Exploiting Semantic Web Technologies in the Management of a Smart Classroom

Maria Koutraki

Thesis submitted in partial fulfillment of the requirements for the

Masters' of Science degree in Computer Science

University of Crete
School of Sciences and Engineering
Computer Science Department
Knossou Av., P.O. Box 2208, Heraklion, GR-71409, Greece

Thesis Advisors: Prof. *Dimitris Plexousakis*

This work has been performed at the **University of Crete, School of Science and Engineering, Computer Science Department** and at the **Institute of Computer Science of Foundation for Research and Technology-Hellas (FORTH).**

This work has been supported by the **Institute of Computer Science of Foundation for Research and Technology-Hellas (FORTH).**

University of Crete
Computer Science Department

Exploiting Semantic Web Technologies in the Management of a Smart Classroom

Thesis submitted by
Maria Koutraki
in partial fulfillment of the requirements for the
Masters' of Science degree in Computer Science

THESIS APPROVAL

Author: ______________________
Maria Koutraki

Committee approvals: ______________________
Dimitris Plexousakis
Professor, Supervisor

____
Vassilis Christophides
Professor, Committee Member

____
Constantine Stephanidis
Professor, Committee Member

Departmental approval: ______________________
Angelos Bilas
Professor, Director of Graduate Studies

Heraklion, September 2012

Abstract

Interactive systems have been among the prevailing computing paradigms of recent years. After a lot of research in this field, *implicit* human computer interaction is growing. It takes the user context into account when creating new applications for ambient intelligence. Context and context-awareness are central issues to ambient intelligence. The use of context information in interactive systems offers new possibilities to adapt applications and systems to the current situation "on-the-fly".

In this thesis, we present the management system of a university smart classroom based on semantic web technologies. This work constitutes a real-time, context-aware system, applied in a smart classroom domain, which aims to assist its users after recognizing occurring activities. We have designed and developed several ontologies that capture and formally describe user profiles, context information and learning material information. Rule-based reasoning is applied for the elaboration of the stored knowledge and an existing recognition system undertakes to identify the current activity by exploiting the reasoning result. After the activity recognition, rules are responsible for the assistance of the activity.

We describe an overview of our system as well as typical usage scenarios to indicate how our system would react in these situations. Depending on the current activity the system will provide different type of assistance to its users. Assistance consists of modifying the status of the devices that are located in the classroom or supplying the students with learning material. A live demonstration of our system with real users into a smart space is described in the end of this thesis.

Supervisor: Dimitris Plexousakis
Professor

Περίληψη

Τα αλληλεπιδραστικά συστήματα βρίσκονται μεταξύ των κύριων υπολογιστικών προτύπων τα τελευταία χρόνια. Μετά από πολλή έρευνα σε αυτόν τον τομέα, έχει αναπτυχθεί η έμμεση αλληλεπίδραση ανθρώπου-υπολογιστή. Κατά τη δημιουργία μιας εφαρμογής διάχυτης νοημοσύνης πέραν άλλων, λαμβάνεται υπόψη και το περιβάλλον του χρήστη. Το περιβάλλον και τα δεδομένα του περιβάλλοντος είναι κεντρικά ζητήματα στη διάχυτη νοημοσύνη. Η χρήση των δεδομένων του περιβάλλοντος στα αλληλεπιδραστικά συστήματα προσφέρει νέες δυνατότητες στην προσαρμογή των εφαρμογών και των συστημάτων άμεσα ("$on-the-fly$") στην τρέχουσα κατάσταση.

Σε αυτή την εργασία, παρουσιάζουμε το σύστημα διαχείρισης μιας έξυπνης τάξης πανεπιστημίου βασιζόμενο σε τεχνολογίες του σημασιολογικού ιστού. Η δουλειά αυτή είναι ένα πραγματικού χρόνου σύστημα που λαμβάνει υπόψη του τις παραμέτρους του περιβάλλοντος και εφαρμόζεται σε μία έξυπνη τάξη. Έχει ως στόχο να υποβοηθήσει τους χρήστες του μετά την αναγνώριση κάθε δραστηριότητας που συμβαίνει μέσα στον έξυπνο χώρο (έξυπνη τάξη). Έχουμε σχεδιάσει και αναπτύξει διάφορες οντολογίες που αποτυπώνουν και περιγράφουν τυπικά τα προφίλ των χρηστών, πληροφορίες περιβάλλοντος και μαθησιακό υλικό. Για την επεξεργασία της αποθηκευμένης γνώσης γίνεται χρήση συλλογιστικής βασισμένη σε κανόνες και ένα υπάρχον σύστημα αναγνώρισης αναλαμβάνει να εντοπίσει την τρέχουσα δραστηριότητα χρησιμοποιώντας το αποτέλεσμα του συλλογισμού. Μετά την αναγνώριση της εκάστοτε δραστηριότητας, κανόνες είναι υπεύθυνοι για τη υποβοήθησή της.

Γίνεται μια επισκόπηση του συστήματός μας και τυπικά σενάρια χρήσης παρατίθενται με σκοπό να δείξουμε πώς το σύστημα θα αντιδράσει στις συγκεκριμένες καταστάσεις. Ανάλογα με την τρέχουσα ενέργεια παρέχεται διαφορετικού είδους υποβοήθηση στους χρήστες του συστήματος. Η υποβοήθηση αφορά την αλλαγή στην κατάσταση των συσκευών που υπάρχουν μέσα στην έξυπνη τάξη διδασκαλίας ή την παροχή διδακτικού υλικού στους φοιτητές. Μια ζωντανή επίδειξη του συστήματος, με πραγματικούς χρήστες σε έναν έξυπνο χώρο περιγράφεται στο τέλος αυτής της εργασίας.

Επόπτης Καθηγητής: Δημήτρης Πλεξουσάκης
Καθηγητής

Ευχαριστίες

Καταρχήν θέλω να εκφράσω τις ευχαριστίες μου στον επόπτη μου, καθηγητή Παν/μίου Κρήτης κ. Δημήτρη Πλεξουσάκη που με καθοδήγησε δίνοντάς μου πολύτιμες συμβουλές ώστε να ολοκληρώσω με επιτυχία την μεταπτυχιακή μου εργασία. Όπως επίσης και στον καθηγητή μου κ. Γρηγόρη Αντωνίου ο οποίος με εμπιστεύτηκε και με ενθάρρυνε να συνεχίσω τις σπουδές μου προτείνοντας μου αυτή την εργασία. Επίσης ευχαριστώ τον καθηγητή κ. Βασίλη Χριστοφίδη και τον καθηγητή κ. Κωνσταντίνο Στεφανίδη για τις εύστοχες παρατηρήσεις και συμβουλές τους ως μέλη της εισηγητικής επιτροπής.

Αυτή η δουλεία δεν θα υπήρχε χωρίς τις ιδέες των απόφοιτων διδακτόρων του τμήματος Θοδωρή Πάτκου και Αντώνη Μπικάκη που μπορεί η συνεργασία μας να διήρκησε μικρό χρονικό διάστημα αλλά η συμβολή και το ενδιαφέρον τους ήταν καθοριστικά για την μετέπειτα πορεία της εργασίας.

Όλα αυτά τα χρόνια είχα την τύχη να συνεργαστώ με πολλά αξιόλογα άτομα που πρόσφερα πάντα τις γνώσεις και τη βοήθειά τους όταν ήταν απαραίτητο. Ευχαριστώ γι' αυτό τον Γιάννη Χρυσάκη και τον Νίκο Φτυλιτάκη και ιδιαιτέρα τον Βασίλη Ευθυμίου που μαζί με την Δήμητρα Ζωγραφιστού μοιραζόμασταν τις ίδιες αγωνίες, άγχη αλλά και χαρές που ακολουθούσαν τις εργασίες μας.

Σ' αυτό το σημείο θα ήθελα να ευχαριστήσω θερμά όλους τους φίλους μου αλλά και τους συμφοιτητές μου στο Ηράκλειο και στο Παρίσι για τα πολύ ωραία χρόνια που περάσαμε. Ιδιαιτέρες ευχαριστίες αρμόζουν στις φίλες μου Νέλη, Σοφία, Δανάη και Δήμητρα για όλες εκείνες τις δύσκολες στιγμές που ήταν εκεί για μένα και με ενθάρρυναν να συνεχίσω ακούγοντας με υπομονή όλους τους φόβους και τις ανησυχίες μου. Ένα μεγάλο ευχαριστώ επίσης στον Γιώργο που δεν με άφησε να τα παρατήσω και πίστευε σε μένα περισσότερο απ' όσο εγώ στον εαυτό μου. Όπως επίσης και στις ξαδέλφες μου Ηρώ και Ράνια για την αγάπη και την συμπαράσταση τους.

Τέλος, θα ήθελα να ευχαριστήσω θερμά την οικογένεια μου και ιδιαιτέρα τους γονείς μου Ορέστη και Μαρία όπως και το μικρότερο αδελφό μου Νίκο για την συνεχή υποστήριξη την αμέριστη συμπαράσταση και την ενθάρρυνση που μου προσφέρουν από τα πρώτα χρόνια της ζωής μου μέχρι και σήμερα. Ήταν πάντα δίπλα μου σε όλες τις επιλογές μου δίνοντας μου την αγάπη τους. Χωρίς εσάς τίποτα δεν θα ήταν ίδιο!

Σας ευχαριστώ!

Μαρία Κουτράκη,
Σεπτέμβριος 2012

Στους γονείς μου,
που μου δίδαξαν τι είναι σημαντικό στη ζωή.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

During the centuries, human beings continuously try to adapt themselves to the environment in order to survive. Nowadays, Ambient Intelligence (AmI) technologies can change this status through the development of "intelligent" environments with main purpose to adapt to human needs and generally improve living conditions. This is our vision of 'Ambient Intelligence': people living easily in digital environments in which the electronics are sensitive to people's needs, personalized to their requirements, anticipatory of their behaviour and responsive to their presence. The intelligent environments are divided based on the different needs of their users. Some of the most usual smart environments are the smart home, the smart hospital, the smart office or the smart classroom.

In this work we focus on the intelligent environment of a university classroom. The decision to deal with this specific environment was taken after we realize how much useful time is spending on technical issues, such as lighting, projector set-up, photocopy distribution and other simple but time consuming actions that take place in a classroom. This time could be replaced with "teaching time", if all these issues were solved automatically.

Depending on the time, a "smart classroom" is used to refer to different things. In each period a smart classroom is a classroom that contains latest technologies. In the past, a smart classroom could have contained a computer with internet access, DVD player, VCR, document camera, and a projector or even before that, it may had been a simple overhead projector. Today, a "smart classroom" usually refers to a classroom that is equipped with an interactive whiteboard (Smart Board) and/or Smart computer software, other sensors about localizations searching and sensors responsible to control the environmental conditions.

This work extends our previous works [25] and [13] and the aim is to be able to use and combine all these existing technologies in order to create an intelligent environment that can recognize the needs of the users and improve the conditions into the smart space based on these needs. We present a system that assists instructors and students in a university smart classroom, in order to avoid spending time in such minor issues and stay focused on the teaching and learning process, by also having more studying material at their disposal. To accomplish this, we have taken advantage of the benefits that semantic web offers.

1.1 Goals - Contributions

Ambient Intelligence (AmI) could be also described as the branch of Artificial Intelligence in which the context of a so-called "smart" space can be modelled, processed and even altered in ways that satisfy certain needs. Systems that are able to adapt any possible context changes are called context-aware. In this thesis we are going to create a context-aware system that would be able to assist/help the users of an intelligent environment and more specific, the users of a smart University classroom after the identification of a specific activity that takes place into this environment. In order to accomplish that we use semantic web technologies and more specifically, ontologies for the context representation and semantic web rule language for the reasoning on the context data. Moreover, we use an existing activity recognition system that takes as input minor activities that happening in the classroom and identifying the main activity based on a given scenario.

The contributions of this work are focused on how we can combine several technologies in order to make "smarter" an environment that contains smart devices. We tried to fully represent the context of this specific smart environment through ontologies. We use two level of ontologies. An upper-level for the general features and the domain specific ontologies that are fully oriented to the smart classroom domain. An other contribution is the combination of the semantic web technologies, ontologies and semantic web rule language, with an activity recognition system based on machine learning. Moreover, the domain that we focused can maintain multiple users with different characteristics and needs. A contribution of this work is to assist all the users independently based on their personal characteristics and needs. An other very important point in the context-aware systems is that they are able to adapt to the human needs the right time. Our system is a real-time assistance system that is able to adapt in the new circumstances on time.

1.2 Motivation Scenarios

Scenarios in our work express the activities that can take place in a classroom e.g. lecture, exam. In order to identify these activities, a series of simpler events should be applied, in each case. After that, we undertake to assist the particular activity.

Lecture Presentation. A typical scenario in a classroom is a *lecture presentation*. In this scenario, a professor is giving a lecture presentation in the Smart Classroom. The audience consists of the students participating in the lecture.

Each person in the audience carries an electronic device (smart-phone/laptop). Through the ontologies we have access to the profile of the users, their personal calendars. Their position is calculated by using rfid tags and rfid readers. Each user has an rfid tag and when the tag connects with an rfid reader we can have the user's location. The calendar of the classroom is checked for lecture scheduled at this time and the audience members' personal calendars are checked for participation in this lecture, in order to know that everyone is in the right room. The lights and the projector are currently on. The professor stands near the display of presentation. Professor's profile matches to the "presenter's" profile in the calendar of the classroom. The students are seated and the noise level is low, indicating that there is a teaching activity in the classroom. Knowing these simple facts, "Lecture Presentation" activity is identified.

After the identification of the activity, the presentation is assisted by lowering the lights, closing the blinds and the windows and sending the presentations' file in students' devices. The microphone in the presenter's stand is activated and the audience microphones are deactivated. The audio output of the classroom is connected to the microphone. The audience has the chance to record the whole, or part of this presentation. In case a member of the audience has different language preferences and a translator is available, the audio output of this member's device is set to be the translator's output. The slides of the presentation is also provided to this member translated. Moreover, the profiles of the students that participate in the lecture are considered independently in order to have information about the learning level each user has, based on the theme of the lecture. Additional notes are sending to the students. Depending on the level of his knowledge in the specific lecture each user will receive different material. Moreover, a service responsible for Google search searches for basic term of the lecture and sends the results to the students. The lecture presentation file is also sent to the students that had to be in the lecture but they didn't appear. In addition, 10 minutes before the scheduled end of the lecture the professor is informed that the time ends in order to summarize.

Exceptional occasions may happen in this activity. One of them is to appear a problem in the projector or in one of the other devices and resources that are needed, e.g. in the translator. In order to avoid to interrupt or cancel the lecture because of technical problems, 20 minutes before the beginning of the lecture all the needed devices are checked from the system. If a problem appears then the system checks for alternatives. So first of all it checks for similar devices in the same room that can replace the problematic devices and after that sends message to the administrator of the classroom in order to fix the problem.

When the presentation is over and everyone has left, the classroom is set to its default state.

Conversation. During or independently of the *lecture presentation* event an other scenario/activity can occur, a *conversation* between professor and students. A student rises and makes a question. The lighting of the room is set to normal again(if there was in low level). This student's microphone is activated and in case his language preferences are different, his question is translated. The camera focus on him (if we have a video record). After the question from the student, professor starts talking in order to answer. Professors' microphone is activated and the camera focus on him.

Exam. An other typical activity that may happen in a classroom is an exam. In this scenarios the students are giving an exam on a course with the presence of their professor.

All the students are sitting on their desks. The calendar of the classroom is checked for exam scheduled at this time and the participants' personal calendars are checked for participation in this exam, in order to know that everyone is in the right room.

After the identification of the activity, the exam is assisted by adjusting the lighting in the classroom, closing the windows in order to decrease the noise level and deactivating the microphones.

Break. An other scenario/activity that can appear in a classroom is the *break*. This event happens in parallel to *Lecture Presentation* activity.

Some of the students are located in the corridor, the professor is not standing next to the display, he may be in his office or in the corridor. A lecture is scheduled in the classrooms calendar and it is not finished. Knowing these simple facts, "Break" activity is identified.

After the identification of the activity *break* the system adjusts the lighting in medium level and opens the windows of the classroom for getting fresh air.

Lecture using smart board. A typical scenario in a classroom is a *lecture presentation*. In this scenario, a professor is giving a lecture, by using (writing) a smart board, in the Smart Classroom. The audience consists of the students that participate in the lecture.

Each person in the audience carries an electronic device (smart-phone/laptop). Through the ontologies we have access to the profile of the users, their personal calendars, like the scenario with the presentation. Their position is calculated by using rfid tags and rfid readers. Each user has an rfid tag and when the tag connects with an rfid reader we can have the user's location. The calendar of the classroom is checked for lecture scheduled at this time and the audience members' personal calendars are checked for participation in this lecture, in order to know that everyone is in the right room. The professor stands near the smart board. Professor is writing on the smart board. Professor's profile matches to the "lecturer's" profile in the calendar of the classroom. The students are seated and the noise level is low, indicating that there is a teaching activity in the classroom. Knowing these simple facts, "Lecture using smart board" activity is identified.

After the activity's identification the lecture is assisted by adjusting the lighting level in the classroom (the lights' level is set to "hight"). The microphones of the audience are deactivated. During the lecture the system records and saves the notes from the smart board and sending them to the students. Moreover, it identifies the key points of the presentation by the words that professor underlines and makes a google search in order to find and send to the students the definitions of basic terms.

Exceptional occasions may also happen in this activity. One of them is the appearance of a problem in the smart board. In order to avoid to interrupt or cancel the lecture because of technical problems, 20 minutes before the beginning of the lecture all the devices needed are checked from the system. If a problem appears then the system checks for alternative devices or informs the administrator of the classroom.

1.3 Thesis Organization

The rest structure of this thesis organized as follows:

In Chapter 2 we present a brief explanation of notions and terms that will be used in the rest of this paper.

In Chapter 3 we present the work related to our system; Other AmI systems that focus on learning spaces and also other works that focus on context representation.

Chapter 4 presents the architecture of our work, since it analyses the basic components that compose our system.

Chapter 5 describes the way that we decide to represent the context of the smart classroom and also presents all the ontologies that we have developed for this purpose.

Chapter 6 describes the technique of the rule-based reasoning that we use in order to elaborate with the ontologies data and to draw conclusions about the kind of assistance that we will provide to the users.

Chapter 7 presents a real-time demonstration of our system with real users in a smart environment.

Chapter 8 ends this paper with useful conclusions and suggestions about possible future work. It is a short summary of the most important features of this work and it also presents some more possibilities of this system that could be exploited in the future.

Chapter 2

Background

Before presenting a more detailed view of the proposed real time management system of an intelligent classroom, some technologies that are used for the modeling and for the development of the system will be covered. In this chapter we introduce background information based on Ambient Intelligence Systems and Semantic Web, in order to provide a better understanding of this master's thesis.

2.1 Ambient Intelligence (AmI)

Ambient intelligence is the vision of a technology that will become invisibly embedded in our natural surroundings, present whenever we need it, enabled by simple and effortless interactions, attuned to all our senses, adaptive to users and context-sensitive, and autonomous. High-quality information access and personalized content must be available to everybody, anywhere, and at any time [41].

There are other techniques like *Networks Computing*, *Sensors*, *Human Centric Computer Interfaces (HCI)*, *Pervasive Ubiquitous Computing* and *Artificial Intelligence (AI)* that are all relevant and interrelated but none of them conceptually covers the full scope of AmI. Ambient Intelligence manage to combine all these resources in order to provide flexible and intelligent services fitting in the environment that the users act having as main concept the "disappearing computer" [4] [38]. According to Aarts, Harwig & Schuurmans [2] the technologies and the systems that are used to create an AmI system have to be:

- Embedded: The network devices have to integrated into the environment.
- Personalized: The system (devices) have to recognize users and their situational context.
- Adaptive: Devices can change their status in response to the user.
- Anticipatory: Devices can anticipate users' desires without conscious mediation.

There are five major principle that we based on to design an Ambient Intelligence system according to [4]:
Who: the identification of a `user` of the system and the role that user plays within the system in relation to other users or devices. This can be extended to identifying important elements like pets, robots and objects of interest within the environment.
Where: the tracking of the `location` where a user or an object is geographically located at each moment during the system operation. This can demand a mix of technologies, for example technology that may work well indoors may be useless outdoors and vice-versa.
When: the association of activities with `time` is required to build a realistic picture of a system's dynamic. For example, users living in a house will change location often change location and knowing when those changes happened and for how long they lasted are fundamental to the understanding of how an environment is evolving.
What: the recognition of `activities` and tasks users are performing is fundamental in order to provide appropriate help if required. The multiplicity of possible scenarios that can follow an action makes this very difficult. Spatial and temporal awareness help to achieve task awareness.
Why: the capability to infer and understand `intentions` and goals behind activities is one of the hardest challenges in the area but a fundamental one which allows the system to anticipate needs and serve users in a sensible way.

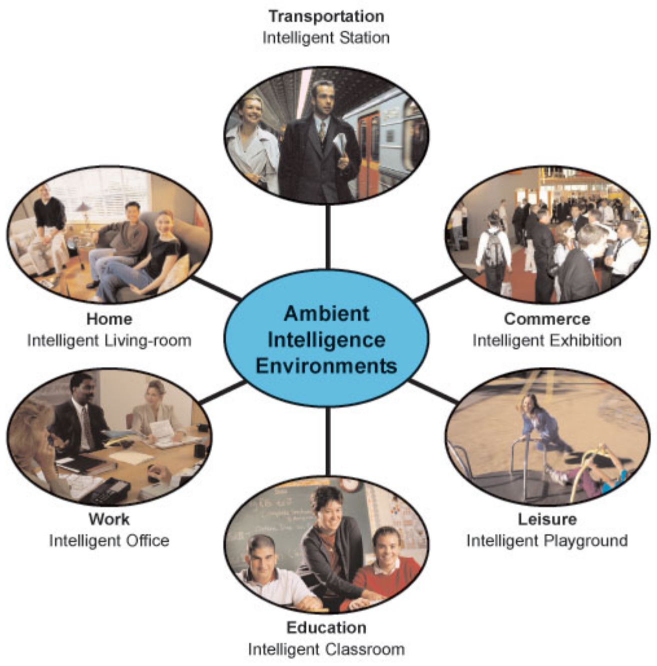


Figure 2.1: The key thematic areas in the context of AmI Programme.

Ambient Intelligence may have applications in many thematic areas as it mentioned in AmI Programme of Institute of Computer Science(ICS) in the Foundation for Research and Technology-Hellas (FORTH) [15]. The selected thematic areas represent both private/restricted and public environments and include: *Home*, *Work*, *Education*, *Transportation* and *Commerce*, *Leisure* (Figure 2.1).

2.2 Context in AmI

According to Dey(2001) [11],

"Context is 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".

It's easier for the application developers by using this definition to define the context for a given application scenario. If a piece of information can be used to characterize the situation of a participant in an interaction, then that information is context. According to Schilit [33] the context divided into three main categories, *User context*, *Computing context* and *Physical context*, there is also another important category of context, the *Time* Human factors related to the context are structured in three main categories:

- User Context:
  - Information on the User: Information like user's profile (name, age, address), knowledge of habits or interests, emotional state of the user, bio-physiological conditions(special needs that may have because of the biological conditions).
  - User's social environment: Information like co-location of other users (family, friends), social interaction, group dynamics etc.
  - User's task: Information like spontaneous activities, engaged tasks, general goals, circumstances or situations that the user is in etc.
- Computing Context: Surrounding resources for computation, comunication, connectivity, bandwidth, nearby resources (e.g. printers, computers, displays, projectors etc) etc.
- Physical Context:
  - Location: Describe information like absolute location of a user or a device, relative location of a user or a device compared with another user or device of the context or co-location with others.
  - Physical Conditions: Information about noise, lighting, pressure, traffic conditions etc.

- Time: time of the day, day description, time description etc.

In computer science *context awareness* refers to the idea that computers can both sense, and react based on the context information that are taking from their environment. Devices may have information about the circumstances under which they are able to operate and based on rules, or an intelligent stimulus, react accordingly. The term context-awareness in ubiquitous computing was introduced by Schilit (1994) [32] [33] .

Context aware devices may also try to make assumptions about the user's current situation. As the user's activity and the location are important for many applications, context awareness has been focused more in the research fields of location awareness and the activity recognition.

As mentioned in [34], the context aware systems are concerned with:

- the acquisition of the context : Achieving that by using sensors to perceive a situation.
- the understanding of the context: By matching a perceived sensory stimulus to a specific context.
- the application based on the recognized context: By causing actions based on the context.

The study of Ambient Intelligence respecting context and its usage has raise several research issues. Some of those issues are classified in the following categories:

- Sensing context: In order to collect the available context information, sensors are using. The sensors are installing in appropriate positions and capture the information about changes in the context.
- Context representation: One of the main issues in Ambient Intelligence is to find the appropriate model to represent the context information. This model has to address challenges such as interoperability, reusability and information sharing. There are models for structuring context like general context models or context classification models and others like special models e.g. location models. Also there is the solution that come from knowledge representation filed, the *ontologies* (see next chapter).
- Context processing: The delivery of sensed context data at the right time and there storage in the right place is also a fundamental issue for Ambient Intelligence (and Ubiquitous computing) systems. The context data as they come from sensors handlers have different forms, in order to be able to combine all those information the system must transform the context data into the same format.
- Context communication: In order to use all the information that context provide in an Ambient Intelligence system a critical issue is the communication

part with the context. There are many devices and sensors in an AmI system so the communication with context in the (smart) environment is distributed. All the sensor must communicate with a central storage system.

- Interacting with context: Except from collecting and processing context information, the computational devices of the context have also an other role to play, they have to establish the communication between users and environment.

- Reasoning in context: The purpose of the context reasoning is to exploit the real meaning of the data that captured from the sensors. To transform the data into valuable information that AmI system can use in order to conclude about the state of the context.

2.3 Semantic Web

"The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation." [35]

The main purpose of the Semantic Web is driving the evolution of the current Web by enabling users to find, share, and combine information more easily. Humans are capable of using the Web to carry out tasks such as finding the Irish word for "folder", reserving a library book, and searching for the lowest price for a DVD. However, machines cannot accomplish all of these tasks without human direction, because web pages are designed to be read by people, not machines.

The semantic web is a vision of information that can be readily interpreted by machines, so machines can perform more of the tedious work involved in finding, combining, and acting upon information on the web.

The Semantic Web, as originally envisioned, is a system that enables machines to "understand" and respond to complex human requests based on their meaning. Such an "understanding" requires that the relevant information sources be semantically structured.

The Semantic Web is regarded as an integrator across different content, information applications and systems. It has applications in publishing, blogging, and many other areas.

Often the terms "semantics", "metadata", "ontologies" and "Semantic Web" are used inconsistently. In particular, these terms are used as everyday terminology by researchers and practitioners, spanning a vast landscape of different fields, technologies, concepts and application areas. Furthermore, there is confusion with regard to the current status of the enabling technologies envisioned to realize the Semantic Web. The architectural model proposed by Tim Berners-Lee is used as basis to present a status model that reflects current and emerging technologies (Figure 2.2) [16].

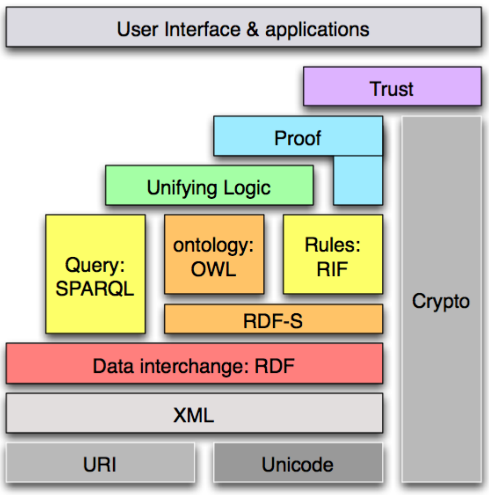


Figure 2.2: The Semantic Web Stack.

2.4 Ontologies in Semantic Web

To describe what ontologies are and what they can offer we will adopt some of the definitions and descriptions that exist in [3]. First of all, to define an ontology, we will repeat T.R. Gruber's definition, later refined by R. Studer: *"An ontology is an explicit and formal specification of a conceptualization"*.

In general, an ontology describes formally a domain of discourse. Typically, an ontology consists of a finite list of terms and the relationships between these terms. The terms denote important concepts (classes of objects) of the domain. For example, in a university setting, staff members, students, courses, lecture theatres, and disciplines are some important concepts.

The relationships typically include hierarchies of classes. A hierarchy specifies a class C to be a subclass of another class C´ if every object in C is also included in C´. For example, all faculty are staff members.

Apart from subclass relationships, ontologies may include information such as

- properties (X teaches Y),
- value restrictions (only faculty members may teach courses),
- disjointness statements (faculty and general staff are disjoint),
- specifications of logical relationships between objects (every department must include at least ten faculty members).

In the context of the Web, ontologies provide a shared understanding of a domain. Such a shared understanding is necessary to overcome differences in terminology. One application's zip code may be the same as another application's area code. Ontologies are useful for the organization and navigation of Web sites. Many Web sites today expose on the left-hand side of the page the top levels of a concept hierarchy of terms. The user may click on one of them to expand the subcategories.

Also, ontologies are useful for improving the accuracy of Web searches. The search engines can look for pages that refer to a precise concept in an ontology instead of collecting all pages in which certain, generally ambiguous, keywords occur. In this way, differences in terminology between Web pages and the queries can be overcome.

In Artificial Intelligence (AI) there is a long tradition of developing and using ontology languages. It is a foundation Semantic Web research can build upon. At present, the most important ontology languages for the Web are RDF, RDFS and OWL, all of which are based on XML. In order to define these languages we will present their definition and their examples from W3C and [3].

2.4.1 XML

Extensible Markup Language (XML) [5] is a simple, very flexible text format derived from SGML (ISO 8879). Originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere. HTML documents do not contain structural information, that is, information about pieces of the document and their relationships. In contrast, the XML document is far more easily accessible to machines because every piece of information is described.

Moreover, some forms of their relations are also defined through the nesting structure. For example, the <author> tags appear within the <book> tags, so they describe properties of the particular book. A machine processing the XML document would be able to deduce that the author element refers to the enclosing book element, rather than having to infer this fact from proximity considerations, as in HTML.

An additional advantage is that XML allows the definition of constraints on values (for example, that a year must be a number of four digits, that the number must be less than 3,000). XML allows the representation of information that is also machine-accessible.

2.4.2 RDF

The Resource Description Framework (RDF) [6] is a general-purpose language for representing information in the Web. This specification describes how to use RDF to describe RDF vocabularies. This specification defines a vocabulary for this purpose and defines other built-in RDF vocabulary initially specified in the RDF Model and Syntax Specification. RDF's vocabulary description language, RDF Schema, is a semantic extension of RDF. It provides mechanisms for describing groups of related resources and the relationships between these resources. RDF

Schema vocabulary descriptions are written in RDF using the terms described in this document. These resources are used to determine characteristics of other resources, such as the domains and ranges of properties.

The RDF data model [1] is similar to classic conceptual modeling approaches such as entity-relationship or class diagrams, as it is based upon the idea of making statements about resources in the form of subject-predicate-object expressions. These expressions are known as triples in RDF terminology. For example, one way to represent the notion "The see is blue" in RDF is as the triple: a subject denoting "the see", a predicate denoting "is", and an object denoting "blue".

A collection of RDF statements intrinsically represents a labeled, directed multi-graph. As such, an RDF-based data model is more naturally suited to certain kinds of knowledge representation than the relational model and other ontological models. However, in practice, RDF data is often persisted in relational database. As RDFS and OWL demonstrate, additional ontology languages can be built upon RDF.

2.4.3 OWL

The OWL Web Ontology Language [27] is designed for use by applications that need to process the content of information instead of just presenting information to humans. OWL facilitates greater machine interpretability of Web content than that supported by XML, RDF, and RDF Schema (RDF-S) by providing additional vocabulary along with a formal semantics. OWL has three increasingly-expressive sublanguages: OWL Lite, OWL DL, and OWL Full.

- *OWL Lite*: supports those users primarily needing a classification hierarchy and simple constraint features. For example, while OWL Lite supports cardinality constraints, it only permits cardinality values of 0 or 1. It should be simpler to provide tool support for OWL Lite than its more expressive relatives, and provide a quick migration path for thesauri and other taxonomies.

- *OWL DL*: supports those users who want the maximum expressiveness possible while retaining computational completeness (either φ or $\neg\varphi$ belong), decidability (there is an effective procedure to determine whether φ is derivable or not), and the availability of practical reasoning algorithms. OWL DL includes all OWL language constructs, but they can be used only under certain restrictions (for example, number restrictions may not be placed upon properties which are declared to be transitive). OWL DL is so named due to its correspondence with description logic, a field of research that has studied the logics that form the formal foundation of OWL.

- *OWL Full*: is meant for users who want maximum expressiveness and the syntactic freedom of RDF with no computational guarantees. For example, in OWL Full a class can be treated simultaneously as a collection of individuals and as an individual in its own right. OWL Full allows an ontology to augment the meaning of the pre-defined (RDF or OWL) vocabulary. It is unlikely that any reasoning software will be able to support every feature of OWL Full.

Each of these sublanguages is an extension of its simpler predecessor, both in what can be legally expressed and in what can be validly concluded. The following set of relations hold. Their inverses do not.

- Every legal OWL Lite ontology is a legal OWL DL ontology.
- Every legal OWL DL ontology is a legal OWL Full ontology.
- Every valid OWL Lite conclusion is a valid OWL DL conclusion.
- Every valid OWL DL conclusion is a valid OWL Full conclusion.

Languages in the OWL family are capable of creating classes, properties, defining instances and its operations.

- Instances: An instance is an object. It corresponds to a description logic individual.
- Classes: A class is a collection of objects. It corresponds to a description logic (DL) concept. A class may contain individuals, instances of the class. A class may have any number of instances. An instance may belong to none, one or more classes. A class may be a subclass of another, inheriting characteristics from its parent superclass.
- Properties: A property is a directed binary relation that specifies class characteristics. It corresponds to a description logic role. They are attributes of instances and sometimes act as data values or link to other instances. Properties may possess logical capabilities such as being transitive, symmetric, inverse and functional. Properties may also have domains and ranges.
  - Data type Properties: Data type properties are relations between instances of classes and RDF literals or XML schema datatypes. For example, *modelName(String datatype)* is the property of *Manufacturer* class. They are formulated using `owl:DatatypeProperty` type.
  - Object Properties: Object properties are relations between instances of two classes. For example, *ownedBy* may be an object type property of the *Vehicle* class and may have a range which is the class *Person*. They are formulated using `owl:ObjectProperty`.

There are some kind of properties in OWL syntax that can express "special" functionality:
`owl:TransitiveProperty` defines a transitive property, such as "has better grade than", "is taller than", or "is ancestor of".

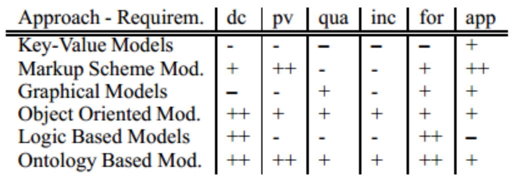
`owl:SymmetricProperty` defines a symmetric property, such as "has same grade as" or "is sibling of".

`owl:FunctionalProperty` defines a property that has at most one value for each object, such as "age", "height", or "directSupervisor".

`owl:InverseFunctionalProperty` defines a property for which two different objects cannot have the same value, for example, the property "isTheSocialSecurityNumberfor" (a social security number is assigned to one person only).

2.5 Benefits of Ontologies in Context aware systems

According to a context modeling survey [37] from Strang and Popien they evaluate many modeling approaches for context representation in terms of distributed composition (dc), partial validation (pv), richness and quality of information (qua), incompleteness and ambiguity (inc), level of formality (for), applicability to existing environments (app). The results are presenting in the Figure 2.3.



Approach - Requirem.	dc	pv	qua	inc	for	app
Key-Value Models	-	-	–	–	–	+
Markup Scheme Mod.	+	++	-	-	+	++
Graphical Models	–	-	+	-	+	+
Object Oriented Mod.	++	+	+	+	+	+
Logic Based Models	++	-	-	-	++	–
Ontology Based Mod.	++	++	+	+	++	+

Figure 2.3: Survey results on context modeling approaches.

There are several reasons for developing context-aware systems based on ontologies [40]

- Knowledge Sharing. The use of context ontology enables computational entities such as agents and services in pervasive computing environments to have a common set of concepts about context while interacting with one another.

- Logic Inference. Based on ontology, context-aware computing can exploit various existing logic reasoning mechanisms to deduce high-level, conceptual context from low-level, raw context, and to check and solve inconsistent context knowledge due to imperfect sensing.

- Knowledge Reuse. By reusing well-defined Web ontologies of different domains (e.g., temporal and spatial ontology), we can compose largescale context ontology without starting from scratch.

Chapter 3

Related Work

In this chapter we present approaches that are related to our work. We divide them into two categories. Works that are describe smart learning spaces and works that describe context representation in ambient intelligent systems.

3.1 AmI Systems

3.1.1 Learning Environments in AmI

Typically, Smart Classroom systems aim to modify the context of the classroom in order to improve the conditions of a class.

[30] focuses on making real-time context decisions in a smart classroom based on information collected from environment sensors, policies and rules. The Context Aware Smart Classroom, CASC, was designed to react to changes in the environment according to rules present in the system. A rules algorithm was designed to check information stored in local database tables and makes decisions based on current system context and preset policies. The prototype system implements two main rules:

What should happen when lecturer enters the room?
Which student should the notes transfer to and how?

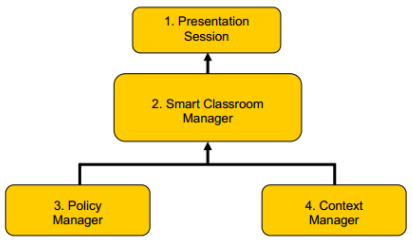


Figure 3.1: CASC Block Diagram.

The smart environment CASC has been developed that supports the delivery of course materials to students by lecturers according to the academic schedule. A block diagram of the key components of CASC is shown in Figure 3.1.

The presentation session component is a client that resides on a local PC in the smart classroom that is responsible for locating the appropriate PowerPoint file and displaying it, starting at the appropriate slide.

The core of the CASC system is the smart classroom manager that provides the adaptive behaviour based on a set of system rules. The manager collects the policy settings of the users and the current context by connecting to appropriate database tables and is responsible for transferring material to students according to student specific policies.

Students and lecturers can log into the system and modify their policies based on the set of options presented. The room operation is set by a room policy that can be modified by the lecturer and the CASC administrator. The notes policy is set by the lecturer and can be used to initiate a change of venue to ensure that appropriate facilities are available in a room.

The context manager is responsible for collecting real-time data and storing the information into appropriate tables in a database for use by the smart classroom manager. The context manager relies on a Bluetooth monitoring daemon on the local computer for communicating with user devices.

The work of Margetis et al. [26] discusses an education-centric approach towards ambient intelligence in the classroom, raising fundamental requirements that should be taken into consideration, in order to efficiently provide genuine students' education enhancement. These requirements are addressed by an integrated architecture for pervasive computing environments, named ClassMATE, which facilitates all necessary mechanisms for context – aware ubiquitous computing in the classroom.

ClassMATE aims to provide a robust and open ubiquitous computing framework suitable for a school environment that: (i) provides a context aware classroom orchestration based on information coming from the ambient environment, (ii) addresses heterogeneous interoperability of AmI services and devices, (iii), facilitates synchronous and asynchronous communication, (iv) supports user profiling and behavioral patterns discovery and (v) encapsulates content classification and supports content discovery and filtering.

Many systems focus on distance learning for a Smart Classroom domain.

This work [31] provides an overview of the technologies used in smart classrooms for distance education by classifying smart classrooms into four categories and discussing the type of technologies used in their implementation. It examines the methodologies being used to make distance learning. Also shows how a similar architecture can be extended to a model where a single local class can simultaneously cater to the needs of multiple remote classrooms and presents a workable solution to some of the technical challenges that exits in multipoint, multiple classroom architecture. It gives an example of a successful implementation of distance education technology that had been used to link university campuses in Japan and in USA.

In an other work [43] multimedia communication systems let teachers and students in different location participate in the class synchronously. Teachers can use multiple natural modalities while interacting remotely students in order to achieve the same result as a classroom with students physically present.

Beside the works in the Smart Classroom domain, there are several intelligent systems focusing on other domains. In [8] presented a smart meeting room system that explores the use of multi-agent systems, Semantic Web ontologies and reasoning. This work provides services and information to meeting participants based on their current situation needs.

3.2 Knowledge Representation

3.2.1 Knowledge Representation in AmI Systems

Apart from the works that focus on improving the environment conditions in an intelligent classroom, there are many works that describe the context representation in ubiquitous environments.

Myoung-woo Hong and Dae-jea Cho [28] in their work describe a conceptual architecture and ontology-based context model for providing context-aware learning services in ubiquitous learning environments. They present a context-aware management architecture to support learning environments. It describes an ontology for context representation in an intelligent school domain. This work is divided into four main constituents:

- Context providing module: It obtains context information from various sensors, a person, an activity agent, and computing entity
- Context knowledge base: It preserves and shares context knowledge on behalf of the personal agent, restricted by resource, and a computing entity.
- Context reasoning engine: It reasons out the context, interpreting context information.
- Learning service coordination module: It coordinates and provides learning services based on context information through a user defined learning support rule.

They propose an ontology in OWL-DL, CALA-ONT that defines Person, Place, Activities and Computational Entities Figure 3.2.

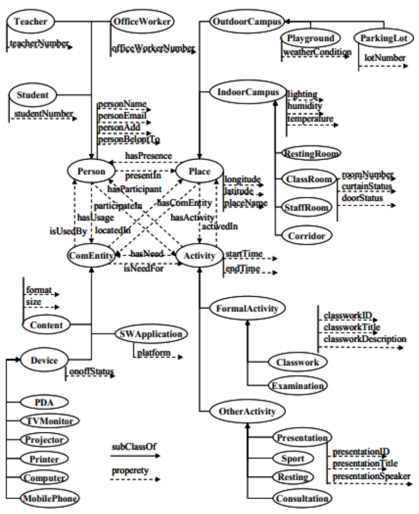


Figure 3.2: graphic presentation about the classification of CALA-ONT model and its property.

In [23], the authors propose a context ontology model for smart meeting space. The proposed context ontology model defines seven major concepts those can be reused as well as provides the basic infrastructure to build an ontology model for smart space environments. As each smart environment may have different kinds of users, devices etc., they design domain-specific ontology for smart meeting space and implement a prototype for this ontology model. They divide their context ontology model into general ontology that can be reused for any application domain and domain specific ontology that is capable of supporting extensibility for the domain of interest. The general ontology (Figure 3.3) is a high-level ontology with general features. Specific ontology is a collection of ontology set that describe the features of a smart meeting space.

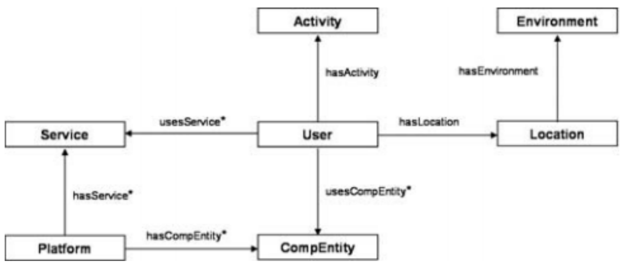


Figure 3.3: General Ontology Overview.

In [19], authors propose a formal context model based on ontology using OWL to address issues including semantic context representation, context reasoning and knowledge sharing, context classification, context dependency and quality of context. The main benefit of this model is the ability to reason about various contexts. Based on their context model, they also present a Service-Oriented Context-Aware Middleware (SOCAM) architecture for building of context-aware services. Their model describes the basic concepts of person, location, computational entity and activity (Figure 3.4). They support automated context reasoning which is the process of reasoning about various types of contexts and their properties.

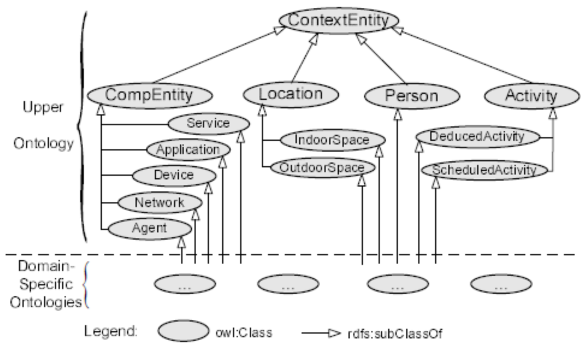


Figure 3.4: Class hierarchy diagram for SOCAM context ontologies.

This work [36] describes a context modelling approach using ontologies as a formal fundament. Authors introduce an Aspect-Scale-Context (ASC) model and show how it is related to some other models. A Context Ontology Language (CoOL) is derived from the model, which may be used to enable context-awareness and contextual interoperability during service discovery and execution in a proposed distributed system architecture. A core component of this architecture is a reasoner which infers conclusions about the context based on an ontology built with CoOL.

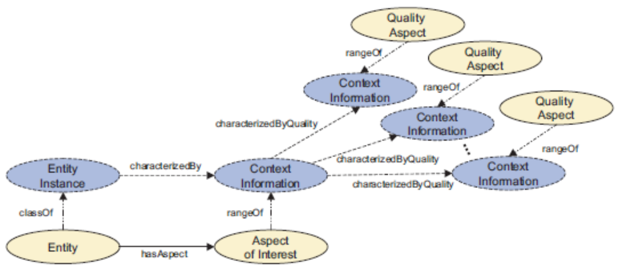


Figure 3.5: Context Information being an Entity itself.

In this work on an abstract level, context information may be seen as content data complemented by some meta data characterizing the content data. Each context information has an associated scale defining the range of valid instances of that type of context information (Figure 3.5). In their system architecture the ontology reasoner is employed to determine interrelationship dependencies and relevance conditions, which may affect a service interaction at any stage.

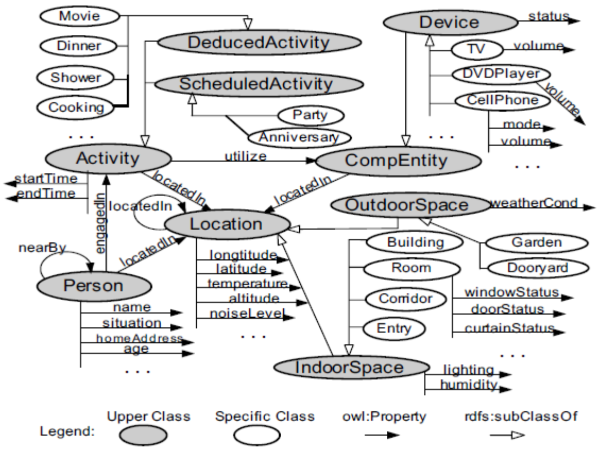


Figure 3.6: Partial definition of a specific ontology for home domain.

In this [40] paper, authors propose an OWL encoded context ontology (CONON) for modeling context in pervasive computing environments, and for supporting logic based context reasoning. CONON provides an upper context ontology that captures general concepts about basic context, and also provides extensibility for adding domain-specific ontology in a hierarchical manner. Based on this context ontology, they have studied the use of logic reasoning to check the consistency of context information, and to reason over low-level, explicit context to derive high-level, implicit context. Figure 3.6 shows a partial definition of specific ontology for a smart home application domain. Besides general classes defined in CONON upper ontology, a number of concrete sub-classes are defined to model specific context in a given environment. The reasoning tasks in this work are grouped into two categories: ontology reasoning using description logic, and user-defined reasoning using first-order logic.

In this work [9], authors propose a universal ontology for smart environments aiming firstly to overcome the limitations of the existing ontologies, and secondly extend its capabilities by adding new environmental aspects such as those mentioned previously. They also demonstrate how their ontology can be used to describe domains as well as applications. The design of their ontology is based on the general idea which states that, a being lives and interacts in an environment with a certain dynamic. From here, they extract the major concepts of the top level ontology, which are: Being, Environment and Dynamic. Figure 3.7 shows these three concepts.

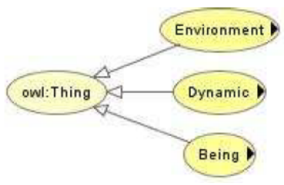


Figure 3.7: Top level concepts of our ontology.

The concept of Environment refers to the environment in which a Being evolves. This environment includes all what is necessary for the evolution of a Being, in terms of habitation, food, travel, leisure, care, etc. In this work, the concept of Environment is subdivided into two main subconcepts. The first subconcept is Tangible, which refers to all things that can be able to be touched or capable of being given a physical existence. The second subconcept is Intangible, which refers to something that cannot have a physical existence like geometric shapes and computational entities.

In this ontology, the concept of Being is divided into two important components. A Physical component that reflects the physical existence of the Being. The second component focuses mainly on the profile of the Being that allows to efficiently characterize the Being's state, in terms of behavior, preferences, disabilities and so on.

The concept of Dynamic is not limited only to activities related to a Being like daily living activities. Therefore, the concept of Dynamic can be broadly classified into two subconcepts such as Virtual, which refers to all computational activities, and Non Virtual which refers to all activities that can be achieved by a human being or a machine such as a robot.

There are also works that are focused on modeling specific parts of the context in intelligent environments like the work of Golemati et al. [24]. The aim is to create a user profile ontology that incorporates concepts and properties used to model the user profile. Existing literature, applications and ontologies related to the domain of user context and profiling have been taken into account in order to create a general, comprehensive and extensible user model. This ontology can be used as a reference model, in order to alleviate the aforementioned issues. This ontology presents information that is mostly static and permanent. More dynamic characteristics like the current position of the user when moving are currently not included. Some of the main classes in upper level ontology are Person, Characteristic, Ability, Interest, Living Conditions, Activity, Education and Profession.

Chapter 4

Intelligent Classroom Assistance - System Components

This chapter provides a synoptic description of the different tasks and components that constitute the Intelligent Classroom Management System. In order to create a full management system of an intelligent space many components have to be combined. The architecture that we propose is initially published in [25] and [13]. Our system consists of five main components:

1. Sensors' handler.
2. Knowledge Base (KB).
3. Rule-based Reasoner.
4. Activity Recognition system.
5. Classroom Assistant.

A simple description of a complete cycle of the system is depicted in Figure 4.1. Nodes represent the different components and arrows represent the data flow between components:

Step 1 Data from sensors and from services of AmI Sandbox [10], or generally of a smart space, after the preprocessing in order to be in appropriate form are stored in the Context KB, i.e. Ontologies in our case. These services provide functionality like localization searching, speech recognition or simply providing the sensor data or devices status. For example, a localization searching service (of a smart classroom), provides data about students' and teacher's current locations, which are stored into our ontology as well as data from lights sensors or data from RFID sensors for people identification. This step runs continuously, in order to avoid missing information. In each iteration, all the changed values about the status of the devices and the changed values

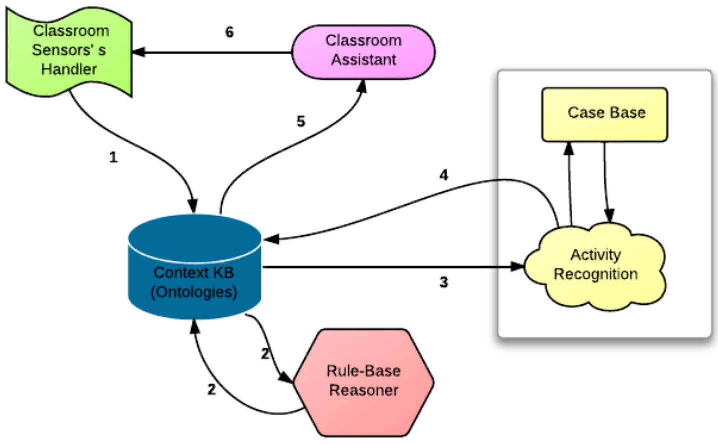


Figure 4.1: System design: Sensor data are stored in the ontology (step 1). A first level of reasoning is applied in sensors' data to identify Simple Events (step 2). Ontology data are parsed from the activity recognition system (step 3), the current activity is recognized and stored back to the ontology (step 4), the recognized activity and the data from the KB are provided to the classroom assistant (step 5). The Classroom Assistant possibly changes the state of the Classroom (step 6).

coming from services are stored in the ontologies. So, the Context KB is always up to date.

Step 2 *SWRL rules* [22] are used for a first level of reasoning on the data coming from sensors. From this process simpler results (*Simple Events*) are concluded e.g "the lights in the classroom are on" or "the student John is located in the classroom" and are stored back to the ontology (Simple Event class).

Step 3 The simple results concluded from step 2 are passed to the Activity Recognition system periodically. The Activity Recognition system parses these Simple Events, to recognize the current Activity. It loads the past cases from a database with correctly recognized activities (Case Base). Using these cases as a machine learning database, it identifies the most possible current Activity. The Activity Recognition system that is used in this work is designed and implemented by V. Efthymiou in his master thesis [12].

Step 4 The Activity that is identified by the Activity Recognition system is stored back to the KB.

Step 5 The classroom assistant system is using the recognized activity and also all the rest knowledge from the KB in order to decide for the kind of the assistance that it will provide to the users of the classroom.

Step 6 Depending on the current recognized activity, that is given as input,the

Classroom Assistant changes the context by giving specific commands to the services of the smart classroom. For example, it lowers the lighting.

4.1 Sensors' Handler

The system that we design can adapt in any smart classroom environment that respects the system's requirements. That means, the existence of smart devices, sensors and other services providing environmental conditions are required. At this point the system is adapted in a smart space located in the Institute of Computer Science of the Foundation of Research and Technology – Hellas (FORTH - ICS).

The FORTH - ICS has initiated a long RTD AmI program [7] aiming to develop pioneering human-centric AmI technologies and Smart Environments, capable of "understanding" and fulfilling individual human needs. Under this program it created a laboratory space of about 100m^2 comprising six rooms, aiming to provide researchers the opportunity to bring along and share their know-how and resources in order to obtain hands-on experience and experiment in a highly flexible setting. In this space, various AmI technologies and applications are installed, integrated and demonstrated, and multiple ideas and solutions are cooperatively developed, studied and tested [10].

AmI services are defined through a tool called Idlematic which creates and keeps service interfaces. All services communicate through FAMINE middleware, responsible for creating, connecting and consuming these services. It is CORBA-based and provides support for C++, Java, .NET, Python, and Flash/ActionScript languages.

Sensor data are handled like mouse/keyboard events in Java. Some (already implemented) services (which we found by using the Idlematic tool) can be used to catch the sensor signals and inform the user about the type of signal and what it means (by defining an event). In the AmI Sandbox there are sensors for sound, image, lighting, interaction, "smart" devices and more, including cameras, RFID readers, IRIS scanner, door-chair controllers, touch screens, projectors, PDAs, speakers and workstations. When an event occurs then it can be caught using event handlers. These event handlers are responsible to store the sensor data into the respective ontology.

4.2 Knowledge Base

Generally, the Knowledge bases contain the actual information or knowledge that has to be stored and reused. This information has to be structured in some way. In this work the information is structured according to the ontology language that we use (OWL [27]) and the ontologies that have been defined based on this language (Chapter 5). This knowledge can be seen as a collection of instances of the classes defined by the ontologies.

Because of the nature of sensors' data (different formats etc), after the handling processes the aim is to transform the data in a readable and useful form in order to be stored into a knowledge base that obeys to an ontology schema.

In order to store the knowledge we propose a context ontology model for an intelligent university Classroom that responds to students' and teachers' needs. Our context ontology is divided into two hierarchical parts, upper-level ontology and low-level ontologies. All the ontologies that we have design for the needs of this domain are presented with more details in Chapter 5.

4.3 Rule-based Reasoning

In our implementation we try to transform our scenarios for smart classroom into rules. This rule based approach is implemented by using Semantic Web Rule Language (SWRL) [22]. The first step is to capture data from sensors and services from Smart Classroom and store them into the ontology (e.g. status of devices).

Because of the nature of those data (sensors' data) SWRL rules are applied on them in order to produce more useful results for our scenarios. After the reasoning part the results are stored into the *Simple Event* class (Chapter 5). SWRL rules are also applied for user assistance part. After the activity recognition part, the aim is to manage to assist the activity that takes place into the Smart Classroom in order to be easier for the users to participate on the activity.

Chapter 6 presents in details the reasoning techniques and the rules that we use.

4.4 Activity Recognition

The responsible system for the activity recognition part was designed by V. Efthymiou for the purpose of his Master thesis [12]. It is based on machine learning and is presented in this paragraph.

After the SWRL rules trigger, the resulting *Simple Events* that occurred within a time frame are sent to the Activity Recognition system, building an unsolved case. One or no activity is then recognized and the solved case is added to the case base.

Based on a given set of activities that are to be recognized, a case base is initially created and classified manually. It is essential for this dataset to be a product of real observations and not just random cases. Instead of implementing the machine learning algorithms, he uses WEKA [21] with default parameters, so his dataset is stored as an arff file.

In our domain each case represents a time-frame typically lasting 10 seconds, but generally depending on the specific domain. This means that for an hour of reasoning we need about 360 cases/lines and for a day of reasoning we need around 8640 cases/lines in our case base/arff file. Soon it becomes obvious why it is important to have some data reduction algorithms. WEKA's Resample filter can perform this task and return a new arff file as a result.

When the challenge is a real-time activity recognition, there is a need for accuracy as well as speed. V.Efthymiou approach aims to take advantage of the rich expressiveness that ontologies can offer and provide solid answers, using machine learning algorithms, like Support Vector Machines (SVMs) or Bayesian networks. The key factor that led to the design of a new system, using machine learning, is the lack of speed observed in the already existing systems that use algorithms such as kNN. Apart from that, the robustness and accuracy of Bayesian networks led to a system faster and more accurate than other systems that we are aware of.

There are some limitations in this activity recognition system that affect the functionality of our work. So,we have design our system by taking into account these limitations. Every case in our scenarios has only one solution. This means that no parallelism is achieved when it comes to activities. For example a person might be talking to the phone and watching TV at the same time. The system would (ideally) return only one of these activities as a result. Moreover, the returned activity is only referring to one individual. In an environment with more than one individuals (like a smart classroom) the activity can be described as a "team" or "general" activity (like slide presentation, lecture etc). This means that we have not found a way to return one activity for each of the individuals present.

4.5 Assistance

The point of our work is to manage to assist the *Activities* that can take place in a Smart Classroom after their identification by the Activity recognition part. The kind of assistance that we can provide is based on the context. The adaptation of the context to humans needs. The general purpose of every cognitive system is to be able to change the environment conditions based on the current conditions given as input. That happens in order to make it easier for the users to focus on the main activity e.g. *Presentation* and not to be disturbed by changing the environment by themselves. More specifically, if the lights in the Smart Classroom are off and the lighting level from the outside environment is also low, the *Smart Classroom Assistant* will turn the lights on.

In our particular domain (classroom), apart from the environmental conditions that can be adapted, we can also assist the activities by using information stored in the ontology which are related to the particular activity. For example, by using information from user's calendar to remind him the upcoming lecture that he has to participate. More details in Chapter 6.

In order to accomplish the adaptation of the environment for the users' needs, we apply reasoning once more to all the data that are stored into our ontologies (Chapter 5). After this part, we configure the devices of the Smart Classroom with specific values, for example by setting the value "on" on the projector through the projector service.

Apart from the event handlers, Idlematic and Famine, of AmI Sandbox, also provide the ability to change the context, by calling some methods. These are methods that interact with the environment (such as the ability to close a door, turn a device on etc).

4.6 Architecture

One of the aims of our system is to be able to run in real-time in order to be useful for the users. It has to get the data from the sensors, to work on them by using SWRL, to run the activity recognition system, in order to identify the activity happening and in the end, to support the specific activity. All these tasks should execute continuously, not only once and may need to execute at the same time. For that reason the architecture of our system is based on parallelism. In order to manage to execute all the task independently and in parallel we use *threads*. Each task corresponds to a different thread. So, we have a thread for the *sensors' handler*, a thread responsible to execute the *SWRL rules* and produce the *Simple Events*, a thread that runs the *activity recognition* system and in the end a thread that is responsible for the *assistance*. The communication of the threads depicts in the data that are sharing e.g. the ontologies or activity recognition results. This sharing part is safe since only one thread can have access in shared memory each time.

The sensors' handler thread is runs continuously through the system execution, receives data from the sensors and stores them into ontologies (shared memory). The SWRL thread is executed every 10 seconds and enables the *Simple Events* that are appropriate based on the current context situation. After that, the Activity recognition thread is executed and identifies the current activity according to the enabled Simple Events. In the end of the execution cycle the "assistant" thread is running. The "10 seconds" is an approximate duration that is suited in our domain because the activities does not change more often.

If threads were not used, then the total execution time of a "reasoning cycle" would be significantly greater. Moreover, another more significant problem is that if the thread that is responsible for sensor handling and also the one responsible for writing Simple Events to the ontology, were not running continuously, would be "inactive" for a considerable time. This means that some signals, some information about the users actions during this time of inactivity would be missed.

Chapter 5

Context Modeling

In this chapter we analyse the modeling of the context of an intelligent university classroom and generally the modeling of the knowledge about the living and the non-living things which are related to this specific a smart space. All this information is described and represented through ontologies. The ontologies are designed to capture knowledge related to smart spaces and more specific to smart classroom.

Our context ontology *"Smart Classroom"* is divided into two hierarchical parts. The first is the upper-level ontology that capture general features and characteristics of AmI spaces and the low-level ontologies that are focused to describe specific concepts based on an intelligent classroom. All the ontologies (upper-level and low-level) are imported to *SmartClassroom* ontology, connected with each other through object properties and extended based on the smart classroom domain (Figure 5.1).

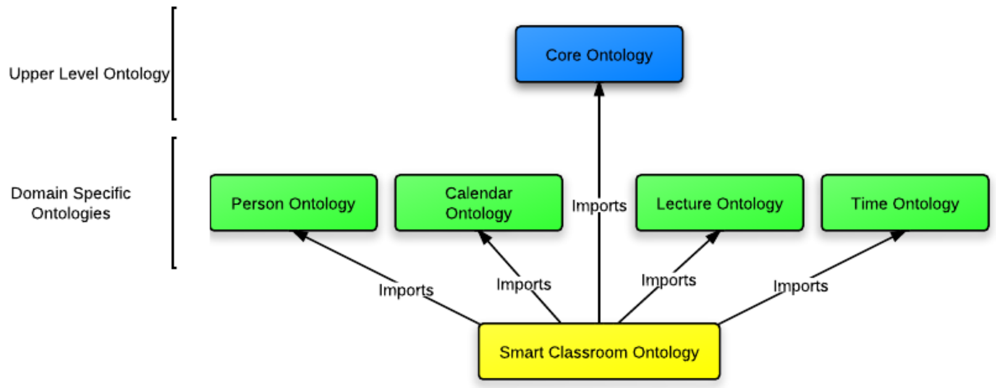


Figure 5.1: Importing and extending the core ontology (Upper level) and the domain specific ontologies and relate them in order to create the Smart Classroom Ontology.

All the ontologies that we will present are implemented in OWL DL language [27] because is expressive enough in order to cover the characteristics that we want to represent.

5.1 Core Ontology

As it is defined in [42], in information science, an upper ontology or *Core Ontology* is an ontology which describes very general concepts and characteristics that are the same across all knowledge domains. An important function of an upper ontology is to support very broad semantic interoperability between a large number of ontologies which are accessible ranking "under" this upper ontology. As the rank metaphor suggests, it is usually a hierarchy of entities and associated rules that attempts to describe those general entities that do not belong to a specific domain. The *Core Ontology* in this work, is designed by using and extending an already proposed ontology, CONON [40], in a way that can be reused to model different smart space environments, like smart homes or smart meeting spaces. The context model of this ontology is structured by a set of abstract entities like *Computational Entity*, *Person*, *Activity*, *Location*, *Simple Event*, *Environment* and *Time* (Figure 5.2). All these entities are widely used in context representation through ontologies and capture only general characteristics that can model the context of different AmI systems.

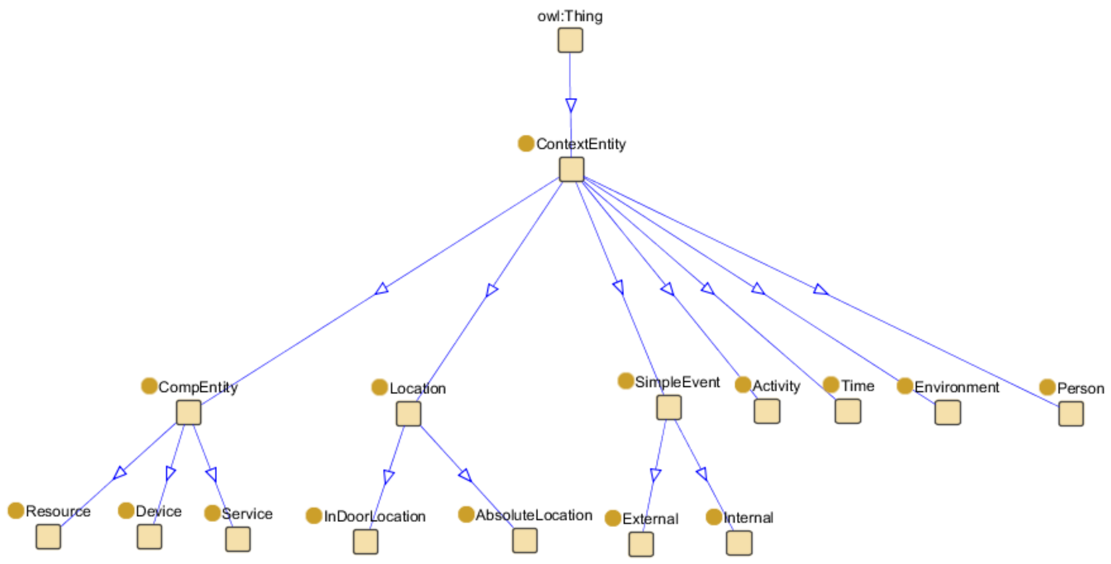


Figure 5.2: Class Hierarchy Diagram of Upper-Level of Intelligent Classroom.

Bellow, all the classes and the subclasses of *Core Ontology* (Figure 5.2) are analysed:

- CompEntity: It is a super class that describes the computational entities of the smart space. It gathers not only the computational devices that exist in the classroom but also the resources and the services. There are three sub classes of *CompEntity*, the *Service*, *Device* and *Resource*.

  - Service: Services in a smart space and specially in AmI Sandbox [10] are software implemented methods that can be used to catch sensors' signal and "make" the devices to interact with the users of the smart space. For example in AmI Sandbox, including others, there is a service that can control the lights of the room. All the information about the services, like the inputs and outputs that are expecting/giving is stored into *Service* class.
  - Device: Through this class the devices that the users of the smart space can interact with, like TVs or computers are represented.
  - Resource: Is a sub class of *CompEntity* that describes all the resources of the smart spaces. For example resources could be the desks and the chairs in a classroom or even the doors and the windows.

- Simple Event: In our perspective *Simple Events* are the simpler activities that compose a main activity. All the simpler components that help us conclude in a bigger activity. Usually simple events are the actions that happen in the smart space and change the status of the devices or activate the services. These actions may be momentary or with duration. For example, an instance of class *Simple Event* can be "the lights are tuned on" or "the window is open" or "Prof. Plexousakis is located in the classroom". The knowledge that is captured into *Simple Event* ontology obtained after processing on sensors' data that are stored into *Service* ontology. The data of the sensors for the lights of the room, for the windows and the data from the sensor that is responsible to define the location of a person, are taken into account in these examples. Moreover, simple events may not be only changes of the environment but also other information obtained after the rule-based reasoning on the ontology's data, e.g. according to classroom's calendar there is an arranged presentation at this moment.

  The simple events are divided into two categories, *Internal* and *External* events.

  - Internal: Is a sub-class of *Simple Event* and describes simple events that take place inside the smart space that we deal with. For example, "the lights are tuned on" is an internal simple event.
  - External: Is a sub-class of *Simple Event* and describes simple events that take place out of the smart space that we deal with. For example, "the noise of the road" outside of the smart classroom is an external simple event.

- Activity: This class describes the main activities that may happen in a smart space. Usually, an activity is the combination of many simpler activities

(*Simple Events*) and is not momentary. However, depends form the domain and the specific scenarios that have to be modeled. For example, in a smart home domain an activity may be "Jack is watching tv".

- Person: The user plays a vital role in smart space environments. All the services in the smart space should be adapted to user's needs. With this class we can describe the characteristics and the needs of a user.
- Location: This class expresses the location of the smart space by representing specific rooms and areas inside the rooms, of the smart space. Moreover, through this class we can express the absolute location of a user or a resource even their relative location with other users or resources.
  - AbsoluteLocation: Is a sub-class of *Location* and expresses the absolute location of a Person, Device or Resource based on x, y, z coordinates.
  - InDoorLocation: Is a sub-class of *Location* and can expresses the rooms of a smart space. i.e. "Classroom 201" or "Corridor A".
- Environment: Is a class that describes information about the environmental condition of the smart space. The indoor temperature, the noise level, the humidity and the lighting. The knowledge about the environmental condition is very important in order to control devices and to give proper condition for any kind of activity that takes place in a smart space.
- Time: Through this class the duration of activities and the time description are modeled. The knowledge of the time plays a major role in context representation because it is so important identifying an activity as knowing the order that activities happen in a smart space.

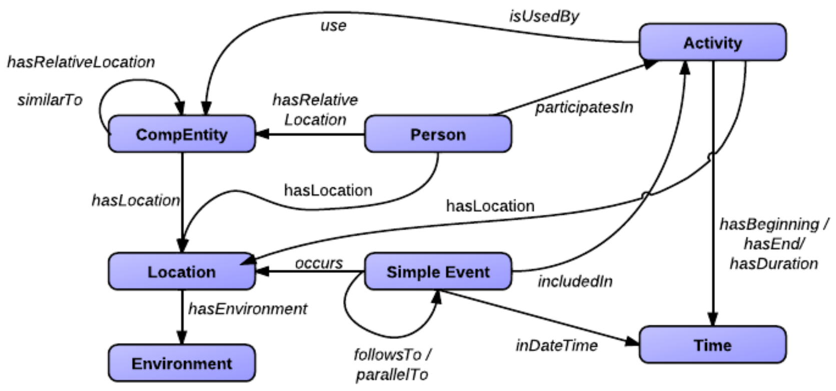


Figure 5.3: Upper-Level or Core Ontology of Intelligent Classroom. Domain-Range of properties.

All the classes of Core ontology are connected through object properties that are shown in Figure 5.3.

- hasRelativeLocation: This property is sub property of a property called *located* and relates a Person or a CompEntity to a CompEntity. By using this property we can describe the relative location of a Person or a Device/Resource to another Device/Resource. The property *hasRelativeLocation* has the following sub-properties: *above*, *behind*, $inFrontOf$, *into*, *on*, $rightTo$, $leftTo$, *under* and $facing$. With these properties we can express the relative locations of a Person, i.e. John is in front of the desk.
- hasLocation: The property *hasLocation* is also sub property *located*. It is used to express where a Person, a Device/Resource or an Activity is located. Their position in a specific smart space (e.g. `John hasLocation Corridor01`).
- hasEnvironment: This property relates a room or an area of a smart space with a set of environmental conditions.
- participatesIn: This property relates a Person with an activity that he participates.
- occurs: This property relates a Simple Event with a Location. Describes the specific location that takes place the simple event.
- followsTo / parallelTo: These properties relate two or more Simple Events to each other. We can define the time order that the events happen and to know if an event follows another. In some cases is very important to have the knowledge about the order of events to identify an Activity that combines more than one Simple Events.
- includedIn: This property relates a Simple Event with the Activity it participates.
- inDateTime: This property relates a Simple Event with the specific time description that it starts.
- hasBeggining / hasEnd / hasDuration: These properties relate an Activity with a Time description. The point is to describe the duration of the activity and the specific time that it starts/stops.
- use and isUsedBy: These properties relate the Activity with the Computational Entities (devices, resources) use during the specific activity e.g. a projector is needed during a presentation. Are symmetric properties.
- similarTo: This property has as domain and range the class *CompEntity*. It relates the computational entities that are similar, have similar characteristics.

The data type properties of Core ontology are presenting in the Figure 5.4. We emphasize to *isActivated* and *isIdentified* properties. These two properties have as domain the Simple Event class and the Activity class respectively. They Relate the instances of those two classes with the values *true* or *false* and their purpose is to express if a simple event is active or not and if an activity is identified or not.

Moreover, two important properties are the properties *currentStatus* and *changedStatus*. These two properties have as domain the Device class and the Resource class. With the property currentStatus we represent the current status of each device or resource as it is given from the sensors' handlers. With the changedStatus we represent the status that we want to has a specific device or resource after the assistance. We need two different properties to represent the status of devices for testing reasons. If the two properties have different value for a specific device we can deduce that there is a problem with the specific device and it can not response our commands.

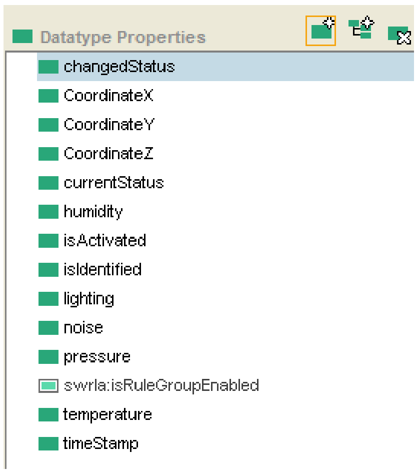

Datatype Properties
changedStatus
CoordinateX
CoordinateY
CoordinateZ
currentStatus
humidity
isActivated
isIdentified
lighting
noise
pressure
swrla:isRuleGroupEnabled
temperature
timeStamp

Figure 5.4: The data type properties of Core ontology.

5.2 Smart Classroom Ontologies

In the previous section we describe our high-level ontology (upper-ontology) which captures general context knowledge about physical world in pervasive computing environments. It defines the basic concepts of person, location, activity, computational entity, environment, simple event and time as shown in Figure 5.2. In order to describe more expressive scenarios we have to enrich the existing concepts based on the specific domain. The domain in this work is an *Intelligent University Classroom* and based on it we extend and analyse more the basic concepts of the upper-ontology. The details of these concepts are defined in the domain-specific ontologies that are presented in this section.

5.2.1 Person Ontology

The User is the most important entity in a smart space environment. All the actions happening in a pervasive computing environment have as a main purpose to sub serve the user's needs. Besides, the reason that the smart spaces existed in the first place is to make the user's life more comfortable by doing operations he used to do by himself, e.g turn on or turn off devices when is needed. For that reason the user has a central role in the modeling of the context in an ambient intelligence system.

There are works that create user profiles by capturing the general features of a person like [18]. In our work we proposed an ontology that models person profiles, focusing in their academic nature. We describe and represent characteristics, abilities, preferences and information about knowledge level that are useful to explore when we consider a smart environment which has as main purpose to assist users that participate in educational procedures.

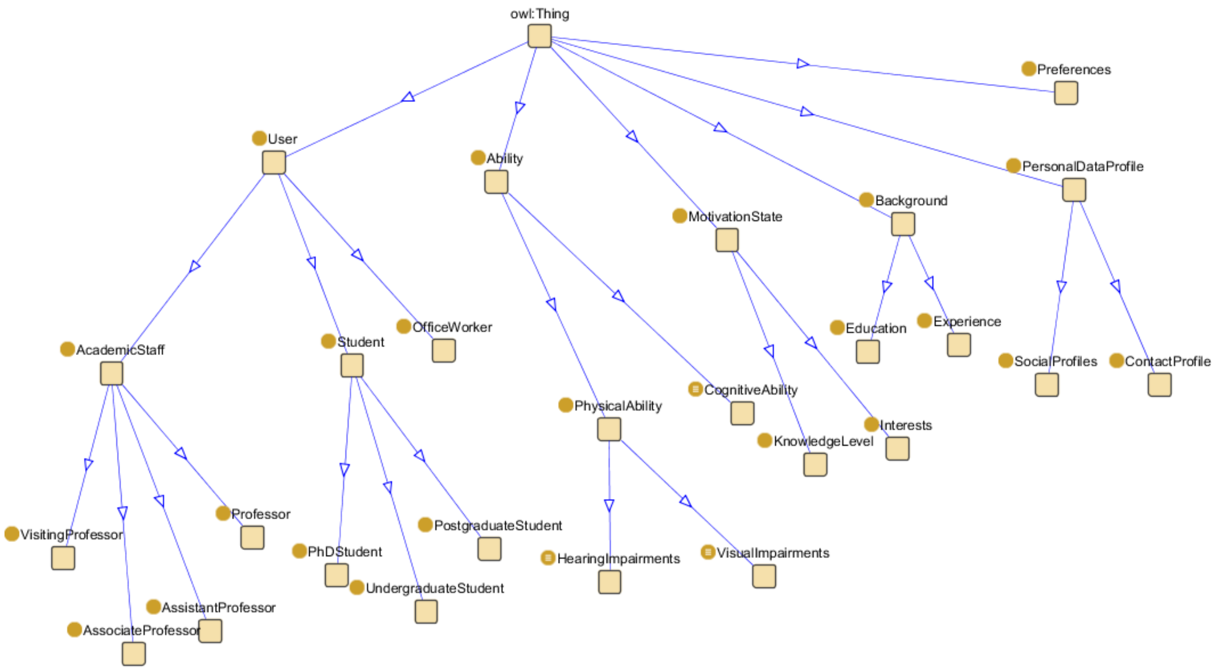


Figure 5.5: Person Ontology of Intelligent Classroom. Class Hierarchy.

Below the classes of the *Person ontology* are presented (Figure 5.5).

- User: The *User* is the major class of the Person ontology. It is the class that links the Person ontology with the *Smart Classroom* ontology. Its instances are divided into different other classes like *Academic Staff* that describes users with an academic position like *Professors*, *Visiting Professors*, *Associate Professors* and *Assistant Professors.* Also, the class *User*, describes *Students* and divides them in different sub-classes according to their educational level like *Under Graduate Students*, *Post Graduate Students* and *PhD Students.* Except from the students and the professors, in a smart classroom environment other users like *Office Worker* or system administrators may appear.

- Ability: With the *Ability* class we can express the physical or the cognitive abilities that a user may have. This differentiation is achieved through the two sub-classes *Physical Ability* and *Cognitive Ability.* With the Physical Ability class physical impairments of users that may affect the learning procedure are described e.g. a myopic or a total blind user have to be treated differently by the system in some cases. The two categories that we divide the physical impairments are: *Visual Impairments* and *Hearing Impairments.* Moreover, there is the *Cognitive Ability* sub-class, that describe cognitive impairments that users may have and can affect the learning procedure like problem about focus or bad long term memory.

- MotivationState: In this class, we model the interests of a user on an academic subjects as well as the level of his knowledge in different topics. It is important to have information about the interests of a user while building a system that has as main purpose to make the life easier for each user of the system. Also, because the smart environment that we describe is an intelligent classroom the information about the knowledge level of the user in many/different subjects is very needful. For example, a user that has beginner's knowledge in Java will need different kind of assistance compared to a user that has advanced knowledge in the same subject.

- Background: In this class of *Person Ontology* we can describe the background of a user, his *Education* and his *Experience.* The sub-class *Background* models information about what kind of degree the user has, from which university etc. The *Experience* sub-class models information about the experience of the users in spesific topics e.g John has two years experience in C++ programming.

- PersonalDataProfile: This class models personal information about the user. The name, age and gender of the user, also contact information like, address, e-mails or phone numbers and information about possible social profiles.

- Preferences: In this class any other preferences that user may have are modeled and can be proved useful for the system e.g. John prefers low level lighting when he is writing exams.

In the Figure 5.6 we can see the main properties of the Person ontology. The blue double arrows present the object properties that connect the main class *User* with the other classes of the ontology. All of them are inverse properties, which means that If the property $p1$ is stated to be the inverse of the property $p2$, then if X is related to Y by the $p1$ property, then Y is related to X by the $p2$ property. For example, if *John* (instance of class User) `hasPersonalDataProfile` $profile1$ then $profile1$ `isProfileOf` *John*.

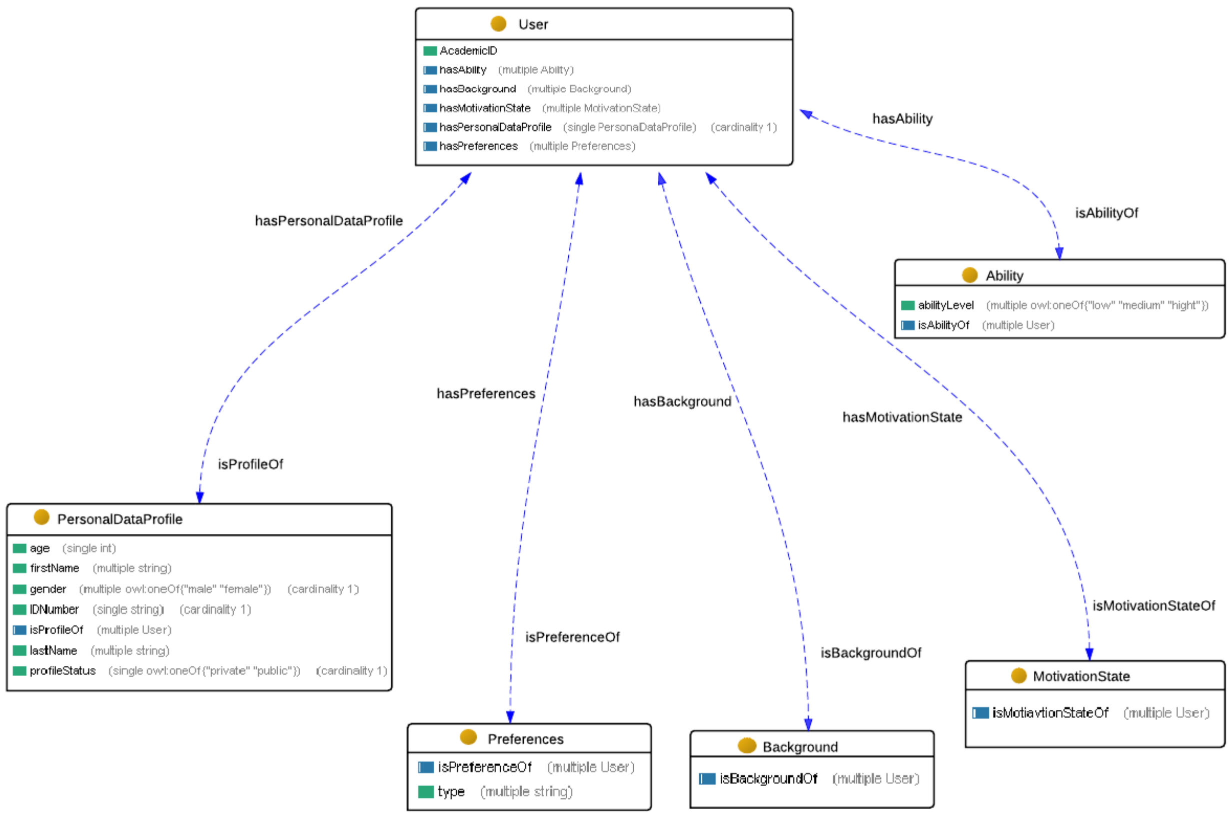

User
AcademicID
hasAbility (multiple Ability)
hasBackground (multiple Background)
hasMotivationState (multiple MotivationState)
hasPersonalDataProfile (single PersonaDataProfile) (cardinality 1)
hasPreferences (multiple Preferences)
hasAbility
isAbilityOf
Ability
abilityLevel (multiple owl:oneOf{"low" "medium" "hight"})
isAbilityOf (multiple User)
hasPersonalDataProfile
isProfileOf
hasPreferences
hasBackground
hasMotivationState
PersonalDataProfile
age (single int)
firstName (multiple string)
gender (multiple owl:oneOf{"male" "female"}) (cardinality 1)
IDNumber (single string) (cardinality 1)
isProfileOf (multiple User)
lastName (multiple string)
profileStatus (single owl:oneOf{"private" "public"}) (cardinality 1)
isPreferenceOf
isBackgroundOf
isMotivationStateOf
MotivationState
isMotiavtionStateOf (multiple User)
Preferences
isPreferenceOf (multiple User)
type (multiple string)
Background
isBackgroundOf (multiple User)

Figure 5.6: Person Ontology of Intelligent Classroom. Connection between classes, basic properties.

In Table 5.1 are presenting the data type properties of Person ontology. For each property we know the domain and the range and also the cardinality and constraints about allowed values that the property can take.

5.2.2 Calendar Ontology

With the *Calendar Ontology* we can model the schedule of a user of the smart classroom. We can have information about the events that a user has to participate,

Name	**Domain**	**Range**	**Cardinality**	**Constraints**
abilityLevel	Ability	string		allowed values: "low","medium", "hight"
AcademicID	User	int	exactly 1	
professorID	AcademicStaff	int		SubPropertyof: AcademicID
studentID	Student	int		SubPropertyof: AcademicID
address	ContactProfile	string		
age	PersonalDataProfile	int		
deegreeType	Education	string		allowed values: "Bachelor", "Master", "PhD"
email	ContactProfile	string		
experienceDescription	Experience	string		
experienceDuration	Experience	string		
experienceType	Experience	string		
firstName	PersonalDataProfile	string		
gender	PersonalDataProfile	string	exactly 1	allowed values: "male", "female"
grade	Education	float		
IDNumber	PersonalDataProfile	sting	exactly 1	
knowledgeDescription	KnowledgeLevel	string		
knowledgeLevel	KnowledgeLevel	string		allowed values: "Beginner", "Intermediate", "Advanced"
lastName	PersonalDataProfile	string		
organization	Education	string		
phoneNumber	ContactProfile	sting		
profileStatus	PersonalDataProfile	string	1	allowed values: "private", "public"
socialProfileType	SocialProfiles	string		allowed values: "FBProfile", "GPlusProfile", "LinkedInProfile", "TwitterProfile"
title	Education	string		
type	Preference	string		
userName	SocialProfiles	string		

Table 5.1: This table shows the data type properties of Person ontology. Describes the domain and the range of each property and also the cardinality and restrictions about allowed values.

what kind of events are (lecture, exam, etc.), where each specific event is going to take place etc. Is very useful for a system that has to assist user's activities to have all these information about his schedule, e.g. the system can inform the user for a time change of an activity that he will participate.

Moreover, the *Calendar Ontology* is used not only to model the calendar of a user, but also to model the calendar of the *smart classroom* that we design. All the schedule of the classroom, the lectures, the exams the breaks with their participants and all the details about each activity(start time, end time, topic of the lecture) can be expressed through the ontology *Calendar*. Having the knowledge for the schedule of the classroom the assistance part becoming easier and more accurate e.g. Monday morning 9:00 am to 11:00 am a lecture of cs180 is scheduled in the *smart classroom* and a projector is needed. The management system of the classroom takes this information from the calendar and checks the available projector. If there is any problem in the available projector, the administrator of the classroom is informed about the problem in order to fix it before the scheduled start of the lecture.

As we can see in Figure 5.7 the root class *Thing* has the subclasses *Calendar* and *Calendar Event*.

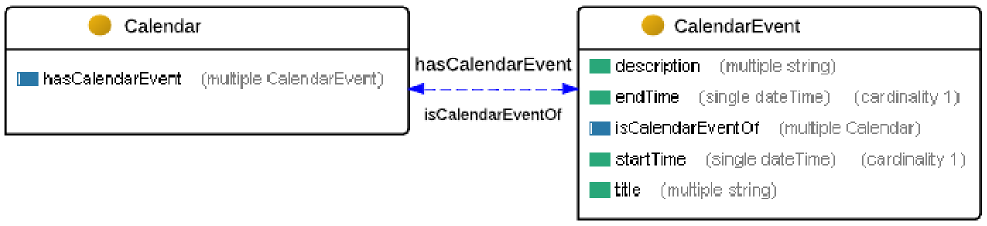

Calendar
hasCalendarEvent (multiple CalendarEvent)
hasCalendarEvent
isCalendarEventOf
CalendarEvent
description (multiple string)
endTime (single dateTime) (cardinality 1)
isCalendarEventOf (multiple Calendar)
startTime (single dateTime) (cardinality 1)
title (multiple string)

Figure 5.7: Calendar Ontology of Intelligent Classroom. Connection between classes, basic properties.

- Calendar: Through the class *Calendar* the linking between the user and the smart classroom with the calendar ontology is achieved. Each user or classroom has only one calendar.

- Calendar Event: Each instance of the class *Calendar* is related to one or more instances of the class *Calendar Event*. This class models the events existing in a calendar of a User or a smart classroom. There are four data type properties (Figure 5.7 green squares) `start time`, `end time`, `description` and `title` that describe a calendar event.

5.2.3 Lecture Ontology

The domain that we examine in this work(*Smart Classroom*) is directly related with the procedures of a classroom, the lectures the exams etc. For that reason we had to find a way to represent the content of the lectures that take place in the classroom. We propose the *Lecture Ontology*, in this ontology we express the learning material(notes, tutorials etc.) that is related to the lectures and the presentations that happen in the classroom.

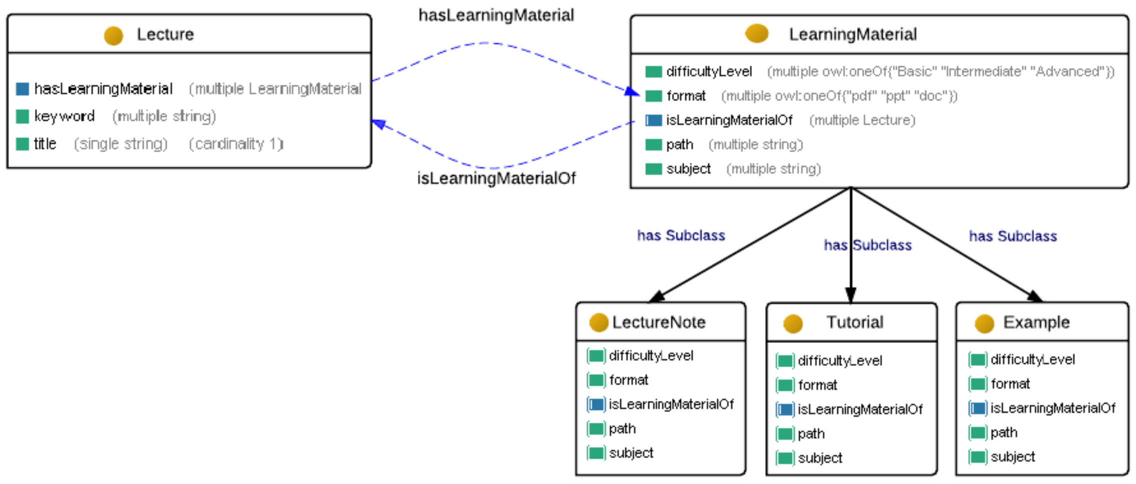

Lecture
hasLearningMaterial (multiple LearningMaterial
keyword (multiple string)
title (single string) (cardinality 1)
hasLearningMaterial
isLearningMaterialOf
LearningMaterial
difficultyLevel (multiple owl:oneOf{"Basic" "Intermediate" "Advanced"})
format (multiple owl:oneOf{"pdf" "ppt" "doc"})
isLearningMaterialOf (multiple Lecture)
path (multiple string)
subject (multiple string)
has Subclass
has Subclass
has Subclass
LectureNote
Tutorial
Example
difficultyLevel
format
isLearningMaterialOf
path
subject

Figure 5.8: Lecture Ontology of Intelligent Classroom. Connection between classes, basic properties.

In the Figure 5.8 presenting the *Lecture Ontology* with the classes *Lecture* and *Learning Material*.

- Lecture: Through the class *Lecture* the linking between the smart classroom ontology with the lecture ontology is achieved. In the smart classroom ontology, the Lecture class is linked directly with the *Calendar Event* class, of the *Calendar* ontology. Each instance of calendar event class that is relevant with lecture or presentation is linked with an instance of Lecture class. As we can see in the Figure 5.8 in the Lecture class there are two data type properties (green squares) `keyword` and `title` and one object property(blue square) `hasLearningMaterial` that links the instance of the lecture with the corresponding learning material(e.g. note of the lecture). This property is inverse with the property `isLearningMaterialOf`.

- Learning Material: In this class the learning material related to the lecture are described. Categorized into three sub-classes, *Lecture Note*, *Tutorial*, *Example*. There are four data type properties for this class, `subject`: the subject of the given lecture, `difficultyLevel`: the difficulty level of the current note or tutorial, e.g. there are notes for basic knowledge for the specific subject and for advanced knowledge. If a user (student) is new in a specific field he will need the notes for the basic difficulty level.

5.2.4 Time Ontology

In order to represent the sentence of time and duration in smart classroom ontology we use an ontology standardized from W3C, the *Time Ontology* [39]. This ontology provides a vocabulary for expressing facts about topological relations among instants and intervals, together with information about durations, and about date-time information. In the Figure 5.9 are showing the basic classes of Time ontology.

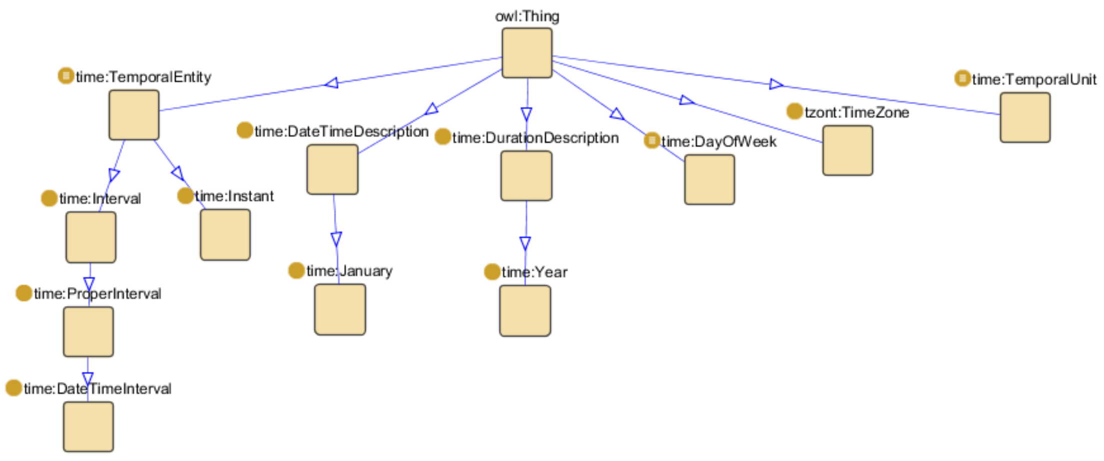


Figure 5.9: Time Ontology, W3C standard of time representation. Basic classes.

5.3 Smart Classroom Ontology

In the final *SmartClassroom Ontology* all the previous ontologies are imported. After that, some properties are defined in order to connect the classes of the independent ontologies. The extra properties that we define are shown in the Table 5.2.

Name	**Domain**	**Range**	**Cardinality**	**Constraints**
currentLecture	Classrooms	Lecture: Lecture		
EventHasLocation	Calendar: CalendarEvent	Classrooms	exactly 1	
hasCalendar	Person: User ∪ Classrooms	Calendar: Calendar		inverse: isCalendarOf
hasEndTime	Calendar: CalendarEvent	Time: DateTimeDescription	exactly 1	
hasLecture	Calendar: CalendarEvent	Lecture: Lecture		
hasLecturer	Lecture: Lecture	Person:User		inverse: isLecturerOf
hasParticipants	Calendar: CalendarEvent	Person: User		
hasRFID	Person: User	RFIDTags		
hasStartTime	Calendar: CalendarEvent	Time: DateTimeDescription	exactly 1	
isCalendarOf	Calendar: Calendar	Person: User ∪ Classrooms		inverse: hasCalendar
isLecturerOf	Person: User	Lecture: Lecture		inverse: hasLecturer
relatedWith	Core: Activity	Lecture: Lecture		symetric: relatedWith
useCompEntity	Calendar: CalendarEvent	Core: CompEntity	exactly 1	allowed values: "male", "female"
enable	Core: Service	boolean		allowed values: "true", "false"
lightsLevel	Lights	sting		allowed values: "low", "medium", "hight"
participants	Classrooms	int		
studentsParticipants	Classrooms	int		SubPropertyof: participants
teachersParticipants	Classrooms	int		SubPropertyof: participants

Table 5.2: All the extra properties that are defined in order to connect the classes of the ontologies: *Core*, *Person*, *Calendar*, *Lecture*, *Time*.

5.4 Ontologies Evaluation

The following criteria are used to evaluate the ontologies and are influenced by [17] [20]. Because there is no other mechanism to evaluate a self-created ontology we will comment the following criteria based on our ontology architecture. The considered facts touch issues like flexibility, extensibility and completeness of the ontology, consistency and granularity of the concepts and properties, as well as the language formalism applied.

Reusability, standardization Increasing the reusability implies the maximization of using the ontology among several independent tasks, while usability rather means the maximization of applications using the ontology for the same or similar tasks.*The ontologies that we proposed in this thesis can reused to models other domain like smart office or smart meeting space. Lecture ontology can be used for the representation of learning material in smart classroom of other level e.g smart classroom of elementary school. The Calendar ontology can used to models the calendar of users in many other smart spaces.*

Flexibility, extensibility This criterion refers to the possibility of adding new definitions to the ontology without altering the existing dependencies. *In our ontologies it does not need much effort to change or extend the ontologies because of the distinguish between the concepts.*

Genericity Ontologies are about the integration of knowledge and the relationship of resources. It is important that a generic and multi-functional backbone is provided for the modeling of information. Ontologies that are applicable across large sets of domains are referred to as upper ontologies and belong to the most general category of ontologies. *In our perspective there is an upper-level ontology which captures general features and characteristics of main concepts.*

Granularity This criterion is highly related to the details of the concepts defined and the scope of their meaning. A fine-grained ontology defines concepts for closely related objects, while in contrast a coarse-grained model knows more general and distinguishable terms. Insofar granularity is related to the diversity and coverage of individual concepts. Upper ontologies are often coarse-grained, while application ontologies become more fine-grained. *We can support both coarse-grained and fine-grained because of the two level of ontologies that we developed.*

Consistency This criterion is about the existence of explicit or implicit contradictions in the represented ontological content. *The check of consistency for our ontologies is accomplished with the usage of Pellet reasoner provided by Protege. All the independent ontologies are consistent and the final SmartClassroom Ontology is consistent too.*

- For the Core ontology: 0.3 seconds
- For the Person ontology: 0.215 seconds

- For the Lecture ontology: 0.12 seconds
- For the Calendar ontology: 0.185 seconds
- For the SmartClassroom ontology: 0.53 seconds

Completeness According to [17] an ontology is complete if it (explicitly or implicitly) covers the intended domain. A ontology can thus be complete without covering all possible aspects, if its target domain is restrictive to some particular world. *As far as we can say our ontologies covers the relevant concepts that are needed to model the context of a smart classroom.*

Readability This measure accounts the usage of intuitive labels to denominate the ontological entities. The importance of this criterion is limited to the understandability of humans and is hence of lower importance to context-aware computing. *Our ontologies are expressing enough with representative labels for the classes and the properties.*

Scalability Ontology engineers distinguish three types of scalability: cognitive scalability which refers to the possibilities of humans to oversee and understand the ontology, engineering scalability which refers to the available tool support that is still quite limited for large scale ontologies, and reasoning scalability which refers to the difficulties of reasoning with large data sets. *We believe that is not quite difficult for someone to understand the meaning of the concepts using in our ontologies. As for engineering scalability, we used the Protege tool (chapter 9) which is very useful to create your own ontologies.*

Language, formalism This criterion looks at the language used to model the ontology and the expressive thereof. Possible languages are standard FirstOrder Logic and subsets thereof like Description Logic, as well as non-monotonic rule languages like investigated in Logic Programming. UML-based languages are excluded, as they do not have a model theoretic semantics. *We use DL with its standardized syntax OWL.*

5.5 Summary - Limitations

In this chapter we presented the ontologies that we have designed and implemented in order to capture knowledge depicts in the smart environment that we focus on. We first design a general ontology that can be applied in any smart environment and continuously we specialized it in our smart classroom domain. The ontologies that we present may not be able to describe all the information that someone can imagine about a classroom but are descriptive enough to represent the required knowledge for our proposed scenarios and also can be easily enriched in order to express more information. They are completely focused on the smart classroom domain from all the perspectives. We have a *Person* ontology that focuses on a classroom domain, also *Calendar* ontology for persons and classrooms and *Learning material* ontology.

Chapter 6

Reasoning

In the previous chapter we analysed the way that all the information about the context and the data from the sensors of the the smart environment that we explore (smart classroom) are stored and representing by using ontologies. We have design several ontologies in order to represent as much knowledge as needed to conclude useful results. All the environmental conditions, the characteristics and preferences of the users and also the sensor's data are stored into ontologies.

In this chapter we will present how all this given information from the context can be exploited. In order to have useful conclusions that will help us identify the activities occurring in a classroom or to assist the users, we have to process the stored knowledge. This process is achieved by defining specific rules, which are executed by a rule engine and are responsible to draw conclusions based on the given *scenarios*.

We have divided the rules into two main categories based on their results. The first category of rules exams the given knowledges that is stored to KB and enrich the KB with inferred knowledge coming from the execution of the rules. The second category is responsible for the assistance part. After the execution of the rules we were based on the conclusions in order to apply changes to the context. With more details, those rules, are responsible for changing the devices status or for calling specific services.

6.1 Knowledge inference through rule-based reasoning

As we described in previous chapters, after storing the context data (coming from sensors or form users' profile) into the ontologies, we apply rule-based reasoning by using SWRL rules. These rules are responsible to process the stored data and add the inferred knowledge into ontologies. As it is described in Figure 6.1 according to the procedure that is following, the stored (asserted) knowledge of the KB is processed by the SWRL rules and the inferred knowledge coming from the execution imported in KB.

We decide to use SWRL for this part of reasoning because it is an extension of

OWL language (the language that we have build our ontologies) based on OWL-DL and OWL-Lite and is able to reason the context information stored in OWL ontologies. Moreover, it is able to to define and reason the changes in context information coming from previous rules executions.

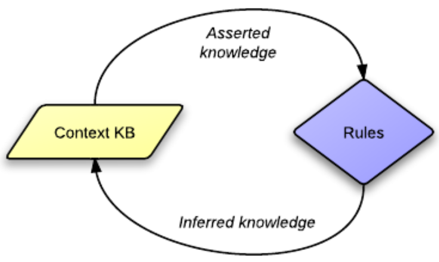


Figure 6.1: Reasoning on KB data and add inferred knowledge back to ontology.

Rules for enabling Simple Events. This category of rules includes the rules that are responsible to export simple results (as we called Simple Events). The results are subsections of the main activity (Lecture, Exam) that we want to identify and are given to the activity recognition system in order to be able to recognize the activity. In the class *Simple Event* there are instances that represent all the simple events that can occur based on our scenarios. The SWRL rules can activate or deactivate the simple events by relating them with the values "true" or "false" through the data type property *isActivated*.

These rules have the form:

context data ∧ ... ∧ *context data* → *isActivated("simpe event",true)*

or

context data ∧ ... ∧ *context data* → *isActivated("simpe event",false)*

Some representative examples of such rules are given below:

All the classes and properties belong to the Core ontology are used with the prefix Core in front of them and respectively the classes or properties of the other ontologies e.g. the class Environment of Core ontology declared like `Core:Environment` and the property hasEnvironment declared like `Core:hasEnvironment`.

Rule 1 `Projector(Projector1)` ∧ `Core:currentStatus(Projector1, on)` →

`isActivated(Projector_is_On, true)`

Description: This rule checks the status of a device (Projector1) and if the status is "on" activates the simple event "Projector_is_On". The activation obtained by connecting the the instance "Projector_is_On" with the value "true" through the data type property *isActivated*.

Rule 2 `Lights(ClassroomLights)` ∧ `Core:currentStatus(ClassroomLights, on)` →
`isActivated(Classroom_Lights_On, true)` ∧
`isActivated(Classroom_Lights_Off, false)`

Description: This rule checks the status of the lights in the classroom (ClassroomLights) and if the status is "on" enables the simple event "Classroom Lights On" and disables the simple event "Classroom Lights Off" in case that was enable.

Rule 3 `ProjectorSurface(Prj1)` ∧ `Person:Student(?s)` ∧
`Core:inFrontOf(?s, Prj1)` →
`isActivated(Student_inFrontOf_Projector, true)`

Description: In this rule the relative location of a student inside the classroom is expressed. The "?s" is a variable which does not represent a specific student but can be applied to all the instances of class *Student*.

Rule 4 `SmartBoard(Board1)` ∧ `Person:AcademicStaff(?a)` ∧
`Core:inFrontOf(?a, Board1)` →
`isActivated(Professor_inFrontOf_SmartBoard, true)`

Description: In this rule expressed the relative location of an academic staff inside the classroom. The "?a" is a variable again and the Academic staff is in front of(relative location) the Smart Board (Board1).

Rule 5 *`Person:AcademicStaff(?a)`* ∧ *`Core:hasLocation(?a, SC1)`* →
`isActivated(Professor_LocatedIn_Classroom, true)` ∧
`isActivated(Professor_LocatedIn_Office, false)` ∧
`isActivated(Professor_LocatedIn_Hallway, false)`

Description: In this rule the position of an academic staff is expressed. The professor is located in the Smart classroom (SmartClassroom1), so the respective simple event (Professor_LocatedIn_Classroom) is activated and the remaining contrasting simple events (Professor_LocatedIn_Office and Professor_LocatedIn_Hallway) are deactivated.

Rule 6 *SmartClassroom(SC1)* ∧ *hasCalendar(SC1, ?c)* ∧
Calendar:hasCalendarEvent(?c, ?evnt) ∧ *hasStartTime(?evnt, ?st)* ∧
hasEndTime(?evnt, ?et) ∧
swrlb:greaterThanOrEqual(CurrentTime, ?st) ∧
swrlb:lessThan(CurrentTime, ?et) ∧
Calendar:description(?evnt, "Slide Presentation") →
isActivated(ClassroomCalendarEvent_LectureSlidePresentation, true)

Description: This rule activates the simple event "Classroom Calendar Event - Lecture Slide Presentation" which describes that at the current moment there is a calendar event (?evnt) for the classroom with the description "Slide Presentation". The *swrlb:greaterThanOrEqual* and *swrlb:lessThan* are SWRL-build Ins for comparisons.

Rules for adding other inferred knowledge. There are other rules that applying on the data of KB and add inferred knowledge back to ontologies. This knowledge does not necessarily enable the simple events. It may conclude other kind of information based on the data that are stored into the ontology.

An example of such rules is shown below:

Rule 7 *Core:Person(?u)* ∧ *hasRFID(?u, ?rfid)* ∧
commnunicateWith(?rfid, ?rfidReader) ∧
Core:hasLocation(?rfidReader, ?loc) ∧
Core:currentStatus(?rfid, "on") →
Core:hasLocation(?rfid, ?loc) ∧ *Core:hasLocation(?u, ?loc)*

Description: This rule can estimate the current location of a user (?u). It is a general rule and can be applied to all the users of the system (professor, students) because of the variable ?u. Each user is connected with a RFID tag. These rules check whether the rfid tag of a user (?rfid) communicates with a rfid reader(?rfidReader). The RFID tag is a resource and its location is known so the user as well as the RFID tag have the same location with the RFID reader.

6.2 Reasoning for User Assistance

As we mentioned before (section 4.5), as for all the cognitive systems the same purpose also holds for our system: to make the life of their users easier. How do we accomplish this? By changing the context and adapting it in humans needs. More specifically in our system the assistance focuses in the following tasks. The system can change the environmental conditions, if it is needed, without the intervention of the user. We can assist the specific activities happening in the classroom. For example if the activity is "Presentation using projector" and the lighting level is very high the system will adapt the lighting level in order to facilitate the procedure.

Moreover, an other difficult task in our system is there may exist many users at the same moment in the smart classroom. Each of them has different characteristics and different needs. So the assistance has to be different for each of them. We use content information that are stored into the ontologies in order to accomplish that. For example, if we have the information from the calendar of a student who has to participate in a specific lecture the system will send him a notification one hour before the beginning of the lecture. However, the system will not send the notification to all the users that are stored into the ontologies, but only to those that have to participate to the lecture. In an other example, if the activity that takes place in the Smart Classroom is a *lecture presentation* we can find all the information that describe this particular lecture in the Lecture ontology (Chapter 5). After that, we can assist each particular user by providing the file of the presentation or other additional notes related to the subject of presentation. The assistance will be different for each user according to his profile. If someone has a low level background on the presentation's subject, the Smart Classroom Assistant will provide him also with additional material tutorials, or basic notes about the subject.

As we can see in the Figure 6.2 in this part of reasoning we can use information from the Context, from the users' profiles and also the identified activity.

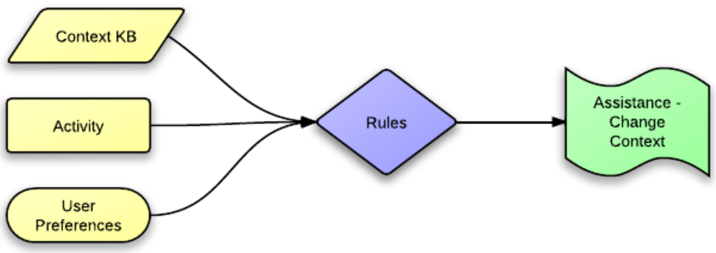


Figure 6.2: Reasoning using sensors data and/or the identified Activity and/or the user preferences and assist the users.

Using recognized activity. In this category we have rules that are responsible to draw conclusions having as a presupposition the identification of a specific activity. For each particular activity that can be recognized from the system based to our scenarios, there are different rules for the assistance part.

Except from the SWRL rules that we use in order to draw conclusions and to proceed to the assistance part by calling the appropriate service that can adapt the devices, we also use SQWRL queries. We can query our ontologies and extract the information needed in order to call the services. More specifically, for the services that we want to have different behaviour for each specific user or for a specific sub-category of users we execute SQWRL queries, get the required information in order to give this information as input to the services.

Some examples of this category of rules are presenting bellow.

Rule 8 *Core:isIdentified(LectureWithSlidePresentation, true)* → *Core:changedStatus(ClassroomLights, "low")* ∧ *Core:changedStatus(ClassroomBlinds, "off")* ∧ *Core:changedStatus(ClassroomWindows, "off")*

Description: The identified activity is Lecture with slide presentation. A part of the assistance in this activity is to change the status to the lights, the blinds and the windows of the classroom. We give the right values in the property *changedStatus* for each resource.

Rule 9 *Core:isIdentified(LectureWithSlidePresentation, true)* ∧ *Core:Person(?p)* ∧ *Person:hasPreferences(?p, ?pref)* ∧ *Person:type(?pref, "English Language")* → *sqwrl:select(?p)*

Description: The identified activity is Lecture with slide presentation. We check for users with specific preference in English Language. For those users we will call a service that is responsible for translation.

Rule 10 *Core:isIdentified(LectureWithSlidePresentation, true)* ∧ *relatedWith(LectureWithSlidePresentation, ?lect)* ∧ *Lecture:title(?lect, ?title)* ∧ *Core:located(?user, SmartClassroom1)* ∧ *Person:hasMotivationState(?u, ?mot)* ∧ *Person:knowledgeDescription(?mot, ?title)* ∧ *Person:knowledgeLevel(?mot, "Beginner")* → *sqwrl:select(?u)*

Description: By using this query we want to get all the users that are participate to the activity *Lecture with slide presentation* and their knowledge level in the subject of the lecture(?lect) is low. Will send learning material that suits to these students.

Rule 11 *Core:isIdentified(LectureWithSlidePresentation, true)* ∧ *Core:participatesIn(?p, LectureWithSlidePresentation)* →

sqwrl:select(?p)

Description: With this query we get the users that participates in the identified activity *LectureWithSlidePresentation*.

Rule 12 *Core:isIdentified(LectureWithSlidePresentation, true)* $\wedge$ *hasCalendar(?user, ?calendar)* $\wedge$ *Calendar:hasCalendarEvent(?calendar, ?evnt)* $\wedge$ *Calendar:title(?evnt, "LectureCS180")* $\rightarrow$ *sqwrl:select(?user)*

Description: By this query we can get all the users that have to their calendar the event "LectureCS180".

By combining the results from the rules 12 and 13 we can have the users that had to participate in the activity *LectureWithSlidePresentation* for the lecture "LectureCS180" but they didn't appear.

Without using recognized activity. This category consists of rules that are part of the assistance in the classroom, moreover can be executed independently of the activities that may happen into the classroom (e.g. lectures, exams etc.). They can fire any time and the main purpose is to cause changes to the context if the environmental conditions require it. Most of those rules can be applied not only to a smart classroom environment but also to other smart environments like smart homes. The aim is to adapt the environmental conditions or to change the status of the devices if it is needed. For example, in case that the temperature inside the smart classroom is lower than the normal temperature the system will turn on the hitter. Or the opposite, if the temperature is higher than the normal standards the system will adapt the temperature by putting the right degree to the air-condition system.

Some examples of this category of rules are presenting bellow.

Rule 13 `Rule: SmartClassroom(SC1)` $\wedge$ `Core:Environment(SC1Env)` $\wedge$ `Core:hasEnvironment (SC1, SC1Env)` $\wedge$ `Core:noise(SC1Env, ?noise)` $\wedge$ `swrlb:greaterThan(?noise, 70)` $\wedge$ `Windows(SC1Window)` $\wedge$ `Core:currentStatus(SC1Window, on)` $\rightarrow$ `Core:changedStatus(SC1Window,off)`

Description: The noise level in a smart classroom (SC1) is higher than the acceptable limits and the windows are open. According to this rule, the system will change the status of the resource (window).

Rule 14 `Rule: SmartClassroom(SC1)` $\wedge$
`Core:Environment(SC1Env)` $\wedge$

```
Core:hasEnvironment(SC1, SC1Env) ∧
Core:temperature(SC1Env, ?temperature) ∧
swrlb:greaterThan(?temperature, 28) →
Core:changedStatus(airCondSC1,on)
```

Description: The temperature in a smart classroom (SC1) is higher than the acceptable limits. According to this rule, the system will turn the air condition of the classroom (airCondSC1) on.

Exceptional occasions. According to our scenarios they are predefined operations that can occur inside the classroom. But because the world is not perfect always may happen unexpected actions that may incommode the learning procedure. Based on our scenarios the exceptional occasions that may appear generally concern technical problems that can happen unexpectedly and encumber the normal process e.g. problem in a device used in a lecture presentation. Moreover, as exceptional occasion is considered the different than the expected behaviour from the users of the system, e.g. a student had to participate in the lecture presentation according to his calendar but he didn't appear. To identify these exceptional occasions we use the information that is stored into our ontologies. For the computational entities of the classroom we have defined two properties that express the status of each device. The first one (*Core:currentStatus*) has the status of the device as it is the current moment. The second one (*Core:changedStatus*) has the status that the specific device must have after the reasoning. If those two properties have different values for the same device means that there is a problem with this device and for some reason is can not response.

The assistance that the system can provide in these exceptional occasions differs depending on the specific scenario and the specific occasion. The general perspective is that if a problem with a device occurs then the system checks for similar devices that can replace the problematic device (content adaptation) and also inform the administrator of the classroom about the problem. If the exceptional occasion is about the behaviour of a user then the approximation is to inform the involving users. However, the assistance depends on the specific situation, the scenario and the approach.

Some examples of this category of rules are presenting bellow.

Rule 15 *hasCalendar(SmartClassroom1, ?calendar) ∧*
Calendar:hasCalendarEvent(?calendar, ?evnt) ∧
hasStartTime(?evnt, ?st) ∧ Time:day(?st, ?day) ∧
Time:day(CurrentTime,?currentDay) ∧
swrlb:equal(?day,?currentDay) ∧
Core:currentStatus(?comEnt, "error") ∧
Core:similarTo(?comEnt, ?comEnt2) ∧
Core:located(?comEnt2, SmartClassroom1) →

useCompEntity(?evnt, ?comEnt2)

Description: In order to avoid the cancellation of a class because of technical problems on devices that are needed for the process the system checks the status of the devices before the beginning of the class. We use the information from the calendar of the Smart Classroom, check for events that are going to take place today. Then, checks the status of the the device that is using to this event(?comEnt). If the status is "error", checks if similar devices (?comEnt2) are located in the SmartClassroom1 and relates the event of the calendar (?evnt) to use the working device(?comEnt2).

Rule 16 *hasCalendar(SmartClassroom1, ?calendar)* ∧ *Calendar:hasCalendarEvent(?calendar, ?evnt)* ∧ *hasStartTime(?evnt, ?st)* ∧ *Time:day(?st, ?day)* ∧ *Time:day(CurrentTime,?currentDay)* ∧ *swrlb:equal(?day,?currentDay)* ∧ *useCompEntity(?evnt, ?comEnt)* ∧ *Core:currentStatus(?comEnt, "error")* → *enable(SendInformationToAdministrator, true)*

Description: If there is no similar device to replaces the problematic device (or event to the case that there is a similar device) we inform the administrator of the classroom for the problem. He had to fix the problem or to replace the problematic device.

6.3 Summary - Limitations

In this chapter we described the way that we elaborate the information that is stored into our ontologies in order to assist the users of the classroom or to produce extra, more useful knowledge. This elaborations accomplishing by using rule-based reasoning on ontologies' data. SWRL rules are using in this part. This language is full compatible with OWL, the language of our ontologies, and is ideal for context reasoning, but it has some limitations. It can not support negation, disjunction neither priority between rules, thus we have make some agreements.

In order to substitute in some way the negation and the disjunction that SWRL does not provide we use the stored into ontologies knowledge and create opposite rules to produce the opposite conclusions. For example if the status of projector is on or off we have:

If the status is off we can not express the negation, so we have the following:

```
Projector(Projector1) ∧ Core:status(Projector1, off) →
isActivated(Projector_is_On, false) ∧ isActivated(Projector_is_Off, true)
```

An other problem is the conflicts that depicts from the lack of the priority between rules. The conflicts that we can have in this work are focusing on the status of devices or the call of the services. Which means that a conflict for example can be when two different rules give the opposite command for the status of the projector at the same time. These conflicts appear in the rules of the assistance part in the point that a general assistance rule fires in the same time with a rule for a specific activity. For that reason we decide to give priority to the rules of the specific activity assistance by removing the devices and services, that are using from these rules, from the general assistance rules.

Chapter 7

Demonstration

In order to run a real-world demonstration of our system, we have used the AmI Sandbox. The data are real, and they were not based on assumptions. Of course, the ideal evaluation of this system would require its use in a real classroom with people that will be its final users, but this demo is the closest we could get. To be more accurate, the data are based on the physical presence of three human (2 students and 1 professor) in the AmI Sandbox, who acts according to our scenario.

This demo is based on the ontologies presented in Chapter 5, in which Activities are recognized based on Simple Events. The available resources (that could be useful for us) in the AmI Sandbox at the time of the demo were:

- a hospital bed, capable to recognize whether someone is on it or not (is using instead of chairs)
- two televisions, capable of returning their state (on/off, volume, channel, etc) (are using instead of projectors)
- window blinds, capable of returning their state (open/closed)
- lights, capable of returning their intensity
- RFID readers, capable of returning if someone/something approaches them
- MailSender, service responsible to send e-mail to specific receivers

Because we had to build a demo using these resources, the activities presented are limited. These are:

- Lecture Presentation
- Break

The sensor data are stored in the ontology and then periodically parsed from the SWRL thread and the result is passed to activity recognition thread. After the execution of the activity recognition thread the assistance thread executes the needed swrl rules for the assistance of the specific activity.

In this specific scenario, two students are entering to the classroom. The professor also is entering to the classroom. They are identified by their RFID readers and their profiles are enabled. The classroom calendar is checked and a simple event for the "lecture presentation in calendar" is enabled. After the identification of the professor into the classroom and the activation of the simple event "lecture presentation in calendar" the projector is opened.

At this moment the active simple events are:

- Projector is on
- Students located in classroom
- Professor located in classroom
- Classroom Calendar Event: Lecture with slide presentation

The activity recognition is executed and identifies the activity: "Lecture With Slide Presentation". After that the responsible for the assistance thread is running and executes the SWRL rules for the assistance of the activity "Lecture With Slide Presentation". Those rules causes changes to the environmental conditions of the classroom like, lowering the lighting level and also closing the blinds. Moreover, the file of the lecture presentation is sending to the students by using the Mail Sender Service and also students profiles are checked for their knowledge level on the subject of the lecture. One of the students has advanced knowledge level on the subject and the other has low knowledge level. So by using again the Mail Sender Service, additional learning material sent to the student that has low knowledge level on the specific subject.

For time-saving reasons after few minutes the professor informs the students that they will make a break. So the students and the professor are living from the classroom. Moreover, in classroom's calendar still their is an active "Lecture with slide presentation" because of has not passed the end time of the event.

At this moment the active simple events are:

- Projector is on
- Students not located in classroom
- Professor not located in classroom
- Classroom Calendar Event: Lecture with slide presentation

The activity recognition is executed and identifies the activity: "Break". After that, the "assistance" thread is executed again, rising the lighting level and opening the blinds. Because of the professor is not located into the classroom the assistant is turned the projector off for energy saving reasons.

After few minutes, the students and the professor are entering to the classroom again. At this moment the active simple events are:

- Projector is on
- Students located in classroom
- Professor located in classroom
- Classroom Calendar Event: Lecture with slide presentation

The activity recognition is executed and identifies again the activity: "Lecture With Slide Presentation". All the assistance about the projector, the lights and the blinds reapplied. As the lecture is in process an exceptional occasion is happening. An error is occurring to the projector that is using for the presentation. The system identifies the error and setting the presentation file to an other projector that is also located into the classroom in order to continue the lecture process.

The demo is recorded with the help of two videos. The first video shows the students and the professor in the AmISandbox doing some activities, following the scenario that we described. The second video is a recording of the GUI controller that appears on the screen of the laptop that runs the system. This is a GUI that is only created for the need of the demo, and it is not a part of the system.

As for the performance of the system, for the first classification, we have to load the ontology, which cost some time. So for the first classification, we had to wait for average **7.5 seconds** to load the ontology with approximately **100 instances**. However, from the second run and thereafter, we did not have to load the ontology again. The execution of the SWRLs for this scenario costs approximately **0.56 seconds** for each repetition. Moreover, the execution of the Activity recognition costs approximately **0.96 seconds** for each repetition.

The SWRL thread and the Activity recognition thread are not running continuously. The performance of those threads is repeated every 3 seconds in this demonstration.

Chapter 8

Conclusions

8.1 Synopsis

In this work, we have introduced the use of a real-time AmI system in a smart classroom. We have presented how the ontologies can be used to model the context in a smart environment and especially in a smart classroom. We described our upper ontology and our domain specific ontologies *Person*, *Calendar*, *Lecture* and *Time*. We have also presented the way that all the domain specific ontologies are combining and importing to the *Smart Classroom* ontology. Moreover, we exploit the advantages of ontologies in order to model the context and introduce as well a method for extracting information from an ontology and using it in an activity recognition system.

We have presented the architecture of the system and the data flow between the components. First of all, we take data from sensors, store them into our ontologies after that rule-based reasoning applying and producing *Simple Events*. These simple events are giving as input to an activity recognition system which is responsible to identify the occurring activity based on specific scenarios (presentation, exam, break). After the identification of the activity, rule-based reasoning applying again and activates the right services in order to assist the specific activity.

We proposed five scenarios/activities that illustrate how such a system could assist its users. The scenarios are based on the basic activities that can appear in a university classroom.

8.2 Limitations - Future Directions

Even if we tried to model as many features and characteristics of the context as we could into our ontologies always will be other characteristics depict to other scenarios that can not supported by our ontologies. So the ontologies that we propose can evolved if other scenarios required it.

Also, an other limitation is that we do not have a mechanism to confirm the result of the activity recognition system. If the activity recognition system identifies an activity our system will assist this specific activity. But, always there is a chance the identification to be wrong. A future direction can be to include a method that will be responsible to confirm the result of the activity recognition.

An other limitation/future direction can be the replacement of the existing activity recognition system with an other that can identified parallel activities and also would identified the activity of each specific user. The existing system can not identify the activity of each user, for that reason we have adapt our scenarios to more general activities that deal the users like group.

Also, we can enrich the system with more scenarios in order to assist the users in more situations.

We can replace the SWRL rules for the assistance part with an other technique that will be more expressive and also it will support negation, disjunction and priority between the rules.

And last but not least, future work includes more experiments and demonstration of our system with more users scenarios in a smart space oriented to the requirements of the domain.

Chapter 9

Appendix A

In this Chapter we briefly describe the main tools used for the implementation of our system.

9.1 Implementation tools

9.1.1 Protégé

Protégé [14] is a free, open-source platform that provides a growing user community with a suite of tools to construct domain models and knowledge-based applications with ontologies. At its core, Protégé implements a rich set of knowledge-modeling structures and actions that support the creation, visualization, and manipulation of ontologies in various representation formats. Protege can be customized to provide domain-friendly support for creating knowledge models and entering data. Further, Protege can be extended by way of a plug-in architecture and a Java-based Application Programming Interface (API) for building knowledge-based tools and applications.

The Protégé platform supports two main ways of modeling ontologies:

- The Protégé-Frames editor enables users to build and populate ontologies that are frame-based, in accordance with the Open Knowledge Base Connectivity protocol (OKBC). In this model, an ontology consists of a set of classes organized in a subsumption hierarchy to represent a domain's salient concepts, a set of slots associated to classes to describe their properties and relationships, and a set of instances of those classes - individual exemplars of the concepts that hold specific values for their properties.

- The Protégé-OWL editor enables users to build ontologies for the Semantic Web, in particular in the W3C's Web Ontology Language (OWL). "An OWL ontology may include descriptions of classes, properties and their instances. Given such an ontology, the OWL formal semantics specifies how to derive its logical consequences, i.e. facts not literally present in the ontology, but entailed by the semantics. These entailments may be based on a single document or multiple distributed documents that have been combined using defined OWL mechanisms" (see the OWL Web Ontology Language Guide).

Protégé is supported by a strong community of developers and academic, government and corporate users, who are using Protégé for knowledge solutions in areas as diverse as biomedicine, intelligence gathering, and corporate modelling.

We used Protégé to create our ontologies for context modeling, as well as the Protégé OWL-API in order to facilitate the communication of our Java code with the created ontologies.

9.1.2 SWRL and Jess Rule engine

Semantic Web Rules Language (SWRL) is a proposed Semantic Web language combining OWL DL (and thus OWL Lite) with function-free Horn logic, written in Datalog RuleML. Thus it allows Horn-like rules to be combined with OWL DL ontologies.

SWRL rules have the form of an implication between an antecedent (body) and consequent (head). The intended meaning can be read as: whenever the conditions specified in the antecedent hold, then the conditions specified in the consequent must also hold. Both the antecedent (body) and consequent (head) consist of zero or more atoms. An empty antecedent is treated as trivially true (i.e. satisfied by every interpretation), so the consequent must also be satisfied by every interpretation; an empty consequent is treated as trivially false (i.e., not satisfied by any interpretation), so the antecedent must also not be satisfied by any interpretation. Multiple atoms are treated as a conjunction. Summarizing, SWRL rules can be described as follows:

antecedent $\rightarrow$ consequent, in which the antecedent and consequent consist of one or multiple atoms. Typical SWRL reasoning occurs on property and instance levels.

Protégé has a SWRL editor plug-in called SWRLTab. SWRLTab is a very convenient tool for editing SWRL rules since it supports automatic completion of the properties and class names and checks the syntax of the entered rules. The execution of SWRL rules requires the availability of a rule engine. The rule engine can perform reasoning using a set of rules and a set of facts as input. Any new facts that are inferred are used as input to potentially fire more rules (in forward chaining).

We chose the Jess rule engine, because SWRLTab translates SWRL Rules in Jess axioms in order to be executed and saves back the inferred knowledge in the ontology. Jess is a rule engine and scripting environment written entirely in Sun's Java language by Ernest Friedman-Hill at Sandia National Laboratories in Livermore, CA. Using Jess, Java software can be built that has the capacity to "reason" using knowledge that is supplied in the form of declarative rules. Jess is small, light, and one of the fastest rule engines available. Its powerful scripting language gives access to all of Java's APIs.

We also used the SWRL Rule Engine Bridge, a subcomponent of the SWRLTab that provides a bridge between an OWL model with SWRL rules and a rule engine. Its goal is to provide the infrastructure necessary to incorporate rule engines into Protege-OWL to execute SWRL rules.

9.1.3 Query Language SQWRL

SQWRL (Semantic Query-Enhanced Web Rule Language) [29] is a SWRL-based [22] language for querying OWL ontologies. It provides SQL-like operations to retrieve knowledge from OWL.

SQWRL takes a standard SWRL rule antecedent and effectively treats it as a pattern specification for a query. It replaces the rule consequent with a retrieval specification. SQWRL uses SWRL's built-in facility as an extension point.

Using built-ins, it defines a set of operators that that can be used to construct retrieval specifications. The attractiveness of this approach is that no syntactic extensions are required to SWRL. Thus, existing SWRL editors can be used to generate and edit SQWRL queries. In addition, standard SWRL serialization mechanisms can be used so queries can be stored in OWL ontologies.

The core SQWRL operator is sqwrl:select. For example, the following query retrieves all persons in an ontology with a known age that is less than 9, together with their ages:

$$\textit{Person(?p)} \wedge \textit{hasAge(?p, ?a)} \wedge \textit{swrlb:lessThan(?a, 9)} \rightarrow \textit{sqwrl:select(?p, ?a)}$$

Basic counting is also supported by SQWRL, provided by a built-in called sqwrl:count. A query to, say, count of the number of known persons in an ontology can be written:

$$\textit{Person(?p)} \rightarrow \textit{sqwrl:count(?p)}$$

An implementation of SQWRL has been developed in the SWRLTab plugin in Protégé-OWL. The implementation provides a graphical interface to edit and execute SQWRL queries and also provides a JDBC-like Java interface to execute SQWRL queries in Java applications.

9.2 Rules used in Demo

In this paragraph are presenting some of the rules that were used to the demonstration of the system in the smart space of Sandbox.

Rules responsible to activate the simple events.

```
   Core:Person(?u) ∧ hasRFID(?u, ?rfid) ∧ commnunicateWith(?rfid, ?rfidReader)
∧ Core:hasLocation(?rfidReader, ?loc) ∧ Core:currentStatus(?rfid, "on")
→ Core:hasLocation(?rfid, ?loc) ∧ Core:hasLocation(?u, ?loc)
```

```
   SmartClassroom(SmartClassroom1) ∧ studentsParticipants(SmartClassroom1,
?p) ∧ swrlb:equal(?p, 0) → isActivated(Students_Not_LocatedIn_Classroom,
true) ∧ isActivated(Students_LocatedIn_Classroom, false)
```

```
   Person:AcademicStaff(?a) ∧ Core:hasLocation(?a, SmartClassroom1) →
isActivated(Professor_LocatedIn_Classroom, true) ∧
isActivated(Professor_LocatedIn_Office, false) ∧
isActivated(Professor_LocatedIn_Hallway, false)
```

```
   Person:Student(?s) ∧ SmartClassroom(?c) ∧ Core:hasLocation(?s, ?c)
→ sqwrl:count(?s)
```

```
   SmartClassroom(SmartClassroom1) ∧ hasCalendar(SmartClassroom1, ?c)
∧ Calendar:hasCalendarEvent(?c, ?evnt) ∧ hasStartTime(?evnt, ?st) ∧
hasEndTime(?evnt, ?et) ∧ Time:hour(?st, ?t1) ∧ Time:hour(?et, ?t2) ∧
Time:hour(CurrentTime, ?t3) ∧ swrlb:greaterThanOrEqual(?t3, ?t1) ∧
swrlb:lessThan(?t3, ?t2) ∧
Calendar:description(?evnt, "Slide Presentation") ∧
hasLecture(?evnt, ?lect) →
isActivated(ClassroomCalendarEvent_LectureSlidePresentation, true) ∧
currentLecture(SmartClassroom1, ?lect)
```

```
   Projector(Projector1) ∧ Core:currentStatus(Projector1, "on") →
isActivated(Projector_is_On, true)
```

Assistance for "Lecture With Slide Presentation" Activity:

```
   Core:isIdentified(LectureWithSlidePresentation, true) ∧
Core:currentStatus(ClassroomBlinds, "off") →
Core:changedStatus(ClassroomBlinds, "on")
```

Core:isIdentified(LectureWithSlidePresentation, true) ∧ Core:currentStatus(ClassroomLights, "high") → Core:changedStatus(ClassroomLights, "low")

Core:isIdentified(LectureWithSlidePresentation, true) ∧ Core:currentStatus(ClassroomLights, "medium") → Core:changedStatus(ClassroomLights, "low")

Core:isIdentified(LectureWithSlidePresentation, true) ∧ Lecture:title(?lect, "relational algebra") ∧ Person:Student(?s) ∧ Person:hasMotivationState(?s, ?mot) ∧ Person:knowledgeDescription(?mot, "relational algebra") ∧ Person:knowledgeLevel(?mot, "Beginner") ∧ Person:hasContactProfile(?s, ?prof) ∧ Person:email(?prof, ?email) → sqwrl:select(?email)

Assistance for "Break" Activity:

Core:isIdentified(Break, true) ∧ Core:currentStatus(ClassroomBlinds, "on") → Core:changedStatus(ClassroomBlinds, "off")

Core:isIdentified(Break, true) ∧ Core:currentStatus(ClassroomLights, "low") → Core:changedStatus(ClassroomLights, "medium")

Core:isIdentified(Break, true) ∧ Core:currentStatus(ClassroomLights, "high") → Core:changedStatus(ClassroomLights, "medium")

Bibliography

[1] Resource description framework (RDF). model and syntax specification. Technical report, W3C, 2 1999.

[2] E. Aarts, Rick Harwig, and Martin Schuurmans. *Ambient Intelligence*, pages 235–250. McGraw-Hill, 2002.

[3] Grigoris Antoniou and Frank van Harmelen. *A Semantic Web Primer*. MIT Press, Cambridge, MA, 2. edition, 2008.

[4] Juan Carlos Augusto and Paul J. McCullagh. Ambient intelligence: Concepts and applications. *Comput. Sci. Inf. Syst.*, 4(1):1–27, 2007.

[5] Tim Bray, Jean Paoli, and C M Sperberg-McQueen. Extensible markup language (xml). *W3C Recommendation*, 2004(31-05):1–7, 2000.

[6] Dan Brickley and Ramanathan V. Guha. Rdf vocabulary description language 1.0: Rdf schema. *W3C Recommendation*, 10, 2004.

[7] Stefanidis C., Argyros A., Grammenos D., and Zabulis X. Pervasive computing @ ics-forth. In *Pervasive 2008 Workshop*, 2008.

[8] Harry Chen, Timothy W. Finin, Anupam Joshi, Lalana Kagal, Filip Perich, and Dipanjan Chakraborty. Intelligent agents meet the semantic web in smart spaces. *IEEE Internet Computing*, 8(6):69–79, 2004.

[9] Belkacem Chikhaoui, Yazid Benazzouz, and Bessam Abdulrazak. Towards a universal ontology for smart environments. In Gabriele Kotsis, David Taniar, Eric Pardede, and Ismail Khalil Ibrahim, editors, *iiWAS*, pages 80–87. ACM, 2009.

[10] Grammenos D., Zabulis X., Argyros A., and Stefanidis C. Ambient intelligence and smart environments. In *3rd European Conference on Ambient Intelligence (AMI 2009)*, 2009.

[11] Anind K. Dey. Understanding and using context. Technical report, Future Computing Environments Group, Georgia Institute of Technology, Atlanta, GA, USA 30332-0280, 2001.

[12] Vasileios Efthymiou. A real-time semantics-aware activity recognition system. Master's thesis, Univesity of Crete.

[13] Vasilis Efthymiou, Maria Koutraki, and Grigoris Antoniou. Real-time activity recognition and assistance in smart classrooms. *Advances in Distributed Computing and Artificial Intelligence Journal*, 1:9–22, 2012.

[14] Stanford Center for Biomedical Informatics Research. Protégé. http://protege.stanford.edu/index.html.

[15] FORTH-ICS. Ambient intelligence programme, 2005. http://www.ics.forth.gr/ami/.

[16] Aurona Gerber, Alta van der Merwe, and Andries Barnard. A functional semantic web architecture. *The Semantic Web: Research and Applications*, pages 273–287, 2008.

[17] Asunción Gómez-Pérez. Evaluation of ontologies. *Int. J. Intell. Syst.*, 16(3):391–409, 2001.

[18] Maria Golemati, Akrivi Katifori, Costas Vassilakis, George Lepouras, and Constantin Halatsis. Creating an ontology for the user profile: Method and applications. In *Proceedings of the 1st International Conference on Research Challenges in Information Science (RCIS 2007)*, pages 407–412, Ouarzazate, Morocco, 2007.

[19] Tao Gu, Xiao Hang Wang, Hung Keng Pung, and Da Qing Zhang. An ontology-based context model in intelligent environments. In *In Proceedings Of Communication Networks And Distibuded Systems Modeling And Simulation Conference*, pages 270–275, 2004.

[20] Nicola Guarino and Christopher Welty. Evaluating ontological decisions with OntoClean. *Communications of the ACM*, 45(2):61–65, 2010.

[21] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11(1):10–18, 2009.

[22] Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosofand, and Mike Dean. SWRL: A semantic web rule language combining OWL and RuleML. W3C Member Submission, May 2004. Last access on Dez 2008 at: http://www.w3.org/Submission/SWRL/.

[23] Mohammad Rezwanul Huq, Nguyen Thi Thanh Tuyen, Young-Koo Lee, Byeong-Soo Jeong, and Sungyoung Lee. Modeling an ontology for managing contexts in smart meeting space. In Hamid R. Arabnia, editor, *SWWS*, pages 96–102. CSREA Press, 2007.

[24] Akrivi Katifori, Maria Golemati, Costas Vassilakis, George Lepouras, and Constantin Halatsis. Creating an ontology for the user profile: Method and applications. In Colette Rolland, Oscar Pastor, and Jean-Louis Cavarero, editors, *RCIS*, pages 407–412, 2007.

[25] Maria Koutraki, Vasilis Efthymiou, and Grigoris Antoniou. S-creta: Smart classroom real-time assistance. In Paulo Novais, Kasper Hallenborg, Dante I. Tapia, and Juan M. Corchado Rodríguez, editors, *ISAmI*, volume 153 of *Advances in Intelligent and Soft Computing*, pages 67–74. Springer, 2012.

[26] George Margetis, Asterios Leonidis, Margherita Antona, and Constantine Stephanidis. Towards ambient intelligence in the classroom. In Constantine Stephanidis, editor, *HCI (8)*, volume 6768 of *Lecture Notes in Computer Science*, pages 577–586. Springer, 2011.

[27] Deborah L McGuinness and Frank Van Harmelen. OWL Web Ontology Language overview. *W3C Recommendation*, 10:1–19, 2004.

[28] Hong Myoung-woo and Cho Dae-jea. Ontology context model for context-aware learning service in ubiquitous learning environments. *International Journal of Computers*, 2:193–200, 2008.

[29] Martin J. O'Connor and Amar K. Das. Sqwrl: A query language for owl. In Rinke Hoekstra and Peter F. Patel-Schneider, editors, *OWLED*, volume 529 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.

[30] Ciaran O'Driscoll, Mohan Mithileash, Fred Mtenzi, and Bing Wu. Deploying a context aware smart classroom. *Education and Development Conference (INTED)*, 2008.

[31] Davar Pishva and G. G. D. Nishantha. Smart classrooms for distance education and their adoption to multiple classroom architecture. *JNW*, 3(5):54–64, 2008.

[32] Bill N. Schilit, Norman Adams, and Roy Want. Context-aware computing applications. In *In Proceedings of the Workshop on Mobile Computing Systems and Applications*, pages 85–90, 1994.

[33] B.N. Schilit and M.M. Theimer. Disseminating active map information to mobile hosts. *Network, IEEE*, 8(5):22 –32, sep/oct 1994.

[34] Albrecht Schmidt. *Ubiquitous computing - computing in context*. PhD thesis, The University of Lancaster, 2003. http://d-nb.info/969858205.

[35] semanticweb.org. Semantic web. 2012. [Online; accessed January-2012].

[36] Thomas Strang, Claudia Linnhoff-Popien, and Korbinian Frank. Cool: A context ontology language to enable contextual interoperability. In *Distributed Applications and Interoperable Systems*, volume 2893 of *Lecture Notes in Computer Science*, pages 236–247. Springer Berlin / Heidelberg, 2003.

[37] Thomas Strang and Claudia L. Popien. A context modeling survey. In *UbiComp 1st International Workshop on Advanced Context Modelling, Reasoning and Management*, pages 31–41, Nottingham, September 2004.

[38] Norbert Streitz, Achilles Kameas, and Irene Mavrommati, editors. *The Disappearing Computer: Interaction Design, System Infrastructures and Applications for Smart Environments*, volume 4500 of *Lecture Notes in Computer Science*. Springer, Berlin, 2007.

[39] W3C. Time ontology in OWL, 2006. http://www.w3.org/TR/owl-time/.

[40] Xiao Hang Wang, Da Qing Zhang, Tao Gu, and Hung Keng Pung. Ontology based context modeling and reasoning using owl. *Pervasive Computing and Communications Workshops, IEEE International Conference on*, 0:18, 2004.

[41] W. Weber, J.M.Rabaey, and E. Aarts. *Ambient Intelligence*. Springer-Verlag Berlin Heidelberg, 2005.

[42] Wikipedia. Upper ontology (information science) - wikipedia, the free encyclopedia. 2011. [Online; accessed February-2011].

[43] Shi Yuanchun, Xie Weikai, Xu Guangyou, Shi Runting, Chen Enyi, Mao Yanhua, and Liu Fang. The smart classroom: Merging technologies for seamless tele-education. *IEEE CS*, 2003.