped us in achieving our objectives of the research including the creation of a formal computational model FCM, which can be deployed using SE principles and modern software technologies. The FCM advocates formal representation of situational information, for which current situational information is meaningless for the next moment, or as worthless as yesterday's newspaper. The FCM supports extensibility of devices in PCEs and an extension of the formal model according to the situation. PCEs do not have boundaries; there is no need to define the "scope" or "boundary" of a PCE. Situations of a PCE are dynamically created.

This means that boundaries are also changing dynamically. For that reason, a PCE boundary is defined for every situation at the time the PCE_{Δ} is created. In other words, It is the situational information that defines the boundary of a PCE and not the other way round. Software solutions using programming languages and technologies of traditional computing in conjunction with SWT are instrumental to interface with these pervasive devices. This means that we will ultimately be able to run the formal computational model FCM also on mobile and handheld devices.

There is a scope for improvements and future works:

- A PCE must allow its extension, removal and replacement of devices without any restriction. This requires devices to be self-maintaining in terms of the meaningful data they provide. The integration of devices in PCE and their management is another area which would need attention of the FCM.
- We should be looking at implementations of the ASeCS architectures in mobile environments using Android, iOS and similar operating systems. The lightness of our computational solution generated from the FCM is encouraging. It remains to be seen how we can maintain the MVC pattern, so prevalent in modern computing, and dynamic environments of apps by using the FCM.
- We need to explore the efficacy and efficiency of software application based on computations generated from the FCM. All our experiments are very encouraging, but the possible commercialisation of the proposal will require more attention to interfaces and connection to the interpreted contextual data and information which feed the FCM. Consequently, we have to add that we should be able to compare the response time of two PCE systems, one developed with a fixed conceptual model and the other as we proposed in this research.

Considering the lack of published documentation on use of OWL-API for software applications to communicate with OWL ontologies and SWRL rule engines, we intend to provide a guide to help students and researchers to spare the time on their design ideas.

References

Abhishek, S., Michael, C. (2006) '*Survey of Context aware Frameworks- Analysis and Criticism*', (Online) article, *UNC-Chapel Hill ITS, The University of North Carolina*.

Adlam, T. and R. Orpwood (2004) '*Taking the Gloucester Smart House from the Laboratory to the Living Room*', 3rd International Workshop on Ubiquitous Computing. Atlanta Georgia.

Akyildiz, I. et al. (2002) '*A Survey on Sensor Networks*', IEEE Communication Magazine, Vol. 40, No. 8, pp. 102-14.

Albrecht Schmidt, Michael Beigl, Hans-W Gellersen (1999) '*There is more to context than location*', Computers &amp; Graphics, 23, Issue 6, pp. 893-901.

Ark, W. and Selker, T. (1999) '*A look at human interaction with pervasive computers*', IBM Systems Journal special HCI issue with a focus on Pervasive Computing, 38(4), pp. 504-507.

Arnrich, B., Mayora, O., Bardram, J. and Tröster, G. (2010) '*Pervasive healthcare: Paving the way for a pervasive, user-centered and preventive healthcare model*', Methods of Information in Medicine, 49:1, pp. 67-73.

Ayala, C.P., Cruzes, D.S., Hauge, O., Conradi, R. (2011) '*Five Facts on the Adoption of Open Source Software*', Software, IEEE, Vol. 28, No. 2, pp. 95-99.

Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (2007) '*The Description Logic Handbook: Theory implementation and applications*', 2nd Edition.

Bacon, J. (2002) '*Toward Pervasive Computing*', IEEE Pervasive Computing, 1 (2), p. 84.

Bahrami, A., Yuan, J., Smart, P., Shadbolt, N.R.(2007) '*Context Aware Information Retrieval for Enhanced Situation Awareness*',IEEE Military Communications Conference, pp.1-6.

Baldauf, M., Dustdar, S. and Rosenberg, F. (2007) '*A survey on context-aware systems*', Int. Journal of Ad Hoc and Ubiquitous Computing, 2(4), pp. 263–277.

Banavar, G., Beck, J., Gluzberg, E., Munson, J., Sussman, J., and Zukowski, D. (2000) '*Challenges: an application model for pervasive computing*', In Proceedings of the 6th Annual international Conference on Mobile Computing and Networking, Boston, Massachusetts, United States, MobiCom '00. ACM, New York, NY, pp. 266-274.

Bardram, J.E. and Christensen, H.B. (2007) '*Pervasive Computing Support for Hospitals: An overview of the Activity-Based Computing Project*', IEEE Pervasive Computing, 6:1, pp. 44-51.

Barrett, K., Power, R. (2002) '*State of the Art: Context Management*', M-Zones Deliverable, pp. 69-87.

Berners-Lee, T., Hendler, J. and Lassila, O. (2001) '*The Semantic Web. Scientific American*', 284, pp. 34-43.

Berners-Lee, T. (1989) *Information Management: A Proposal*, W3 Archieve, Available at http://www.w3.org/History/1989/proposal.html.

Berners-Lee, T. (2001) Weaving the Web: *The Past, Present and Future of the World Wide Web by its Inventor*, Harper Collins.

Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A, Riboni, D. (2010) '*Survey of Context Modelling and Reasoning Techniques*', Pervasive and Mobile Computing, 6, Issue 2, pp. 161-180.

Bharucha, J. A. (2006) *'CareMedia: Automated Video and sensor analysis for Geriatric care*'.

Bizer, C., Mendes, P.N. and Jentzsch, A. (2012) *Introduction to Web of Data*. *In:* Roberto. De Virgilio, Francesco. Guerra, Yannis. Velegrakis, eds. Available at http://books.google.co.uk/books?id=aDTDXH0-i-oC&pg=PC4&lpg=PC4&dq=Google,+Yahoo+and+Microsoft+have+agreed+on+vocabularies+for+publishing+structured+data+on+the+Web.&source=bl&ots=OWy8pO0mh2&sig=WjqYx8_Z_i7HEKrZtt1aG4ZfkxM&hl=en&sa=X&ei=AJViUe-GGsXmOZW6gOgH&sqi=2&ved=0CEYQ6AEwBA#v=onepage&q=google&f=false.

Bohn, J., Gartner, F. and Vogt, H. (2004) '*Dependability issues of Pervasive Computing in a healthcare Environment*'.

Bolchini, C., Curino, C. A., Quintarelli, E., Schreiber, F. A., and Tanca, L. (2007) '*A data-oriented survey of context models*', SIGMOD Rec. 36, 4, pp. 19-26.

Boyaci, O., Martinez, V. B., Schulzrinne, H. (2012) '*Bridging Communications and the physical world*'.

Brezillon, P. (2011) '*From expert systems to context-based intelligent assistant systems: a testimony*', Engineering, 26, pp. 19-24.

Brickley, D., Guha, R.V. (2000) *Resource Description Framework (RDF) Schema Specification 1.0: W3C Candidate Recommendation*.

Brickley, D., Miller, L. (2010).*FOAF Vocabulary Specification 0.98.* Namespace Document, 9 August 2010 - Marco Polo Edition. Also available at http://xmlns.com/foaf/spec/.

Brown, P.J., Bovey, J.D., Chen, X. (1997) '*Context-Aware Applications: From the laboratory to the Marketplace IEEE Personal Communications*', 4,? New York: IEEE, pp. 58-64.Cambridge University Press.

Byun, H. E. and Cheverst, K. (2004) '*Utilizing context history to provide dynamic Adaptations*', Applied Artificial Intelligence, 18(6), pp. 533-548.

Campbell, R, Al-Muhtadi, J, Naldurg, P, Sampemane, G, Mickunas, M (2002) '*Towards security and privacy for pervasive computing*', ISSS, Tokyo, Japan, pp. 1-15.

Chen, H., Finin, T., Joshi, A. (2003) '*An ontology for context-aware pervasive computing environments*', Special Issue on *Ontologies for Distributed Systems, Knowledge Engineering Review*, 18(3), pp. 197-207, Available at: http://ebiquity.umbc.edu/_file_directory_/papers/63.pdf.

Chen, H. and Finin, T. (2003) '*An Ontology for Context-aware Pervasive Computing Environments*'.

Chen, H., Finin, T., Joshi, A. (2003b) '*Using OWL in a Pervasive Computing Broker*'.

Chen, G. and Kotz, D. (2000) '*A survey of context-aware mobile computing research Technical report*', Dept. of Computer Science, Dartmouth College. Available at: http://www.cs.dartmouth.edu/~dfk/papers/chen:survey-tr.pdf.

Chen, H., Perich, F., Finin, T. W. and Joshi, A. (2004b) '*SOUPA: Standard ontology for ubiquitous and pervasive applications*', Proceedings of the First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous'04), IEEE Computer Society, pp. 258-267.

Chen, H., Finin, T., Anupam, J. (2003d) '*Semantic Web in a Pervasive Context Aware Architecture', In* Proceedings of Artificial Intelligence in Mobile System.

Chen, Y., Xu, J., Tang, Y., Fang, Z. (2011) ‘*Research on smart space oriented context-aware system*’, Electric Information and Control Engineering (ICEICE), International Conference. pp. 698-700.

Coen, M. (1998) ‘*Design Principles for Intelligent Environments*’, Available at: http://citeseer.ist.psu.edu/cache/papers/cs/2468/http:zSzzSzwww.ai.mit.eduzSzpeoplezSzmhcoenzSzIEsymposium.pdf/coen98design.pdf.

Coronato, A. and Pietro, G.D.E. (2010) ‘*Formal specification of wireless and pervasive healthcare applications*’, ACM Tansaction in Embedded Computing Systems, 10:1, Article 12.

Coutaz, J., Crowley, J. L., Dobson, S. and Garlan, D. (2005) ‘*Context is Key Communications of the ACM*’, 48(3), pp. 49-53.

Crawford, D. (2000) ‘*Editorial note, Communications of the ACM*’, 43, No. 3.

Damien, C., Benjamin, B., Nicolas, L. and Charles, C. (2009) ‘*A generative programming approach to developing pervasive computing systems*’, SIGPLAN Not. *45, 2, pp. 137-146.* Available at:
http://doi.acm.org/10.1145/1837852.1621629.

Declan, S. and David, L. (2003) ‘*Semantically driven service interoperability for pervasive computing*’, In Proceedings of the 3rd ACM international workshop on Data engineering for wireless and mobile access (MobiDe '03). ACM, New York, NY, USA, pp. 17-24.

Dey, A. K., Abowd, G. D. and Salber, D. (2001) ‘*A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications*’, Hum.-Comput. Interact. 16, 2, pp. 97-166.

Dey, A., Abowd, G. (1999) ‘*Towards a Better Understanding of Context and Context-Awareness*’, In proceedings of 1st International Symposium on Handheld and Ubiquitous Computing (HUC '99)1707, pp. 304-307, Available at: ftp://ftp.cc.gatech.edu/pub/gvu/tr/1999/99-22.pdf.

Dey, A. K., Abowd, G. D., Wood, A. (1999) ‘*CyberDesk: A Framework for Providing Self-Integrating Context-Aware Services*’, Knowledge-Based Systems, 11, pp. 3-13.

Dey, A. K. (1998) ‘*Context-Aware Computing: The CyberDesk Project*’, AAAI 1998 Spring Symposium on Intelligent Environments, Technical Report SS-98-02, pp. 51-54.

Dey, A. K. (2001) ‘*Understanding and using context*’, Personal and Ubiquitous Computing, 5, No.1, pp. 4-7.

Eiter, T., Ianni, G., Krennwallner, T. and Polleres, A. (2007) ‘*Rules and Ontologies for the Semantic Web*’.

Ellenberg, J., Karstaedt, B., Voskuhl, S., Luck, K.V, and Wendholt, B. (2011) ‘*An environment for context-aware applications in smart homes*’, in to appear in: International Conference on Indoor Positioning and Indoor Navigation (IPIN), Guimaraes, Portugal.

Fahy, P. and Clarke, S. (2004) ‘*CASS – a middleware for mobile context-aware applications*’, Workshop on *Context-Awareness, MobiSys*.

Garlan, D., Siewiorek, D.P., Smailagic, A., Steenkiste, P. (2002) *’Project Aura: toward distraction-free pervasive computing*’, Pervasive Computing, IEEE. 1, No. 2, pp. 22-31.

Gauvin, M., Boury-Brisset, A.C., Auger, A. (2004) *‘Context, ontology and portfolio: key concepts for a situational awareness knowledge portal’*. System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on System Sciences. o., pp.1-10.

Gellersen, H.W., Schmidt, A., Beigl, M. (2002) ‘*Multi-Sensor Context-Awareness in Mobile Devices and Smart Artefacts*’, ACM journal Mobile Networks and Applications (MONET), Vol. 7, No. 5.

Gibson, W. (1984) ‘*Neuromancer*’, London : HarperCollins, 1995.

GO (2000)*, Gene Ontology, The Gene Ontology Consortium: tool for the unification of biology*. Nature Genet, 25, pp. 25-29.

Grau, B. C., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P. and Sattler, U. (2008) ‘*OWL 2: The next step for OWL*’, Web Semant. 6, 4, pp. 309-322.

Gregory, D. A. and Elizabeth, D. M. (2000) ‘*Charting past, present, and future research in ubiquitous computing*’, ACM Trans. Comput.-Hum. Interact. 7, 1, pp. 29-58.

Gregory, D. A., Christopher, G. A., J. H., Sue Long, R. K., and Pinkerton, M. (1997) ‘*Cyberguide: a mobile context-aware tour guide*’, Wirel. Netw. 3, 5, pp. 421-433.

Grimm, R. (2004) '*One world: Experiences with a Pervasive Computing Architecture*', IEEE Pervasive Computing 3, pp. 22-30.

Gruber, T.R. (1993) '*A Translation Approach to Portable Ontology Specification*', Knowledge Acquisition 5(2), pp. 199-220.

Gu, T., Wang, X., Pung, H. and Zhang, D. (2004) '*An Ontology-based Context Model in Intelligent Environments*', In Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference, San Diego, California, USA.

Harter, A., Hopper, A., Steggles, P., Ward, A., Webster, P. (1999) '*The Anatomy of a Context-Aware Application*', Available at: http://citeseer.ist.psu.edu/cache/papers/cs/10242/ftp:zSzzSzftp.uk.research.att.comzSzpubzSzdocszSzattzSztr.1999.7.pdf/harter99anatomy.pdf.

Hartela, F., W., Coronadoa, S., Dionneb, R., Fragosoa, G., Golbeckc, J. (2005) '*Modelling a description logic vocabulary for cancer research*', Journal of Biomedical Informatics, 38, pp. 114-129.

Helal, S. (2011) '*Programming Pervasive Spaces: Workshop on Semantic Interoperability in Smart Spaces*', Applications and the Internet (SAINT), IEEE/IPSJ 11th International Symposium. pp. 279-282.

J. Bourcier, A. Diaconescu, P. Lalanda, and J. A. McCann. (2011) '*AutoHome: An autonomic management framework for pervasive home applications.*' ACM Transactions on Autonomous and Adaptive Systems Vol.6(1), pp.1-10 .

Hellenschmidt, M. (2006) '*Some Issues on Requirements for Pervasive Software Infrastructures*', Proceedings of the 1st International Workshop on Requirements and Solutions for Pervasive Software Infrastructures (RSPSI 2006), Dublin, Ireland, pp. 549-552.

Henricksen, K., Indulska, J. (2004) '*A Software Engineering Framework for Context-Aware Pervasive Computing*', Available at:

http://citeseer.ist.psu.edu/cache/papers/cs2/61/http:zSzzSzhenricksen.id.auzSzpublicationszSzPerCom04.pdf/henricksen04software.pdf.

Henricksen, K. and Indulska J. (2006) '*Developing Context-Aware Pervasive Computing Applications: Models and Approach*', Pervasive and Mobile Computing, 2, Issue 1, pp. 37-64.

Henricksen, K., Indulska, J., McFadden, T. (2005) *'Modelling Context Information with ORM*', Available at: http://henricksen.id.au/publications/ORM05.pdf.

Henricksen, K., Livingstone, S., Indulska, J. (2004) '*Towards a Hybrid Approach to Context Modeling, Reasoning and Interoperation*', Proc. of the 1st International Workshop on Advanced Context Modeling, Reasoning and Management, pp. 54-61.

Henricksen, K., Indulska, J., McFadden, T. and Balasubramaniam, S. (2005) '*Middleware for distributed context-aware systems*', Proc. of On the Move to Meaningful Internet Systems, LNCS 3760, pp. 846-863.

Henrickcon, K., Indulska, J. and Rakotoniraniny, A. (2002) '*Modeling context information in pervasive computing systems*', In Proceedings of the 1st International Conference on Pervasive Computing Systems, 2414 of Lecture Notes in Computer Science, pp. 167-180, Zurich, Switzerland, Springer Verlag.

Hofer, T., Schwinger, W., Pichler, M., Leonhartsberger, G. and Altmann, J. (2002) '*Context-Awareness on Mobile Devices – the Hydrogen Approach*', In Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS'03).

Hong, J., Suh, E. and Kim, S. (2009) '*Context-aware systems: A literature review and classification*', Expert Systems with Applications, 36(4).

Horrocks, I., Patel-Schneider P., Boley, H., Tabet, S., Grosof, B., Dean, M. (2009) '*SWRL Built-ins*', (online) Available at:
Avahttp://www.daml.org/2004/04/swrl/builtins.html.

Horrocks, I., Parsia, B., Patel-Schneider, P. F. and Hendler, J. (2005) '*Semantic web architecture: Stack or two towers?*', In Proc. PPSWR 2005, pp. 37-41, Dagstuhl Castle, Germany.

Hwang, G. J., Chu, H. C., Lin, Y. S., and Tsai, C. C. (2011) '*A knowledge acquisition approach to developing Mindtools for organizing and sharing differentiating knowledge in a ubiquitous learning environment*', Computers and Education, 57(1), pp. 1368-1377.

IBM (2001) '*Autonomic Computing Manifesto'*.
Available at: http://www.research.ibm.com/autonomic/manifesto/.
Intille, S. S. (2002) '*Designing a Home of the Future*', *IEEE Pervasive Computing* 1, 2, pp. 76-82.

Intel (2000) '*Intel's Computing Continuum Conference to Explore New Era of Computing, Communication and Interaction in the Digital World*', Available at: http://www.thefreelibrary.com/REMINDER%2FIntel's+Computing+Continuum+Conference+to+Explore+New+Era+of...-a060057131.

Intille, S. S. (2002) '*Designing a Home of the Future*', IEEE Pervasive Computing, pp. 76-82.

Intille, S. S. (2006) '*The Goal: Smart People, Not Smart Homes*', Massachusetts Institute of Technology, Cambridge, USA.

Jahnke, J. H., Bychkov, Y., Dahlem, D. and Kawasme, L. (2004) *'Implicit, contextaware computing for health care*', In Proceedings of the 1st Int'l Workshop on Modeling and Retrieval of Context.

Jahnke, J. H., Bychkov, Y., Dahlem, D. and Kawasme, L. (2004) '*Implicit, contextaware computing for health care*', In Proceedings of the 1st Int'l Workshop on Modeling and Retrieval of Context (MRC 2004). Schulz, S. and Roth-Berghofer, Th. (Eds.), CEUR Workshop Proceedings, 114.

Jesse, W. (2011) *More about Facebook Linked Data*, W3C Archive, Available at: http://lists.w3.org/Archives/Public/public-lod/2011Oct/0032.html.

Kadak, T. and Kleerova, O. (2006) '*OWL – Ontology Web Language'*, Available at: http://www.pafis.shh.fi/~trikad06/RM.doc.

Kephart, J. O. and Chess, D. M. (2003) '*The vision of autonomic computing*', IEEE Computer, pp. 41-50.

Khalil, I., Ali, F.M., Kotsis, G. (2008) '*A Datalog Model for Context Reasoning in Pervasive Environments*', ISPA '08. International Symposium on Parallel and Distributed Processing with Applications. pp.452-459.

Khedo, K. K. (2006) *'Context-aware systems for mobile and ubiquitous networks*', International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies, p. 123.

Khedr, M. and Karmouch, A. (2004) *'Negotiating Context Information in Context-aware Systems*'.

Kjær, K.E. (*2007*) *'A survey of context-aware middleware',* Proceedings *of 5th* Conference *on* IASTED *International Multi-*Conference*:* Software Engineering*, pp. 148–155.*

Ko, E. J., Lee, H. J. and Lee, J. W. (2007) '*Ontology-Based Context Modeling and Reasoning for U-HealthCare*', IEICE - Trans. Inf. Syst. E90-D, pp. 1262-1270.

Koay, N., Kataria, P., Juric, R. (2010a) '*Semantic Management of Non-Functional Requirements in e-Health Systems*', in Tele-Medicine and e-Health Journal, 2010 May;16(4): pp.461-71

Koay, N., Syal, P., Juric, R. (2010b) '*Reasoning in Pervasive Computational Spaces*', in proceedings of the 17th Automated Reasoning Workshop, University of Westminster, March 30-31, available at :
http://www2.wmin.ac.uk/bolotoa/ARW/arw-2010.html

Korhonen, I., Paavilainen, P., Sarela, A. (2003) '*Application of ubiquitous computing technologies for support of independent living*'.

Korpipää, P. Mantyjarvi, J., Kela, J., Keranen, H. and Malm, E-J. (2003) '*Managing context information in mobile devices*', IEEE Pervasive Computing, 2, No.3,pp. 42-51.

LI, F., Sanjin, S., and Schahram, S. (2010) '*COPAL: An adaptive approach to context provisioning*', In Proceedings of the IEEE 6th International Conference on Wireless and Mobile Computing, Networking, and Communications (WiMob'10), pp. 286-293.

Lukowicz, P., Pentland, S., Ferscha, A. (2012) *'From context awareness to socially aware computing*', Pervasive Computing, Vol. 11, Issue 1, pp. 32-41.

Lyons, M. (2002) '*Pervasive Computing – Control and Freedom in Cyberspace*', ITS 14th Biennial Conference, Seoul, Korea

Matheus, C. (2005) *'Using ontology-based rules for situation awareness and information Fusion'*, W3C Work on Rule Languages for Interoperability.

Matheus, C.J., Kokar, M.M. and Baclawski, K. (2003) *'A Core Ontology for Situation Awareness'*. In Proceedings of the Sixth International Conference on Information Fusion, pages 545 –552.

McDermott, D. (1981) '*Artificial intelligence meets natural stupidity*', Mind Design: Philosophy, Psychology, Artificial Intelligence (Haugeland, J., Ed.) MIT Press, Cambridge, MA, pp. 143-160.

McFadden, T., Henricksen, K., Indulska, J. and Mascaro, P. (2004) '*P. Applying Disciplined Approach to the Development of a Context-aware Communication Application*'.

McGuinness, D.L., Harmelen, F.V. (2003) '*OWL web Ontology Language Overview* ', Available at: www.w3.org.

Mehra, P. (2012) '*Context-aware Computing Beyond Search and Location-based Services*', IEEE Internet Computing, pp. 12-16.

Mike Uschold and Michael Gruninger (1996). *Ontologies: principles, methods and applications*. The Knowledge Engineering Review, 11, pp 93-136.

Milosevic, M., M. Shrove, T., Jovanov, E. (2011) '*Applications of Smartphones for Ubiquitous Health Monitoring and Wellbeing Management*', Journal of Information Technology and Application (JITA).

Miraoui, M., Tadj C. and Amar, C. B. (2008) '*Architectural Survey of Context-Aware Systems in Pervasive Computing Environment*', Ubiquitous Computing and Communication Journal, 3, No. 3.

Mungall, C.J., Gkoutos, G. V., Smith, C.L., Haendel, M. A., Lewis, S. E., Ashburner, M. (2010) '*Integrating phenotype ontologies across multiple species*', Genome Biol, 8, 11(1).

Mühlhäuser, M. and Iryna G. (2008) '*Introduction to Ubiquitous Computing*', Handbook of Research on Ubiquitous Computing Technology for Real Time Enterprises, ed. Max, M. and Iryna, G., pp. 1-20.

Niyato, D., Hossain, E., Camorlinga, S. (2009) '*Remote Patient Monitoring Service using Heterogeneous Wireless Access Networks:* Architecture and Optimization', IEEE Journal on Selected Areas in Communications, 27:4, pp. 412-423.

Nguyen, T. V., Deok-Jai, C. (2008) '*Context Reasoning Using Contextual Graph*', Computer and Information Technology Workshops, 2008. CIT Workshops, IEEE 8th International Conference on Computer and Information Technology Workshops. pp.488-493.

Norman, D.A. (1999) '*The Invisible Computer*', Cambridge, MA: The MIT Press.

Ogden, C. K. and Richards, I. A. (1923) '*The Meaning of Meaning: A Study of the Influence of Language Upon Thought and of the Science of Symbolism*', London: Routledge and Kegan Paul.

Paganelli, F., Bianchi, G. and Giuli, D. (2006) '*A Context Model for Context-aware System Design towards the Ambient Intelligence Vision: Experiences in the eTourism Domain*', in Proc. of 9th ERCIM Workshop "User Interfaces For All", Special Theme: "Universal Access in Ambient Intelligence Environments", Königswinter (Bonn), Germany.

Paganelli, F. and Giuli, D. (2007) '*An Ontology-Based Context Model for Home Health Monitoring and Alerting in Chronic Patient Care Networks*', Proceedings of the 21st international Conference on Advanced information Networking and Applications Workshops, AINAW. IEEE Computer Society, Washington DC, pp. 838-845.

Pascoe, J. (1998) '*Adding Generic Contextual Capabilities to Wearable Computers*', 2nd International Symposium on *Wearable Computers*, pp. 92-99.

Pascoe, J. (1998) '*Adding generic contextual capabilities to wearable computers*', Wearable Computers, 1998. Digest of Papers. Second International Symposium. pp.92-99.

Polze, A., Tröger, P., Hentsche, U., Heinze, T. (2010) '*A scalable, self-adaptive architecture for remote patient monitoring'*. 13th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops', pp. 204-210.

Ranganathan, A. and Campbell, R. (2003) '*An infrastructure for context-awareness based on first order logic*', Personal Ubiquitous Comput. 7, 6, pp. 353-364.

Rolim, C. O., Kochy, F.L., Black, J. and Geyer, C. F. R. (2011) '*Health Solutions Using Low Cost Mobile Phones and Smart Spaces for the Continuous Monitoring and Remote Diagnostics of Chronic Diseases*', In Proceedings of eTELEMED 2011, The Third International Conference on eHealth, Telemedicine, and Social Medicine, Guadeloupe, France.

Romero L.M.R., Tosina L. J. R., Valderrama M.A.E., Arbizu J.C. and Martíne I.R. (2011) '*A Comprehensive View of the Technologies Involved in Pervasive Care*', Communications *in* Medical *and* Care Compunetics, pp. 3-19.

Roy, P., Abdulrazak, B., Belala, Y. (2011) '*A Distributed Architecture for Micro Context-aware Agents'*, Procedia Computer Science, 5, pp. 296-303.

Saha, D. and Mukherjee, A. (2003) *'Pervasive computing: A paradigm for the 21st century*', IEEE Computer, *36*(3), pp. 25-31.

Sang, P., So, W., Jong, L., Sung, K. (2003) '*Smart home – digitally engineered domestic life*', Personal and Ubiquitous Computing. pp. 189-196.

Satyanarayanan, M. (2011) '*Mobile computing: the next decade*', ACM SIGMOBILE Mobile Computing Communications Review, pp. 2-10.

Satyanarayanan, M. (2001) '*Pervasive Computing Vision andChallenges',* IEEE Personal Comm., vol. 6, no. 8, Aug. 2001, pp. 10–1.

Salber, D., Dey, A., Abowd, G. (1999) '*The Context Toolkit: Aiding the Development of Context-enabled Applications*', Proceedings of CHI'99, pp. 434-441.

Schulz, S. and Roth-Berghofer, Th (2005) *' Context-based Retrieval for Explainable Reasoning Applying Context Management' Revue d'intelligence artificelle RIA, Vol. 19 issue 3.*

Schilit, B.N., Adams, N., Want, R. (1994) '*Context-Aware Computing Applications*', Proc. of the Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA, IEEE Computer Society, pp. 85-90.

Schilit, B. N., Adams, N., Gold, R., Tso, M. M. and Want, R. (1993) '*The PARCTab Mobile Computing System*', Technical Report CSL-93-20, Xerox Palo alto Research Center.

Schmidt, A. (2000) *'Implicit Human-Computer Interaction through Context*', PersonalTechnologies, 4 (2 & 3), pp. 191-199.

Shojanoori, R. Juric, R., Lohi, M (2010) '*Reasoning and Decision Making in Managing Pervasive Computational Spaces'* in proceedings of the 17th Automated Reasoning Workshop, University of Westminster, March 30-31, available at http://www2.wmin.ac.uk/bolotoa/ARW/arw-2010.html

Shojanoori, R., Juric, R. and Lohi, M. (2012) *'Computationally Significant Semantics in Pervasive Healthcare'*, Transactions of the SDPS: Journal of Integrated Design and Process Science, 16 (1), 43-62.

Shojanoori, R. and Juric, R. (2013) *'Semantic Remote Patient Monitoring System'*, Telemedicine and e-Health, 19 (2), 1-8.

Shojanoori, R., Juric, R., and Tourani, B. (2010) '*Experiences of building assisted self care systems within smart home environment'*. In: Proceedings of the 15th International Conference on System Design and Process Science, 06 – 11, Dallas, USA.

Singh, S., Puradkar, S., Lee, Y. (2006) '*Ubiquitous computing: connecting Pervasive computing through Semantic Web*', Int. Journal on Information Systems and E-Business Management, Springer, 4, N4, pp. 421-439.

Spackman, K. (2000) *'SNOMED RT and SNOMED CT*', Promise of an international clinical ontology, M. D. Computing 17.

Stevenson, G., Knox, S., Dobson, S. and Nixon, P. (2009) ' *Ontonym: a collection of upper ontologies for developing pervasive systems*', In Proceedings of the 1st Workshop on Context, information and ontologies (Heraklion, Greece, J. M. Gomez-Perez, P. Haase, M. Tilly, and P. Warren, Eds. CIAO '09. ACM, New York, NY, 1-8.

Strang, T. and Linnhoff-Popien, C. (2004) *'context modelling survey*', In 1st Int. Workshop on Advanced Context Modelling, Reasoning and Management.
Comm, 6, No. 8, pp. 10–15.

Streitz, N. and Nixon, P. (2005) '*The Disappearing Computer*', ACM Communications, 48, pp. 33-35.

Thoyib, W., Lee, E.S., Park, M.G. (2011) '*Ubiquitous Healthcare system: A design on the remote monitoring based on walking activities*', Electrical Engineering and Informatics (ICEEI), 2011 International Conference. pp. 1-6.

Truong, H.L., Dustdar, S. (2010) '*Cloud Computing for Small Research Groups in Computational Science and Engineering: Current Status and Outlook'*, Computing Archives for Scientific Computing, Springer-Verlag.

Tsai, C. C., Lin, S. S. J. and Tsai, M. J. (2001) '*Developing an Internet attitude scale for high school students*', Computers & Education, 37(1), pp. 41-51.

W3C, *XML Technology*, Available at: http://www.w3.org/standards/xml/, (accessed 8 March 2013).

W3C (2012a) '*OWL 2 Web Ontology Language Quick Reference Guide (Second Edition)*', Available at: http://www.w3.org/TR/owl2-quick-reference/ (accessed 8 March 2013).

W3C (2012b) '*OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition)*', Available at: http://www.w3.org/TR/owl2-syntax/ (accessed 8 March 2013).

W3C (2011), *SWEO community Project* , Available at: http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData

W3C (2009) '*OWL 2 Web Ontology Language Document Overview (Second Edition)*', Available at: http://www.w3.org/TR/owl2-overview/ (accessed 8 March 2013).

W3C (2004a) '*OWL Web Ontology Language Overview*', Available at: http://www.w3.org/TR/owl-features/, (accessed 8 March 2013).

W3C (2004b) '*Semantic Web Stack*', Available at: http://www.w3.org/2004/Talks/1117-sb-gartnerWS/slidC18-0.html, (accessed 8 March 2013).

W3C (2004c) '*The Resource Description Framework*', Available at: www.w3.org/RDF (accessed 8 March 2013).

W3C (2004d) '*RDF Vocabulary Description Language 1.0: RDF Schema*', Available at: http://www.w3.org/TR/rdf-schema/ (accessed 8 March 2013).

W3C (2004e) '*OWL Web Ontology Language Reference*', Available at: http://www.w3.org/TR/owl-ref/, (accessed 8 March 2013).

Walker, G. H., Stanton, N. A., Young, M. S. (2001) '*Where is Computing Driving Cars*', *Int. J Human-Computer Interaction*, 13(2), pp. 203-229.

Wang, X.H., Zhang, D.Q., Gu, T., Pung, H.K. (2004) '*Ontology based context modelling and reasoning using OWL*', Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on, pp. 18-22.

Want, R., Hopper, A., Falcão, V., and Gibbons, J. (1992) '*The active badge location system*. ACM Trans. Inf. Syst. 10, 1, pp. 91-102.

Weiser, M., Gold, R., Brown, J.S. (1999) '*The origins of ubiquitous computing research at PARC in the late 1980s*', IBM Systems Journal, 38, No. 4.

Weiser, M. (1991) '*The computer of the 21st century*', Scientific American, 265 (3), pp. 66-75.

Weiser, M. and Brown, J.S. (1996) '*Designing Calm Technology*', PowerGrid Journal, 1:1.

Weiser, M. and Brown, J. S. (1998) '*The coming age of calm technology*', P. J. Denning and R. M. Metcalfe (Eds.), *Beyond calculation: The next fifty years of computing*, New York, pp. 75-85.

Winograd, T. (2001) '*Architectures for Context Uuman Computer interaction*', 16(2), pp. 401-419, Available at: http://hci.stanford.edu/winograd/papers/context/context.pdf.

Xiaosheng, T., Qinghua, S., Ping, Z. (2006) '*A distributed context aware model for pervasive service environment*', Wireless Pervasive Computing, 2006 1st International Symposium*,* pp. 16-18.

Yoo, Y. (2010) '*Computing in Everyday Life: A Call for Research on Experiential Computing*', MIS Quarterly, 34, No. 2, pp. 213-231.

Zhang Y., Jia Z. and Chen Y. (2011a) '*A thought on the goals & realization of pervasive healthcare*', Scientific Research & Essays, Vol. 6 (13), pp. 2752-2756.

Zhang, Y., Huang, G.Q., Qu, T., Ho, O., Sun, S. (2011b) '*Agent-based smart objects management system for real-time ubiquitous manufacturing, Robotics and Computer-Integrated Manufacturing*', Vol. 27, Issue 3, pp. 538-549.

Appendix A

In this appendix some screen shots of the implementation of the FCM through the example scenario in section 5.1.3 are presented.

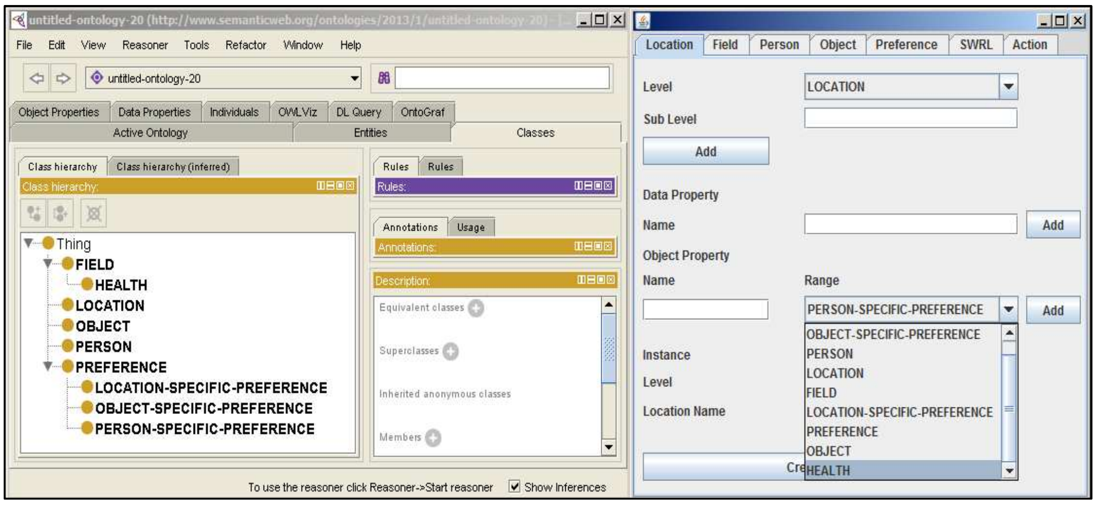
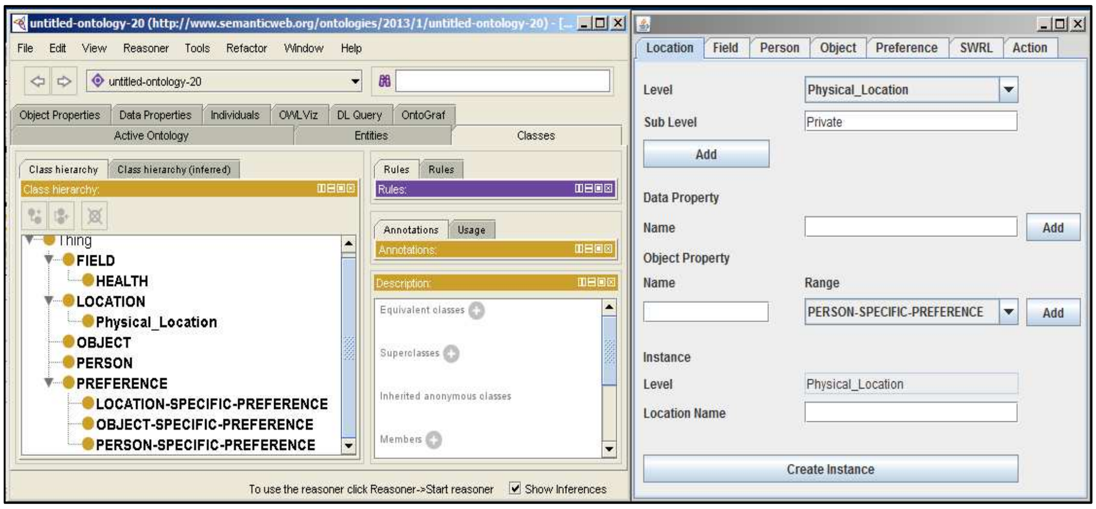

untitled-ontology-20 (http://www.semanticweb.org/ontologies/2013/1/untitled-ontology-20)
File Edit View Reasoner Tools Refactor Window Help
untitled-ontology-20
Object Properties Data Properties Individuals OWLViz DL Query OntoGraf
Active Ontology Entities Classes
Class hierarchy Class hierarchy (inferred)
Thing
FIELD
HEALTH
LOCATION
OBJECT
PERSON
PREFERENCE
LOCATION-SPECIFIC-PREFERENCE
OBJECT-SPECIFIC-PREFERENCE
PERSON-SPECIFIC-PREFERENCE
Rules
Annotations Usage
Description:
Equivalent classes
Superclasses
Inherited anonymous classes
Members
To use the reasoner click Reasoner->Start reasoner
Show Inferences
Location Field Person Object Preference SWRL Action
Level
LOCATION
Sub Level
Add
Data Property
Name
Object Property
Range
PERSON-SPECIFIC-PREFERENCE
OBJECT-SPECIFIC-PREFERENCE
PERSON
LOCATION
FIELD
LOCATION-SPECIFIC-PREFERENCE
PREFERENCE
OBJECT
HEALTH
Instance
Location Name

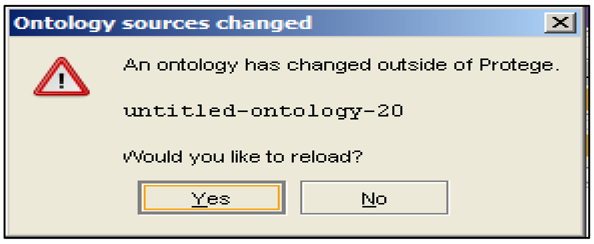
Figure A.1: Classes of GOnto and their display in a drop-down menu of the the Java application


untitled-ontology-20 (http://www.semanticweb.org/ontologies/2013/1/untitled-ontology-20)
File Edit View Reasoner Tools Refactor Window Help
Object Properties Data Properties Individuals OWLViz DL Query OntoGraf
Active Ontology Entities Classes
Class hierarchy Class hierarchy (inferred)
Thing
FIELD
HEALTH
LOCATION
Physical_Location
OBJECT
PERSON
PREFERENCE
LOCATION-SPECIFIC-PREFERENCE
OBJECT-SPECIFIC-PREFERENCE
PERSON-SPECIFIC-PREFERENCE
Rules
Annotations Usage
Description:
Equivalent classes
Superclasses
Inherited anonymous classes
Members
To use the reasoner click Reasoner->Start reasoner
Show Inferences
Location Field Person Object Preference SWRL Action
Add
Data Property
Name
Object Property
Range
Instance
Location Name
Create Instance

Level: Physical_Location
Sub Level: Private
Range: PERSON-SPECIFIC-PREFERENCE
Instance Level: Physical_Location

Figure A.2: Extending SeCHOnto by adding “Private” as a subclass of “Physical-Location”

Each time the ontology changes the dialog box of Figure A.3 appears as the ontology was changed through the application and not locally.



Ontology sources changed

An ontology has changed outside of Protege.

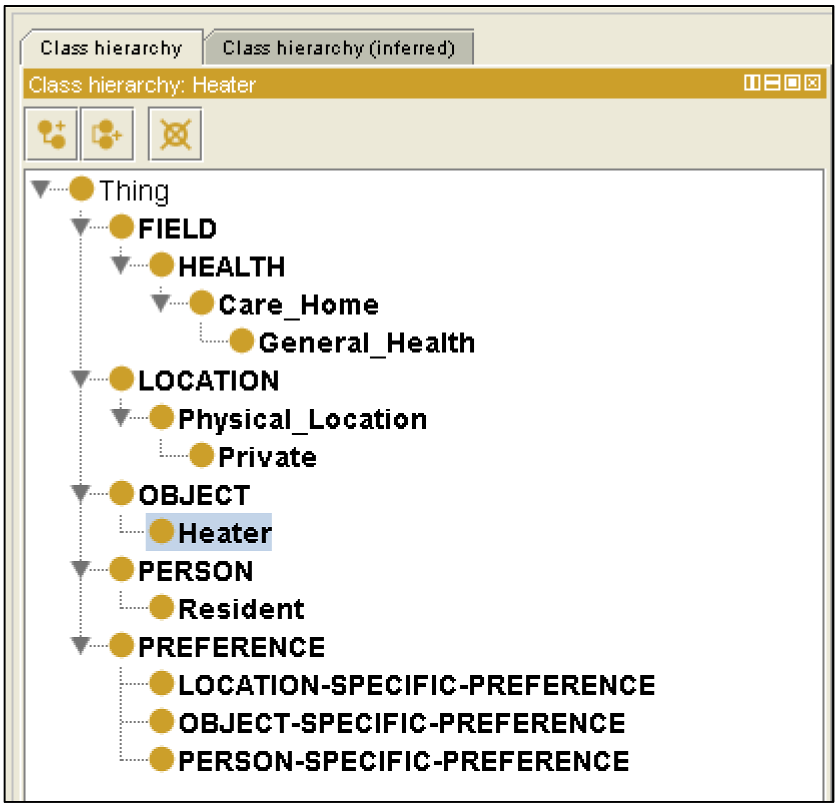
untitled-ontology-20

Would you like to reload?

Yes No

Figure A.3: Reloading of changed ontology dialog box

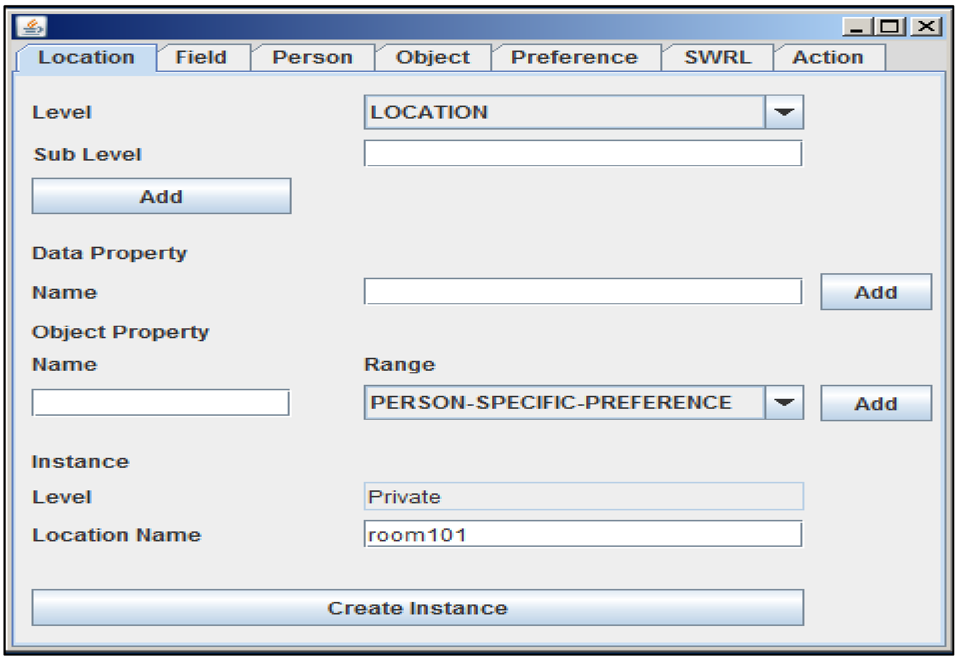
Once all new classes have been added, the SeCHOnto structure is as shown in Figure A:4.


Class hierarchy | Class hierarchy (inferred)

Class hierarchy: Heater

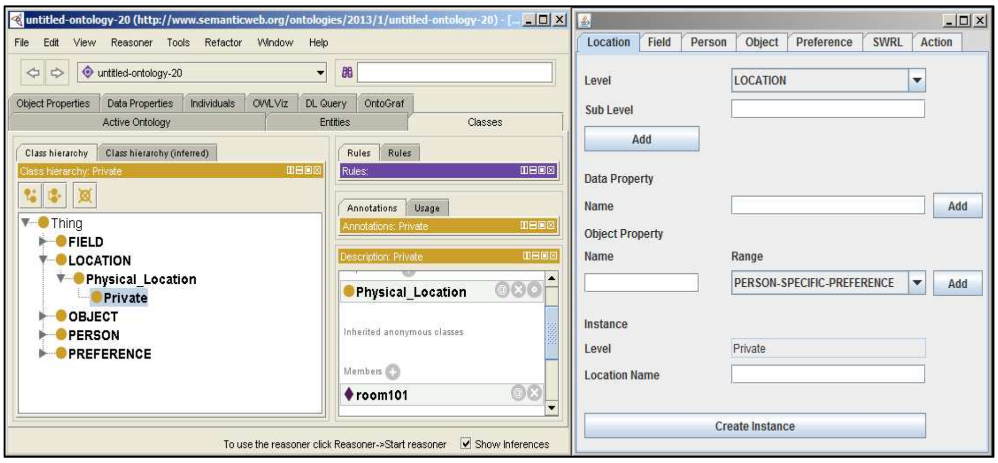
- Thing
  - FIELD
    - HEALTH
      - Care_Home
        - General_Health
  - LOCATION
    - Physical_Location
      - Private
  - OBJECT
    - Heater
  - PERSON
    - Resident
  - PREFERENCE
    - LOCATION-SPECIFIC-PREFERENCE
    - OBJECT-SPECIFIC-PREFERENCE
    - PERSON-SPECIFIC-PREFERENCE

Figure A.4: SeCHOnto when all new classes have been added to the structure



Level	LOCATION	
Sub Level		
Add		
Data Property		
Name		Add
Object Property		
Name	Range	
	PERSON-SPECIFIC-PREFERENCE	Add
Instance		
Level	Private	
Location Name	room101	
Create Instance		

Figure A.5: Asserting an individual "room101" as a member of Private class

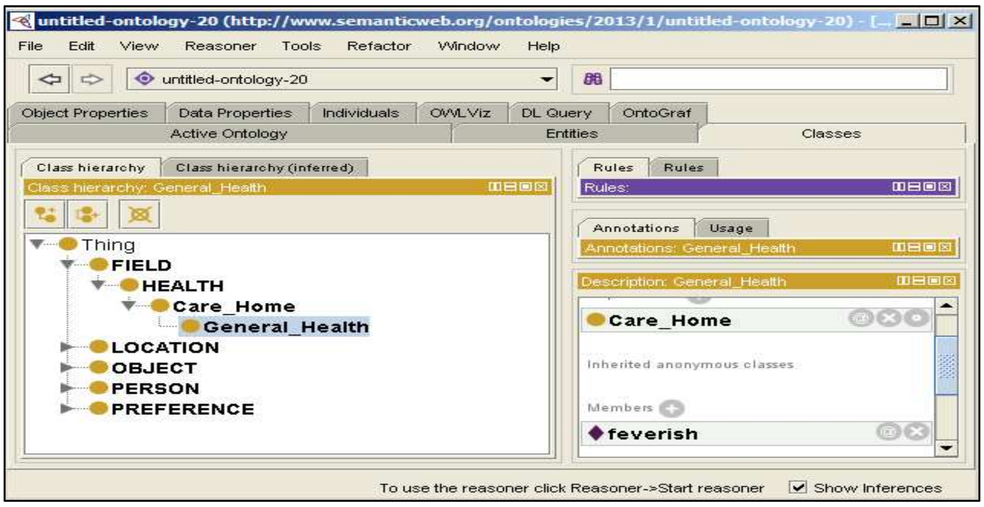

untitled-ontology-20 (http://www.semanticweb.org/ontologies/2013/1/untitled-ontology-20)
File Edit View Reasoner Tools Refactor Window Help
Object Properties Data Properties Individuals OWLViz DL Query OntoGraf
Active Ontology Entities Classes
Class hierarchy Class hierarchy (inferred)
Rules Annotations Usage
To use the reasoner click Reasoner->Start reasoner Show Inferences
Location Field Person Object Preference SWRL Action
Add
Create Instance

Class hierarchy: Private
- Thing
  - FIELD
  - LOCATION
    - Physical_Location
      - Private
  - OBJECT
  - PERSON
  - PREFERENCE

Description: Private
Physical_Location
Inherited anonymous classes
Members
room101

Level	LOCATION
Sub Level	
Data Property Name	
Object Property Name	
Range	PERSON-SPECIFIC-PREFERENCE
Instance Level	Private
Location Name	

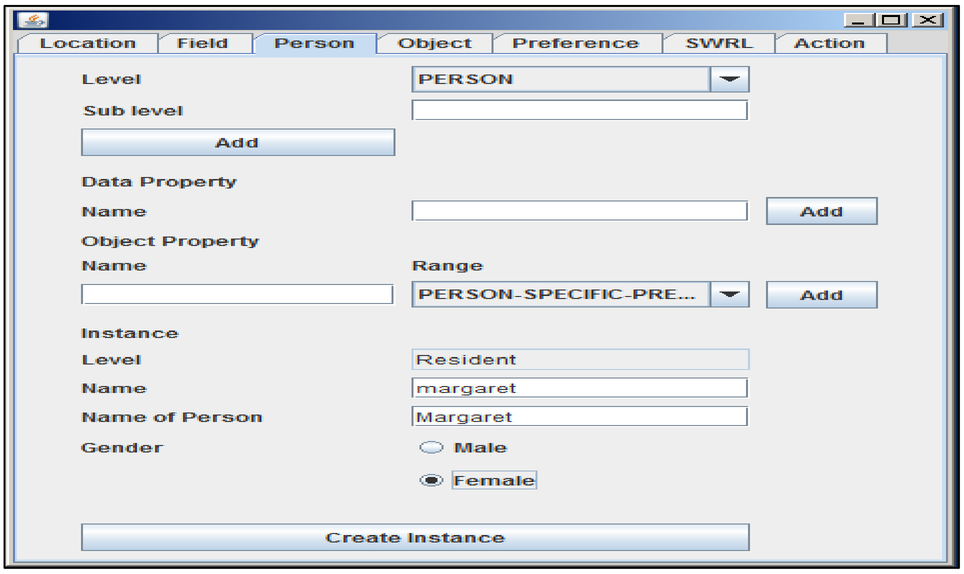
Figure A.6: Asserting an individual “room101” is reflected in SeCHOnto ontology


untitled-ontology-20 (http://www.semanticweb.org/ontologies/2013/1/untitled-ontology-20)
File Edit View Reasoner Tools Refactor Window Help
Object Properties Data Properties Individuals OWLViz DL Query OntoGraf
Active Ontology Entities Classes
Class hierarchy Class hierarchy (inferred)
Rules Annotations Usage
To use the reasoner click Reasoner->Start reasoner Show Inferences

Class hierarchy: General_Health
- Thing
  - FIELD
    - HEALTH
      - Care_Home
        - General_Health
  - LOCATION
  - OBJECT
  - PERSON
  - PREFERENCE

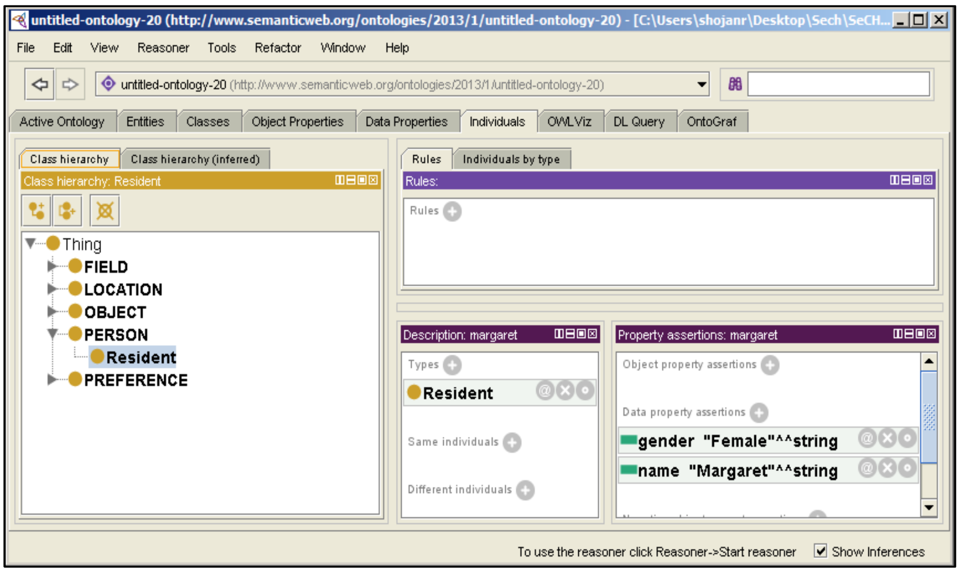
Description: General_Health
Care_Home
Inherited anonymous classes
Members
feverish

Figure A.7: Asserting of feverish as an individual of Care_Home



Level	PERSON
Sub level	
Data Property Name	
Object Property Name	
Range	PERSON-SPECIFIC-PRE...
Instance Level	Resident
Name	margaret
Name of Person	Margaret
Gender	Male [✓ Female]

Figure A.8: Asserting of an individual of Resident along with generic data type properties



untitled-ontology-20 (http://www.semanticweb.org/ontologies/2013/1/untitled-ontology-20) - [C:\Users\shojanr\Desktop\Sech\SeCH...

untitled-ontology-20 (http://www.semanticweb.org/ontologies/2013/1/untitled-ontology-20)

Class hierarchy: Resident

- Thing
  - FIELD
  - LOCATION
  - OBJECT
  - PERSON
    - Resident
  - PREFERENCE

Rules:

Rules

Description: margaret

Types

Resident

Same individuals

Different individuals

Property assertions: margaret

Object property assertions

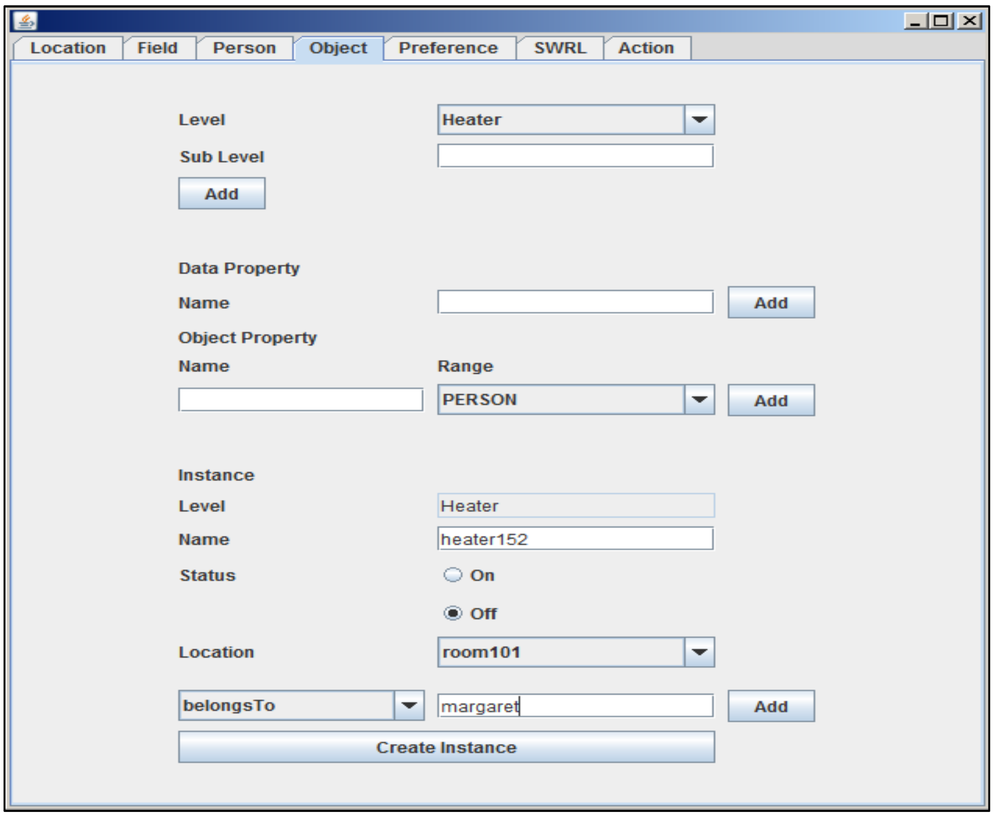
Data property assertions

gender "Female"^^string

name "Margaret"^^string

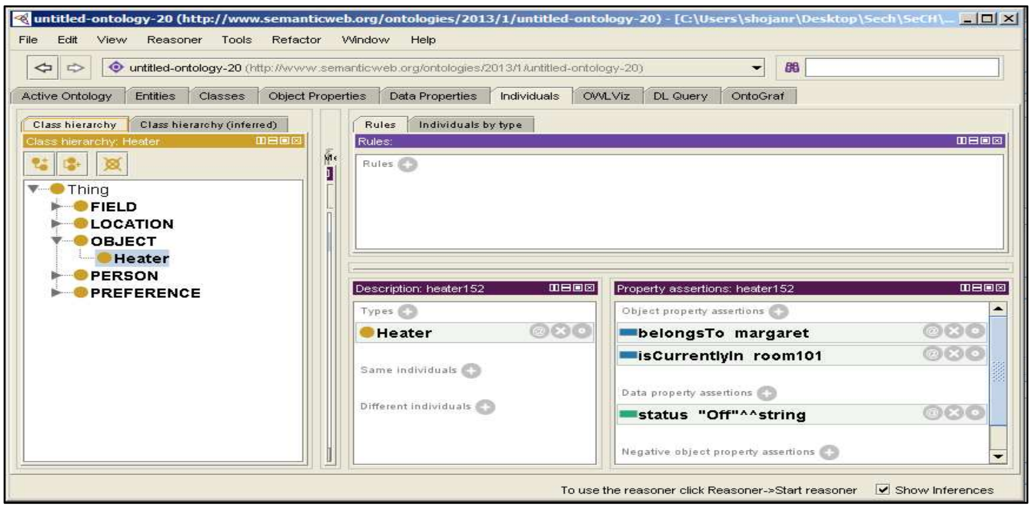
To use the reasoner click Reasoner->Start reasoner ☑ Show Inferences

Figure A.9: Reloading the ontology after action of Figure A.8



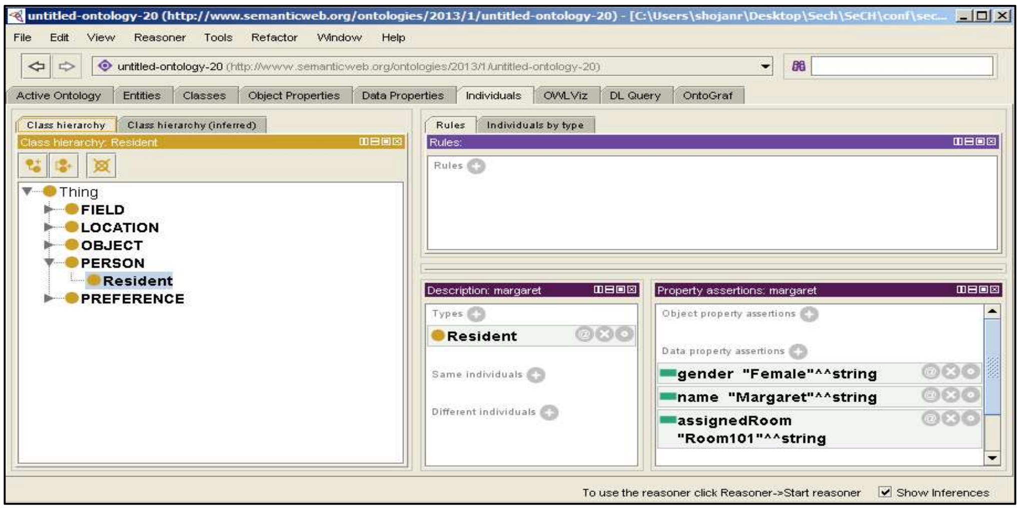
Level	Heater
Sub Level	
Data Property	
Name	
Object Property	
Name	Range
	PERSON
Instance	
Level	Heater
Name	heater152
Status	On [✓ Off]
Location	room101
belongsTo	margaret

Figure A.10: Display of extending SeCHOnto with an object property


untitled-ontology-20 (http://www.semanticweb.org/ontologies/2013/1/untitled-ontology-20) - [C:\Users\shojanr\Desktop\Sech\SeCH\...
File Edit View Reasoner Tools Refactor Window Help
Active Ontology Entities Classes Object Properties Data Properties Individuals OWLViz DL Query OntoGraf
Class hierarchy Class hierarchy (inferred)
Class hierarchy: Heater
Thing
FIELD
LOCATION
OBJECT
Heater
PERSON
PREFERENCE
Rules Individuals by type
Description: heater152
Types
Heater
Same individuals
Different individuals
Property assertions: heater152
Object property assertions
Data property assertions
Negative object property assertions
To use the reasoner click Reasoner->Start reasoner
Show Inferences

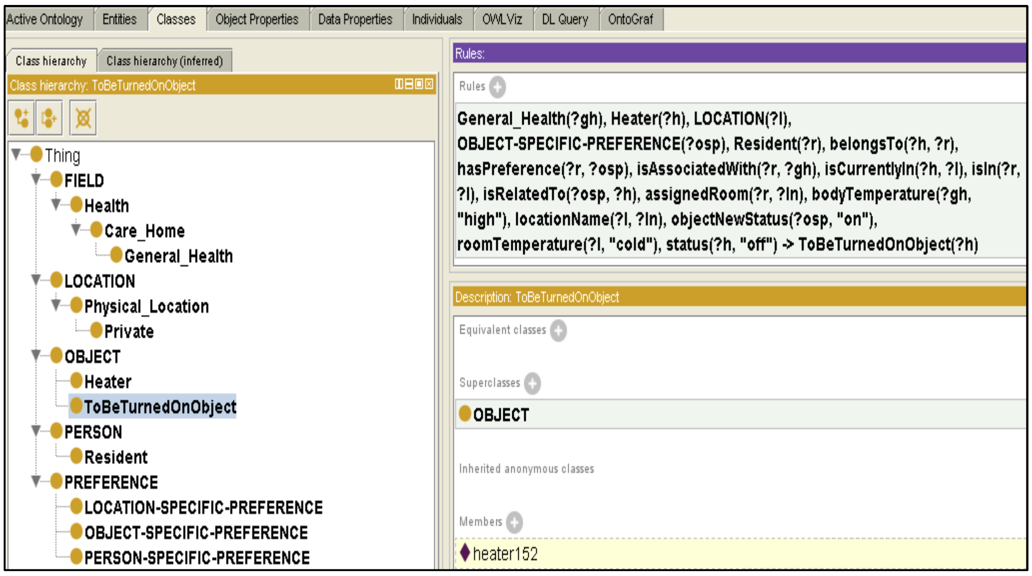
belongsTo margaret
isCurrentlyIn room101
status "Off"^^string

Figure A.11: Display of updated SeCHOnto after extension of figure A.10


untitled-ontology-20 (http://www.semanticweb.org/ontologies/2013/1/untitled-ontology-20) - [C:\Users\shojanr\Desktop\Sech\SeCH\conf\sec...
File Edit View Reasoner Tools Refactor Window Help
Active Ontology Entities Classes Object Properties Data Properties Individuals OWLViz DL Query OntoGraf
Class hierarchy Class hierarchy (inferred)
Class hierarchy: Resident
Thing
FIELD
LOCATION
OBJECT
PERSON
Resident
PREFERENCE
Rules Individuals by type
Description: margaret
Types
Resident
Same individuals
Different individuals
Property assertions: margaret
Object property assertions
Data property assertions
To use the reasoner click Reasoner->Start reasoner
Show Inferences

gender "Female"^^string
name "Margaret"^^string
assignedRoom "Room101"^^string

Figure A.12: Addition of situation-specific data type property “assignedRoom” to “margaret”


Active Ontology Entities Classes Object Properties Data Properties Individuals OWLViz DL Query OntoGraf
Class hierarchy Class hierarchy (inferred)
Class hierarchy: ToBeTurnedOnObject
Thing
FIELD
Health
Care_Home
General_Health
LOCATION
Physical_Location
Private
OBJECT
Heater
ToBeTurnedOnObject
PERSON
Resident
PREFERENCE
LOCATION-SPECIFIC-PREFERENCE
OBJECT-SPECIFIC-PREFERENCE
PERSON-SPECIFIC-PREFERENCE
Rules:
Rules
Description: ToBeTurnedOnObject
Equivalent classes
Superclasses
OBJECT
Inherited anonymous classes
Members
heater152

General_Health(?gh), Heater(?h), LOCATION(?l), OBJECT-SPECIFIC-PREFERENCE(?osp), Resident(?r), belongsTo(?h, ?r), hasPreference(?r, ?osp), isAssociatedWith(?r, ?gh), isCurrentlyIn(?h, ?l), isIn(?r, ?l), isRelatedTo(?osp, ?h), assignedRoom(?r, ?ln), bodyTemperature(?gh, "high"), locationName(?l, ?ln), objectNewStatus(?osp, "on"), roomTemperature(?l, "cold"), status(?h, "off") -> ToBeTurnedOnObject(?h)

Figure A.13: The result of running the SWRL rule for the running example CQ

Appendis B

The software application.