POLITECNICO DI TORINO

Master degree in Computer Engineering

Master Degree Thesis

Context Aware and Device Dependent Interaction in Smart Environments

Supervisors:
Prof. Fulvio Corno
Dott. Dario Bonino
Dott. Luigi De Russis

Candidate:
Emanuele Furci

Academic year 2013-2014

Abstract

As the number of ubiquitous and personal devices grows, new means of interaction between users and smart environments may become available and the integration of such devices in a smart environment could lead to a more natural and sophisticated Human-Computer interaction. This thesis aims at designing new interaction paradigms for smart environments with a particular focus on environment-to-user communication. Messages generated by a smart environment, either by means of inner processes (e.g., detection of some internal status like a door opening) or by bridging external services (e.g., e-mail, chat, etc.), have to be routed to the right recipient, on the most suitable device, depending on the current user location, activity and surrounding context (e.g., noisy vs quiet, dim light vs bright illumination, etc.). The main focus of the work will involve two main goals:

- the creation of a suitable modelling infrastructure for end-user device capabilities, user activities, current context in a home setting and type of messages generated by the smart home.
- the creation of a software infrastructure able to deliver messages to selected end-user devices, and to capture context information from the user surroundings (by exploiting existing systems, e.g., Dog), by accounting for user preferences and possibly negotiating with the user the most suitable interaction pattern to adopt.

To fulfill the previous goals, several simplifying assumptions have been applied to concentrate on the model definition and to evaluate its behaviour in a test case by means of the developed software:

- users' location and activity is assumed to be known
- users' identity is assumed to be known
- end-user devices may only be in a pre-defined set (e.g., mobile phone, smart watch, audio diffusion, tv, computer monitor, in-home display)

The information has been modelled by means of an ontology, Notont. As better discussed in the section 3.2, an ontology is a formal description of concepts in a domain of discourse, properties of each concept describing various features and attributes of the concepts. Such concepts are described by means of classes. Notont is able to represent information about the users, the devices, the house with its appliances and the type of messages to deliver. For the users, we are able to model his current activity, location, obtrusiveness level, as well as his interaction with the house appliances and mobile devices. For the devices, we model their capabilities

such as GPS positioning and their physical features such as screen size. For the house we model the architectural elements such as the rooms, its plants and facilities with their specific functionalities. For the messages we specify their type and their priority. By using Notont it is possible to have a formal description of the data we are interested in. The needed information is modelled as instances of the the proper class and, once that the model is ready, the software starts managing all the incoming messages from the house and, by taking into account user, environmental and devices state, chooses the proper device to send the message to.

A complete scenario has been developed in order to test the capability of the Notont ontology to describe real conditions and to test the overall system performances.

Contents

List of Figures

Chapter 1

Introduction

1.1 Overview

Smart Environment (SmE) is a research field that embraces a variety of disciplines, including artificial intelligence, pervasive and mobile computing, robotics, middleware and agent-based software, sensor networks, and multimedia computing.

A Smart Environment is able to acquire and apply knowledge about the environment and its inhabitants in order to improve their experience in that environment [8].

The basic elements of a SmE are:

- user applications that provide services, statistical information and interface with the users for the sending of commands to devices;
- middleware software (or gateway) that manages the communication between application and devices;
- devices such as sensors or actuators that gather information from the environment and control the plants;
- environment that can range from single rooms to houses and buildings, but also to hospitals, schools, underground stations or other public areas.

The automation process in a smart environment (figure 1.1) can be viewed as a cycle of: perceiving the state of the environment, reasoning about the state together with task goals and outcomes of possible actions, and acting upon the environment to change the state.

Perception of the environment is a bottom-up process. Sensors monitor the environment and make information available to the applications through the middleware layer. The action execution, instead, flows top-down. The decision actions defined by the applications are communicated to the devices through the middleware. The physical layer performs the action with the help of actuators or device controllers, thus changing the state of the environment and triggering a new perception. Since the user interacts either with the environment and the application, he/she stands in the middle of this cycle.

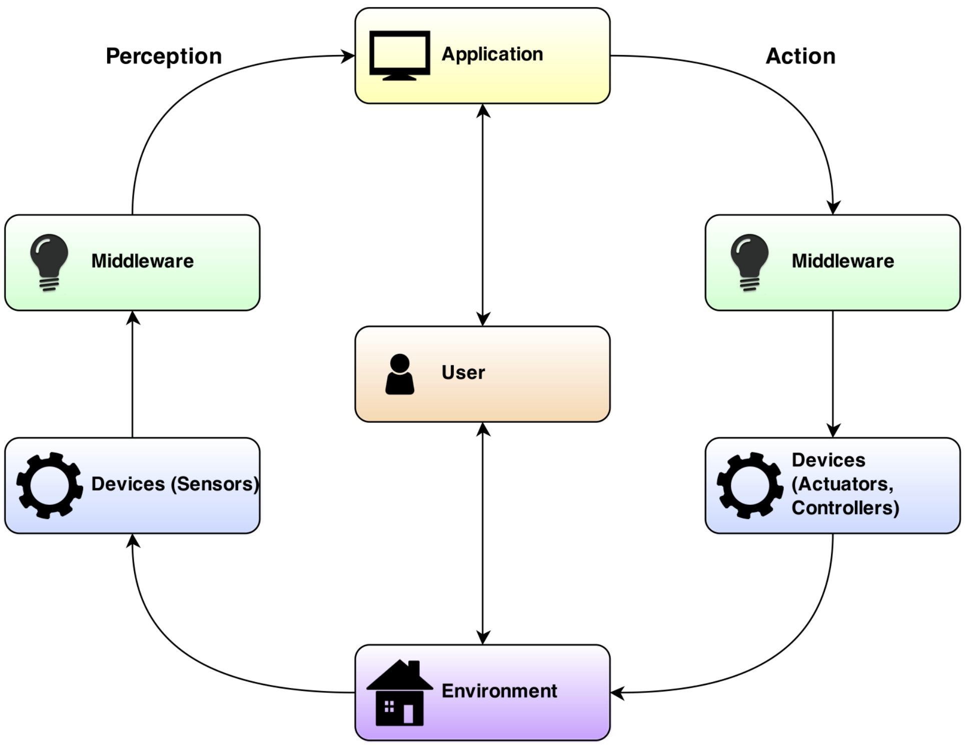


Figure 1.1. SmE Perception/Action Cycle

In a smart environment, thus, an application is able to send commands to plants (by means of the middleware) to performs automated actions, enriching the user experience and increasing the overall comfort. To properly operate, these applications require information about the house, its plants, the user and its preferences; this information need to be always updated by sensors placed all over the house so that the decisions are made on the base of the proper data.

Such information is usually referred as “context”. Over the years, each system modelled the context by means of ad-hoc models, such as XML models, tables, Key-Value pairs, or other data structures. Today, another possible approach is given by the ontology-based modelling [1].

1.2 Motivation

A typical example of smart environment is a smart home where a home automation system, together with an intelligent software, works to provide a more comfortable and usable home to live in.

A smart home, opens a new way of living the home, with a more sophisticated and intelligent management of its plants, an enhanced comfortable environment and an enriched communication between the householders and the house. Today, several communication channels are available, ranging from integrated audio and video systems, touch panels, to more modern smart devices like smartphones, tablets and smartwatches.

The usage of smart devices is often oriented to the user-to-system communication: by means of mobile or web applications, the user is able to remotely control the house and to monitor it retrieving the wanted information. Smart homes make available an increasing amount information such as plants state, real-time environmental conditions, appliances notifications; the multitude of ways to communicate this information with the householders, open new challenges for the creation of new interaction paradigms in smart homes. The house is becoming a more interactive environment, able to communicate its states and to inform on relevant events, in a natural manner. This increased communication capability has to face aspects regarding the amount and the type of information to provide, as well as the way it is provided and the availability of the user in receiving such information. New questions raises, like: how can a home automation system route a particular message to the proper user and the proper user device by taking into account the current context? Which type of interaction can be more effective for the communication between the user and the house? To which set of messages is the user interested in?

We tried to give an answer to these and more questions by working on what we call the Notont Project. T

The aim is to enrich the communication between a smart environment and the users providing a personalized notification system able to take into account the current user state, the current location, activity and obtrusiveness level to deliver the messages. User's devices, described by their features and capability, their usage and state are considered as well.

The knowledge of the current activity, let us infer the user accessibility level, that is his availability in interacting with a device while performing an activity. For

[1]The definition on ontology is given in the Third Chapter. For the moment we can say that is a means for the modelling of generic concepts

instance a user that reads a book can still use its hands for the interaction with a device, while a user that is having a shower can't. The knowledge of the user location, together with the knowledge of the appliances availability for each place (like rooms, garden or others), may provide a wider device choice for the message delivery. By knowing the appliances functionalities, indeed, the system is able to match the most suitable channel to use among all the available ones.

Messages generated by the smart home, are classified under several categories and users can choose the ones they are interested in, so that unnecessary messages can be filtered out. Such messages, are routed to the right recipient, on the most suitable device, depending on the actual context.

Due to the complexity of the intended work and to the limited resources available for it, some simplifications have been made:

- since the available resources don't let us to infer the user activity, it will be provided to the system as an external a priori known input;
- user location and user identity is assumed to be known as well;
- although the potentially usable devices are quite high, we focused only on Android-based devices.

The software has been tested on a sample house environment; for the purpose of this work, the e-Lite[2] lab has been used. Here, a home automation system has been recreated by the e-Lite team, ranging from sensors, to buttons and actuators. The Notont software uses information gathered from the home automation system to generate messages. Such messages are routed to the user devices by using information about users, devices and house plants to infer the end user device to send the message to.

1.3 Organization

This thesis is organized as follows:

on the **Second Chapter**, the state of the art of the home automation systems is presented and an overview of the main available technologies is given.

On the **Third Chapter**, we give a description of the context in a Smart Environment, introducing ontologies as a possible way for context modelling and their usage in ontology-based application such as Dog, a gateway for the management of smart environments developed by the e-Lite group.

[2]http://elite.polito.it

On the **Fourth Chapter**, we present the Notont ontology as a proposed ontology for context modelling, focusing on its main aspects and capabilities.

On the **Fifth Chapter**, we present the software for data model management with the description of its architecture and features..

On the **Sixth Chapter**, we present the results about the software usage. By using it on a simulated home environment, we discuss about the results, strong and weak points of the project and provide hints for possible future works.

Chapter 2

Home Automation Systems: an overview of the current State of the Art

2.1 Introduction

The Home Automation term refers to the integration of Information and Communication Technology (ICT) in a home environment, to control, monitor and manage house appliances with the aim of increasing comfort, security and energy efficiency.

Since the '70s, home automation systems integrated several components in houses for facilities management but, with the introduction of microcontrollers, they started to be more widely used: the availability of new devices, able to perform more complex actions and the usage of computers to automatically manage the house plants, introduced intelligence in the house. However, for many years, due to the high prices of home automation systems, they have been confined to a limited set of people, who could afford their cost. Nowadays, home automation systems are becoming more popular than in previous years, since the increased interest of companies on this market makes available a wider devices choise, with a lower price.

2.2 Home and Building Automation Systems

Today, automation systems are applied both in houses and in buildings like offices or public structures; for this reason we can distinguish between home and building automation systems. A home automation system aim at simplifying the domestic activities, providing automatic plants management, remote control and implementing energy saving strategies. The system is more focused on the user than the house itself and makes available custom configurations like personalized behaviours and system preferences. A building automation system, instead, focuses more on the building than on the people: the aim of the system is to grant that building plants work properly and are correctly managed. Implemented functionalities can refer, for example, to the automatic light and temperature control, access control, security management, etc. Users are not allowed to configure or modify the system behaviour that is strictly controlled by specialized technicians. The correct building management grants a more comfortable environment for employees, high plants efficiency and energy saving. Although different, home and building automation can be seen as two approaches to the same automation problem: simplify the inhabitants' life and increase the comfort by automatically managing plants and facilities in a building.

An automation system relies on a set of components, with specific functions. They can be divided in:

- Sensors, able to measure physical quantities, providing a signal for the correspondent measure. They give information on the surrounding environment needed for a correct management of the house plants. Typical examples are: temperature sensors, smoke detectors and movement sensors.

- Actuators, able to perform physical actions like moving, lifting, pushing or pulling the objects they are connected to. They provide automated actions such as opening the garage door, turning on the lights and starting the irrigation system. They are usually made up of mechanical or electrical motors, pumps or relays.

- Controllers, usually made by microprocessor and built-in sensors, manage plants and provide the necessary signals to command them. Some example are: heating/cooling controllers that automatically start the HVAC (Heating, Ventilating and Air Conditioning) system, or the fan controller that can regulate the fan speed in order to maintain the CO_2 level in a optimal range.

- Central Unit, is the coordinator of all the devices placed in the network. It is responsible of the management and maintenance of the system. It can performs reconfiguration operations and keeps data about usage, errors, statistical reports and others useful information. The central unit is made up by a computational unit, like a personal computer but it is not always needed; smart controllers may communicate each other sharing the required information for devices management.

Devices are connected to a network which can be wired, wireless or hybrid (partially wired and partially wireless). In order to properly communicate, devices have to use the same protocol that defines all the aspect of communication, like: physical interfaces, connectors, electric levels, message formats, addressing mode, type of messages, ect. Since '70s, several bus standards and technologies have been developed from open to proprietary standards:

- KNX, is a merger of three former European bus standards: EIB (European Installation Bus), EHS (European Home Systems Protocol) and BatiBUS. It is typically used in building automation for controlling electrical loads, plugs, shutters and other house and building commodities. The standard defines the means of transmission to use, that are: twisted pairs (TP-0, TP-1), powerline (PL-110, PL-132), ethernet (KNXNet/IP) or radio (RF). Since it is a standard, devices can communicate each other even if produced by different companies.

- ModBus, is one of the most diffused industrial protocol used to control PLC (Programmable Logic Controller). It is an application level protocol, with a master/slave approach communicating with request/reply messages. It defines the type of messages and defines two operating mode for the message transmission: RTU over serial line (RS-232 or RS-485) and TCP over Ethernet-like line.

- RS-485, the serial standard, typically used to connect measuring devices, since it is differential (signals are transmitted as difference between the two wires voltage), it resists to electromagnetic interferences from motors and welding equipment. Although it isn't a complete communication protocol (it just define the electrical requirements), it is one of the oldest protocols used in home automation field.

- LON (Local Operating Network), is a proprietary standard, used for energy-control, steering machinery and access control systems in industry and larger buildings. The standard is mostly known for power line signalling, but also supports signal cables (twisted pair), coaxial cables, radio and fibre optical transmission.

- X10, is a standard for powerline signalling, widely used for management of domestic electrical commodities, like lamps and radiators. It is also used in environmental control systems in single houses.

- Insteon, is another standard operating on power lines or radio frequencies that allows the management of house devices such as switches, sensors and lights. It is compatible with the X10 protocols and users can migrate from X10 to Insteon.

- BACnet, is a standard developed in the USA for the control of functions in large buildings. It defines a number of services used by devices to communicate each other. The protocol services include Who-Is, I-Am, Who-Has, I-Have, which are used for Device and Object discovery. Services such as Read-Property and Write-Property are used for data sharing. The protocol defines a number of Objects that are acted upon by the services. The objects include analog input, analog output, analog value, binary input, ect. It has not widely used in European smart homes.

- DALI (Digital Addressable Lighting Interface), is a protocol designed to perform advanced lighting control strategies. It let monitor and control lighting devices, granting the desired amount of light in a uniform way, and properly mixing the natural and artificial ones. It's typically transmitted on wires, even if a wireless extension is available.

- MyOpen, is a proprietary standard developed by an italian company and used mainly in Italy and south America. This protocol is applied to all the company components and the interoperability between others devices is possible by means of specific gateways.

- EnOcean, is a wireless standard that uses the principle of energy harvesting to retrieve the required energy for devices. Thus, devices don't need a battery but specific energy converters to obtain the needed energy. Devices can be easily installed since no wire is needed, and the transmission range vary from 30 m inside buildings to 300 m for open spaces. The transmission frequencies used are 902 MHz, 928.35 MHz, 868.3 MHz and 315 MHz.

- ZigBee, is a wireless protocol based on the standard 802.15.4 and developed by the ZigBee Alliance. The ZigBee protocol lets control all house commodities with signal on the band of 2.4 Ghz. Battery supplied devices can be easily installed almost in every place and house facilities like oven and fridges are starting to integrate this protocol. Devices are connected in a mesh network where each component can receive and forward messages from and to others devices in order to achieve the message delivery to the proper recipient. Battery supplied devices are not used in the routing process, since they switch in sleep-mode for energy saving. Although all devices provided by companies complying with the ZigBee alliance should communicate each other in a transparent way, it it's not always possible, since the application level may be implemented in a different way.

- ZWave, is a wireless protocol operating in the sub-gigahertz frequency range, around 900 MHz. Several devices are available from the companies in the

ZWave Alliance, that grants the intercommunication between them in a transparent way. Devices can communicate each other within the range of 30 m in open space and are connected in a mesh network where not battery supplied devices provide the messages forwarding from a point to another. Battery supplied devices, due to the sleep mode, are not taken into account in the routing process.

- Bluetooth, is entering as a home automation protocol as well, since newer devices are becoming available in the recent years. Although it has born for a different purpose, Bluetooth standard can be used to connect home automation commodities. It works on the band of 2.4GHz with a master/slave approach. The master can communicate with up to 7 slaves in a piconet.

For inter-protocol communication, the signal translation is required and dedicated gateways are used for this purpose. Ethernet and Wi-Fi technologies are usually used to create backbone between subnetworks, in order to extend the home network and/or provide access from the outside;

Several means of communication are available for the inhabitants to interact with the system, providing commands, monitoring the house, configuring strategies and behaviours. A system can integrate more than one:

- standard plugs: classical switches and buttons are used to control the house facilities like the lighting, shutter, blinds. They can integrate some display and are usually connected to actuators that perform the intended action;

- control panel: they are made of screen placed on a wall, providing information about the house. Commands can be sent via a set of buttons placed in the panel, for the older ones, or via the touch screen;

- SMS: is the oldest way for the remote controlling of the house. Commands are sent via sms, as a codified text, that is interpreted by a gateway and a signal is generated for the intended operation;

- web access: home automation systems usually grant a remote access available through the web via a webserver. The system can be easily monitored and controlled from inside and outside the house by simply connecting to the home gateway URL via a browser;

- smartphone and tablet: as well as web access, for modern systems there are specific mobile applications to remotely access to the house system.

2.3 Smart Homes

In home automation systems, the integration of a central unit brings some intelligence in the management of the house plants and commodities. For this reason, houses with an advanced automation system are often called smart homes or smart houses. The term “smart home” was first coined by the American Association of Housebuilders in 1984, indicating a house with a system able to control and monitor it in a smartly way. Today the therm “smart home” is a way to call a house with a home automation system in which the component integration is not only related to the house plants but, more importantly, with smart things. Smart appliances like oven and fridge, home entertainment systems, security systems like video surveillance, can be connected under a unique management system able to coordinate them to satisfy all the desired user preferences. Beyond the simple configuration and activation of a scenario, where a set of components are commanded to achieve the wanted state (for example the scenario tv-mode could turn off the lights in a room, turn on the projector and the audio system, lower the screen and the blinds), a smart home is able to perform actions in an autonomous way, by taking into account the actual context[1] to perform the intended actions in the best way. Thus, for example, the washing machine can be activated once that the power consumption of the house is under a defined threshold and the energy cost is lower, or the oven can be turned on to warm the dinner, when the user comes back home. In a mobile-phone conversation, the call could be passed on the audio system and, if necessary, recorded on some local system. Smart homes are opening a new way of living home: the central unit can perform advanced control strategies in order to increase the overall comfort and the integration of plants for the exploitation of renewable energy sources can optimize the energy consumption. The management system, by means of monitoring processes, makes available real time information about the house and its plants state, as well as historical data and statistical reports. This large amount of information can be used by the householders to better manage and configure the system and to be aware of the house consumptions. An effort in the direction of appliances integration in a smart home is made by the Energy@Home[2]. The Energy@Home project, in collaboration with the ZigBee Alliance, is working for the creation of a management system able to interact with smart appliances to create a smart environment. The system can automatically schedule appliances usage, accordingly to the energy availability, maximizing energy plants efficiency

[1] with the word context, is intended a set of information from sensors or other sources, able to describe the actual state of the house, its plants and its inhabitants. A better description is presented in the Third Chapter

[2]www.energy-home.it

and energy saving. The framework is called JEMMA and is open-source (LGPL). It implements the Energy@home specifications for energy monitoring and application management.

Learning systems can adapt their behaviour accordingly with the user preferences, evolving over time to better fit the user needs. An example of such technology is given by Nest[3]. The Nest project has developed a smart thermostat able to learn on user habits to maintain the optimal environmental temperature (figure 2.1). Although the project is evolving and had a great success, there are still some weak points related to the learning process and the ability of the thermostat to effectively lead to energy saving [31].

Figure 2.1. Nest Smart Thermostat

A smart home is able to provide useful information coming from the Web: a weather forecast service can suggest the right dress to put on, a connection to the public transport service can provide all necessary information on the nearest bus or train stop. Smart homes offer the great possibility to adapt the house to users needs, for example: by mixing weather forecast and automated irrigation system, the irrigation process can be regulated, stopped or delayed accordingly to the incoming weather. Personal devices, such as fall and pulse detectors, can monitor elderly people health state and trigger an alarm if something goes wrong [21]. A monitoring system can check ill people health state, letting them communicate with a doctor who can ensure that everything is fine. This information can be used for detailed analisis [17]. Smart homes can make life more easy for people with disabilities or

[3]www.nest.com

limited capabilities ensuring a better quality of life: an example is given by the DOGeye (figure 2.2), a multimodal eye-based application for the management and the control of a smart home, based on state-of-the-art technologies in both eye tracking systems and smart environment. It enables people to control their homes through different input devices, possibly combined, so that it does not limit itself to eye tracking only. The presence of various input modalities allows the application to be used by other people present in the house and offers different alternatives to the persons affected by severe and possibly evolving impairments, such as ALS (Amyothrophic Lateral Sclerosis) [9].

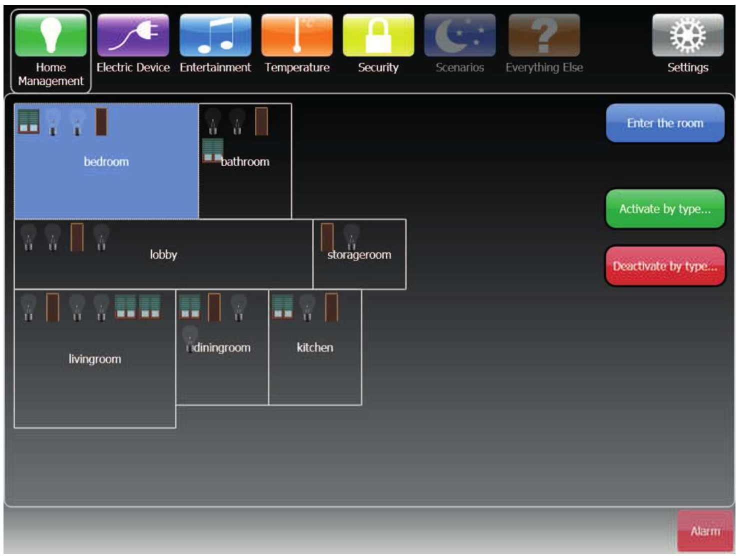
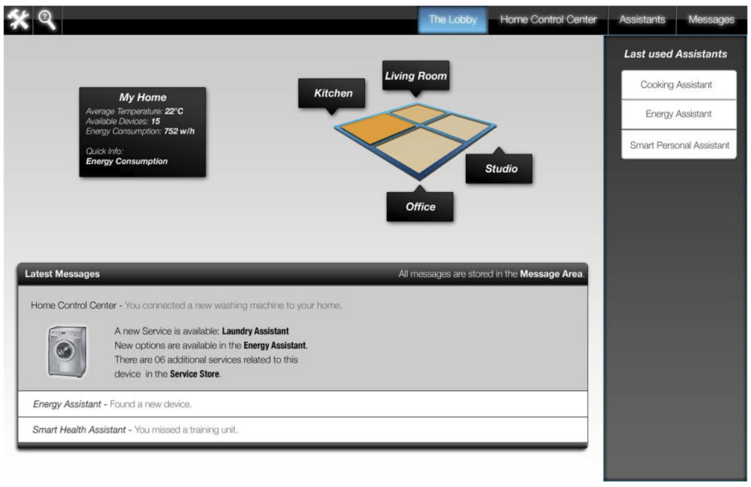


Figure 2.2. DOGeye User Interface

Researches in the field of smart environment and, in particular, of smart homes, lead several smart devices to be available. The smart sofa at Trinity College, for example, contains programmable sensors on the couch legs that identifies the individual sitting on the couch based on their weight distribution. The couch can thus greet the individual and could forseeably customize the immediate surroundings for that person. A number of intelligent and networked kitchen appliances have been designed by companies such as GE and Whirlpool that add multimedia interfaces and status reporting capabilities to the kitchen. The 200ConnectIo device refrigerates food until commanded to cook it by phone, computer, or personal digital assistant (PDA). The MIT Things That Think group has developed intelligent devices such as smart hotpads that determine whether a pan is too hot to touch, a spoon that provides feedback about the temperature and viscosity of food, and a kettle that says how much longer you have to wait for tea. The Philips interactive tablecloth weaves a power circuit into a washable linen tablecloth, so that devices can be charged when they are placed anywhere on the tablecloth [8].

Beside researches about smart devices, a big effort is being done to develop software and operating systems for smart homes. The trends is, indeed, the abstraction of the house as a PC where devices are seen as peripheral connected to a main operating system that takes care of their management and user applications provides services to users. An example is given by the Home Operating System (Home OS), a multimodal interface proposed by the Technical University of Berlin. Home OS allows users to control their homes using touch, speech and gesture interactions. It provides a simple, clear and reduced GUI for the smart home management, allowing users to easily interact with it, even while performing other activities [30]. The figure 2.3 shows the Home Os GUI.



My Home
Average Temperature: **22°C**
Available Devices: **15**
Energy Consumption: **752 w/h**

Quick Info:
Energy Consumption

Latest Messages All messages are stored in the **Message Area**.

Home Control Center - You connected a new washing machine to your home.

A new Service is available: **Laundry Assistant**
New options are available in the **Energy Assistant**.
There are 06 additional services related to this device in the **Service Store**.

Energy Assistant - Found a new device.

Smart Health Assistant - You missed a training unit.

Figure 2.3. Home Os GUI

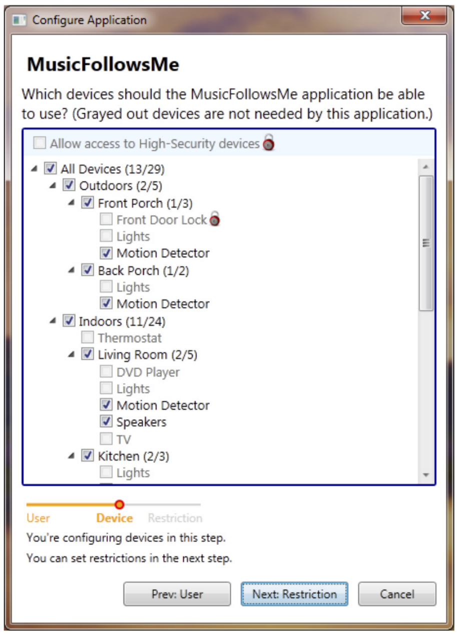
Another contribute to this vision is given by Microsoft with its HomeOS operating system [11]. The system is modular and made by three main layers:

- Device Connectivity Layer (DCL): solves the problems of discovering and associating with devices. This includes dealing with issues arising from protocols designed to operate only on one subnet as well as connecting to devices with multiple connectivity paradigms. The DCL provides higher layers with handles for exchanging messages with devices, avoiding any understanding of device semantics. There is one software module in the DCL for each protocol (e.g., DLNA and Z-Wave). This module is also responsible for device discovery, using protocol- specific methods (e.g., UPnP probes). If it finds an unknown device, it passes that up to the management layer where the proper action can be taken.

- Device Functionality Layer (DFL): takes the handles provided by the DCL and turns them into APIs that developers can easily use. These APIs are services that are independent of device interoperability protocols and the DFL is architected to allow easy incorporation of new devices and interfaces whether they are similar to existing ones or not.

- Management Layer: provides two key functionalities. First, it provides a central place to add and remove applications, devices and users, as well as to specify access control policies. Second, it mediates potentially conflicting accesses to devices, ensuring that applications do not need to build their own mechanisms to handle shared devices.

Applications are available through the HomeStore on-line repository and users can install them to obtain new functionalities from installed devices. The research group is trying the operating system in real test houses where already developed applications provide services like "MusicFollowMe" (figure 2.4), an application that redirect the audio accordingly with the user movements, or the application to control lights by means of a Kinect (figure 2.5)


Configure Application

MusicFollowsMe

Which devices should the MusicFollowsMe application be able to use? (Grayed out devices are not needed by this application.)

- [ ] Allow access to High-Security devices
- [✓] All Devices (13/29)
  - [✓] Outdoors (2/5)
    - [✓] Front Porch (1/3)
      - [ ] Front Door Lock
      - [ ] Lights
      - [✓] Motion Detector
    - [✓] Back Porch (1/2)
      - [ ] Lights
      - [✓] Motion Detector
  - [✓] Indoors (11/24)
    - [ ] Thermostat
    - [✓] Living Room (2/5)
      - [ ] DVD Player
      - [ ] Lights
      - [✓] Motion Detector
      - [✓] Speakers
      - [ ] TV
    - [✓] Kitchen (2/3)
      - [ ] Lights

You're configuring devices in this step.
You can set restrictions in the next step.

Figure 2.4. HomeOs: MusicFollowMe application

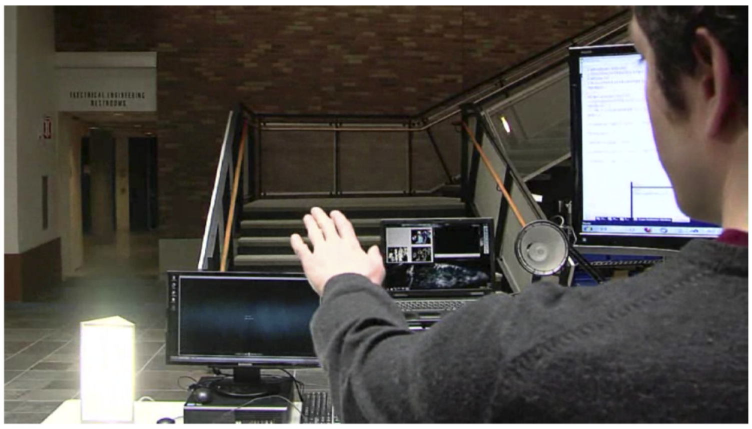


Figure 2.5. HomeOs: kinect-based light control application

2.3.1 Smart Homes: users expectations

As seen in the previous section, smart home technologies have evolved in a way that lead a lot of new services to be available to users. General users expectations about smart homes seem to be in general accord with such available technology although it can be saw a mismatch between the general trends that researches in this field are pursuing, and the real user expectations in terms of home functionality and features. In a survey conducted by the e-Lite team [3] subministered to unfiltered groups, with different sizes and cultural backgrounds, most of the required features from a smart home can be actually addressed with nowadays commercial technologies, either directly or by designing suitable integrations.

The survey "What would you ask to your home if it were intelligent?" aimed at understanding if some gap exists between advanced researches on smart environments and general public expectations and what are the causes.

In the great majority of answers, users show a strong attention to real, tangible needs, while most charming research topics such as information mobility, integration and sharing only play a marginal role. Main of the users expectations, indeed, regards the automatic plants managements such as: "Lighten the kitchen lights at a given hour", "Automatically turn off the lights after a certain time if no one is in the room" or "Switch on the heating system one hour before I'm back home". All these requested features are actually available with one single technology or with a combination of more ones.

Some interesting user expectations, revealed in the survey, require a major effort to be achieved and, for some of them, other researches have to be conducted. Such requests are, for instance: "Clean yourself, please", "Give me an alert when food in the cupboard is near its Expiration Date", "Wash, iron and order clothes in their correct places, without forgetting the moth repeller", or "Prepare my lunches and dinners according to my daily preferences, please". Such user expectations embrace a variety of research field like robotics, home automation, or computer intelligence.

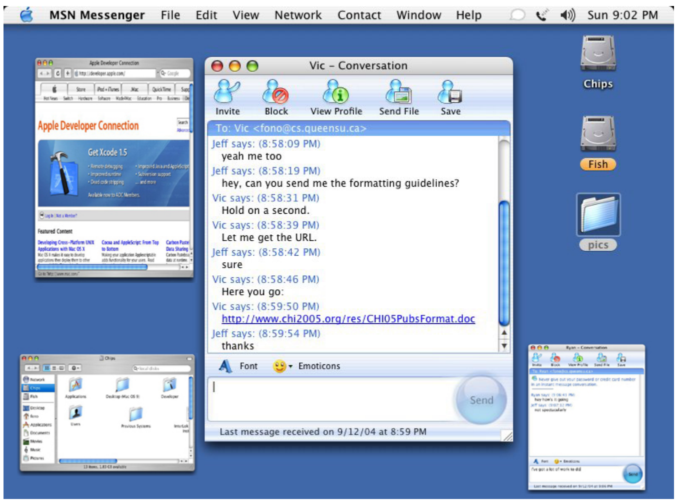
What emerge from the survey is that a greater attention must be given to HHI (Human Home Interaction) design, involving more and more users in the design of their own future homes. Research should better take into account and integrate with existing commercial solutions, and contribute to dissemination and information about their capabilities. Emerging "lightweight automation" solutions based on ZigBee, ZWave and other wireless communication technologies can provide a great contribution to the creation of more affordable intelligent homes, easy to shape around specific user needs.

2.4 Toward Attentive User Systems

In home automation systems, several means of interaction are available. Human-Computer-Interfaces, have evolved in last 35 years from simple text based interface to graphical, voice and gesture interfaces [24].

The way an user can interact with a system and the viceversa is evolving with the aim of reaching an interaction that is as natural as possible.

In recent years, some researches are focusing their work on what is called attentive user interfaces (AUI). An attentive user interface takes care of the current attention state of the user and adapts the way the information is sent to him, regarding its current activity [27]. The idea is to gather the user attention in the same way a person takes the turn in a conversation: in that case, the user who is trying to speak, sends some visual, sound or gestural signs in order to obtain the other person attention, without interrupting him [28]. In an interactive system, the main problem is how to make possible for a computer to understand what the user is paying its attention to and how to interrupt him. Some solutions uses cameras to capture the user state of attention and analyse the eye gaze to determine where the user is paying its attention to. An example is given by the eyeWindows [12] application (figure 2.6), where the opened windows of several applications are automatically resized accordingly to the attention the user is giving them. Another solution is given by Suitor, an attentive information system [19]. By monitoring the user input on the keyboard and by tracking its eye gaze, Suitor tries to understand what the user is doing to provide contextualized hints. In a place like a smart home, we can find what we need to create an AUI: several sensors to reveal the user state, appliances and other output devices to signal the intention to receive attention, input devices like touch panels, or smartphones to receive commands. Some interesting examples are: the eyeAppliances [25], smart appliances that, thanks to sensors, are able to gather the user attention to activate a vocal command interface. The appliances are managed by a reasoner that handles the interaction, called eyeReasoner [28]. An



To: Vic <fono@cs.queensu.ca>
Jeff says: (8:58:09 PM)
yeah me too
Jeff says: (8:58:19 PM)
hey, can you send me the formatting guidelines?
Vic says: (8:58:31 PM)
Hold on a second.
Vic says: (8:58:39 PM)
Let me get the URL.
Jeff says: (8:58:42 PM)
sure
Vic says: (8:58:46 PM)
Here you go:
Vic says: (8:59:50 PM)
http://www.chi2005.org/res/CHI05PubsFormat.doc
Jeff says: (8:59:54 PM)
thanks

Last message received on 9/12/04 at 8:59 PM

Figure 2.6. eyeWindows: the user concentrates on a single focus window with an active conversation. Other windows are distorted but remain visible

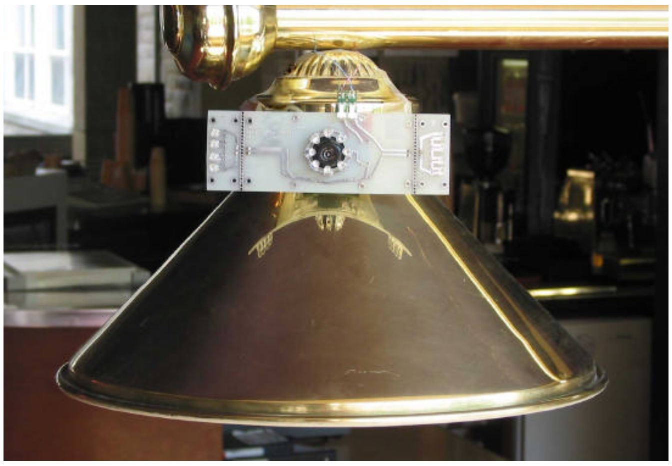


Figure 2.7. eyeLamp: light fixture with eye contact sensor

eyeAppliance is a lamp (figure 2.7) that can be turned on or off by watching at it and saying “on” or “off”.

Researches on this area may lead in the future to new way of interaction with a smart home. Although the potentiality of the implemented solutions, today they haven’t had a great success yet and smart homes still maintain the classical approach of a GUI (graphical user interface) both available on a larger number of devices such as smartphone, tablet and smartwatch for the user to computer interaction and for the computer to human interaction.

Chapter 3

Context and Ontologies as a modelling system

3.1 Introduction

Smart Environment (SmE) is growing as a multi-disciplinary field which allows many areas of research to have a significant beneficial influence into our society. By enriching an environment with technology, a system can be built to take decisions based on real-time information and historical data accumulated in order to benefit the users of that environment. Networks, Sensors, Human Computer Interaction (HCI), Ubiquitous Computing and Artificial Intelligence (AI) are all relevant and inter-related research domains with SmE. In a Smart Environment all these resources provide flexible and intelligent services to users acting in their environments, but the modelling and the management of the needed data is required.

3.2 Smart Environments and Context

To provide services that help inhabitants in their daily life, such as the automatic light control or the quality of air monitor, required information such as user presence or CO_2 level need to be gathered by the network sensors; more information may lead to more services for the users. Information can be divided by the "5Ws" (Who, Where, What, When and Why) design principle [1]:

- Who: the identification of a user of the system and the role that such user plays within the system in relation to the others. This can be extended to identify important elements like 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 and the ones that may work well indoor may be useless outdoor and vice-versa.

- When: the association of activities with time is required to build a realistic picture of a system dynamics. Users living in a house will change location and knowing when those changes happened, and for how long they lasted, is fundamental to understand how an environment is evolving.

- What: the recognition of activities and tasks users are performing is fundamental in order to provide appropriate actions, if required. The multiplicity of possible scenarios that can follow an action makes very difficult the anticipation of the user needs. 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. An important aspect of SmE has to do with interaction: on one side there is a motivation to reduce the human-computer interaction as the system is supposed to use its intelligence to infer situations and user needs from the recorded activities, as if a passive human assistant was observing activities unfold with the expectation to help when (and only if) required. On the other hand, a diversity of users may need, or voluntarily seek, direct interaction with the system to indicate preferences and needs. Today, with so many gadgets incorporating computing power of some sort, HCI continues to thrive as an important area of study [5].

The gathering of information about users, their activity and location, devices and their interaction with users, builds the context which is a fundamental part for a smart environment. As asserted in [10], context is any information that can be used to characterize the situation of entities (i.e. whether a person, place or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves.

Without context information, a SmE couldn't bind a service to the proper user and couldn't help him in his daily activities. Today, the process of acquiring information about users, environment and devices located in such environment is an open challenge for the SmE field. Nowadays, for the environment, a sensor network can provide enough information about the place where a user lives and smart devices are able to notify or be queried about their states. The main challenge regards the acquiring of reliable information about the user, its current activity, location and intended actions; such information would help the system in providing supporting services to the user. For example a system could low down the music and the light intensity to a proper level once it recognizes that an user felt asleep on a couch.

Activity recognition in a smart environment presents several challenges. Activities can be carried out with a high degree of freedom in relation to the way and the sequential order they are performed. Individuals have different life-styles, habits, or abilities and, as such, have their own way of performing activities. Activities usually follow some kind of pattern, but there are no strict constraints on the sequence and duration of the actions. For example, to prepare a meal one can first turn on the cooker and then place a saucepan on the cooker, or vice versa. Moreover, same phenomena usually happen while performing different activities. The wide range of activities and the variability and flexibility in the manner in which they can be performed increase the complexity of the activity modelling and recognition. In a Smart Environment, multimodal sensors generate heterogeneous data different in both formats and semantics. It is often necessary to fuse and interpret sensor data from multiple sources in order to establish the context of the ongoing activity. In addition, sensor data are full of noises (e.g., missed activations and/or faulty readings) and this increases their uncertainty and the reliability of recognition. Most activities are composed of a sequence of temporally related actions, so, sensor data related to an activity are generated incrementally as the activity unfolds. Current researches on activity recognition have mainly focused on the use of probabilistic and statistical analysis methods, the so-called data-driven approach, for single-user and single-activity scenarios. Activity recognition approaches can be generally classified into two categories. The first is based on the use of visual sensing facilities, e.g., camera-based surveillance systems, to monitor an actor's behavior and environmental changes. The approaches in this category exploit computer vision techniques to analyze visual observations for pattern recognition. The second category is based on the use of emerging sensor network technologies for activity monitoring. The sensor data are analyzed using data mining and machine learning techniques to build activity models, which are then used as the basis of activity recognition. In these approaches, sensors can be attached to an actor through wearable sensors, or to objects. Wearable sensors often use inertial measurement units and RFID tags to gather an actor's behavioral information. To build a complete description of the input data for the activity modelling, probabilistic analysis methods such as Markov models and Bayesian networks are used. These methods incorporate inhabitant's preferences by tuning the initial values of the parameters of the probabilistic models. Other approaches may use heuristic (rule-based) approaches, for example, neural networks, linear, or nonlinear discriminant learning. They use machine learning techniques to extract activity patterns from observed daily activities, and later use the patterns as predictive models. A more recent approach is based on ontological activity modelling and representation. It is closer to the logical approach in nature and uses a Description Logic-based markup language such as OWL and

RDF for specifying conceptual structures and relationships. Common, generic activity knowledge can be modelled at the conceptual level as an activity class described by a number of properties. These properties describe the types of objects that can be used to perform the activity. In this way, activity models can be created without the requirement of large amounts of observational data and training processes.

Another challenge in SmE regards the tracking of a user in indoor locations. Since the GPS positioning can't be used in indoor spaces due to the lack of signal in such locations, other means have to be used. As explained in [16], [18] and [20], there are several available technologies for the user location identification such as RFID (Radio-Frequency IDentification), Ultrasound-based systems, WiFi, Bluetooth, Optical/Vision-Based. Moreover, mobile and wearable devices such as smartphones and smartwatches are becoming common means for the user location identification; thanks to the built-in sensors such as gyroscope, accelerometer and compass, it can be possible to track the user movement and obtain important information about the user habits in order to provide better services in Smart Environments.

3.3 Ontologies and the Semantic Web

The information modelling is an important aspect to consider when the size of context information grows. In Smart Environment there is no standard: each system finds its own solution to describe and maintain all needed data. Possible solutions are: XML file, tables, Key-value Pair or other Ad-Hoc data structures. Another possible solution for the context modelling is given by the usage of ontologies. Ontologies, in the ICT, were born in the field of the Artificial Intelligence for the representation of the knowledge, they have been widely used in the Semantic Web for the definition of data models and, nowadays, they are being used in Smart Environments. The Artificial Intelligence literature contains several definitions of an ontology; as defined by Gruber [13], an Ontology is an explicit specification of conceptualization.

It is a formal explicit description of concepts in a domain of discourse, properties of each concept describing various features and attributes of the concept, as well as restrictions on those. An ontology, together with a set of individual instances of classes, constitutes a knowledge base.

Concepts are modelled by means of classes, which can have some attributes describing the concept itself. An example of class may be a car: all the relevant information of the car, such as colour, model, engine power, etc, are described by means of attributes. The attributes of an object are referred as data property. From a class, several subclasses can be defined, as a specialization of the superclass. For instance, we can have car subclasses like utility car, cabriolet car, etc. Each class can be related to other classes by means of relations named object properties. In the previous example a car is actually made of several components, each one described by a class. A car is related to other objects with specific object properties: it has an engine (one and only one), four tires (that must be of the same type) and so on. All the restrictions on a class or a data property, as well as the restrictions on the relations between classes, build the axioms of the ontology, rules that can't be violated for the ontology to be consistent. So a car can't have two or more engine and can't have tyres of different type.

In the context of the Semantic Web, ontologies are expected to play an important role in helping automated processes (so called "intelligent agents") to access information. In particular, ontologies are expected to be used to provide structured vocabularies that explicate the relationships between different terms, allowing intelligent agents (and humans) to interpret their meaning flexibly and unambiguously. For example, a suitable pizza ontology might include the information that Mozzarella and Gorgonzola are kinds of cheese, that cheese is not a kind of meat or fish, and that a vegetarian pizza is one whose toppings do not include any meat or fish. This information allows the term "pizza topped with (only) Mozzarella and Gorgonzola" to be unambiguously interpreted as a specialisation of the term "vegetarian pizza".

The W3C made a big effort in order to formalize a language for the creation of ontologies. Starting from the RDF[1] (Resource Description Framework), the OWL[2] (Web Ontology Language) has been released as a specific formalism for encoding ontologies.

OWL takes the basic fact-stating ability of RDF and the class- and property-structuring capabilities of RDF Schema and extends them. OWL can declare classes, and organise these classes in a subsumption ("subclass") hierarchy, as can RDF Schema. OWL classes can be specified as logical combinations (intersections, unions, or complements) of other classes, or as enumerations of specified objects, going beyond the capabilities of RDFS. OWL can also declare properties, organize these properties into a "subproperty" hierarchy, and provide domains and ranges for these properties, again as in RDFS. The domains of OWL properties are OWL classes, and ranges can be either OWL classes or externally-defined datatypes such as string or integer. OWL can state that a property is transitive, symmetric, functional, or is the inverse of another property, here again extending RDFS. OWL can express which objects, also called "individuals", belong to which classes, and what are the property values of specific individuals. Equivalence statements can be made on classes and properties, disjointness statements can be made on classes, and equality and inequality can be asserted between individuals. However, the major extension

[1]http://www.w3.org/TR/2003/WD-rdf-concepts-20030123/

[2]http://www.w3.org/TR/owl-ref/

over RDFS is the ability in OWL to provide restrictions on how properties behave that are local to a class. OWL can define classes where a particular property is restricted so that all the values for the property in instances of the class must belong to a certain class (or datatype); at least one value must come from a certain class (or datatype); there must be at least certain specific values; and there must be at least or at most a certain number of distinct values [15].

In the first release of the language, three versions, or dialects were available: OWL Full, OWL DL and OWL Lite. OWL Full is based on the first order logic; it has a big expressive power but its decidability is a NP (Nondeterministic Polynomial time) problem. OWL DL is the part of OWL that fits with the Description Logic and models specified with such language are decidable in a polynomial time. OWL Lite is a subset of OWL DL, has a lower expressive power and is not suitable to describe complex concepts. In 2009 the second version of the language has been released, the OWL2[3], that is the recommended version by the W3C. This new version gives a better distinction between the OWL2 Full language and its sub-languages, which are defined by means of the profiles. Three new profiles have been defined:

- **OWL 2 EL**, is suitable for applications where very large ontologies are needed and where expressive power can be treated for performance guarantee.
- **OWL 2 QL**, is suitable for applications where relatively lightweight ontologies are used to organize large numbers of individuals and where is useful to access the data directly via a relational query
- **OWL 2 RL**, is suitable for applications where relatively lightweight ontologies are used to organize large numbers of individuals and where is useful to operate directly on the RDF triples.

An ontology, by means of classes, restrictions on them, data and object properties, defines how a world of interest is made. The data of such world is represented as instances of the ontology classes, properly connected each other, creating a knowledge base. It is possible to infer complex information on the data, by using a reasoner that is a software able to infer logical consequences from a set of asserted facts or axioms that are defined in the ontology model. Since the knowledge in an ontology might not be explicit, a reasoner is required to deduce such implicit knowledge. Widely adopted reasoners are: Pellet[4], Fact++[5] or HermiT[6]. The model can be

[3]http://www.w3.org/TR/owl2-overview/

[4]http://clarkparsia.com/pellet/

[5]http://owl.man.ac.uk/factplusplus/

[6]http://hermit-reasoner.com/

queried by means of the SPARQL, a SQL-like language specific for the ontologies, or by means of the OWL API, a framework for ontology management.

In recent years, the interest for the semantic web has raised and a growing community of developers works to build the basis of the semantic web; several ontologies, called also vocabularies, have been released. On-line repositories make them available for downloading and extending. One of the principles of the semantic web is, in fact, reusing: each ontology may describe its world of interest defining in a proper manner classes, attributes and data properties. However, if a previous definition of that concept already exists in a suitable way, it is preferable not to define it again and import the already defined classes. By working on ontologies created by other authors, it is possible to extend them with new functionalities, mix with other vocabularies to create new ontologies.

The most widely adopted ontologies, that place the basis of the semantic web are: FOAF[7], DBpedia[8], DublinCore[9], SIOC[10] or GoPubMed[11]. A great number of ontologies are available in web repositories like LOV[12] (Linked Open Vocabularies) or Deri Vocabularies[13]. Another list of widely used ontologies can be found in the Semantic Web site section regarding ontologies[14].

The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries. OWL, RDF and Sparql languages are placed in the middle of the semantic web stack, as shown on figure 3.3

3.3.1 Ontology Evaluation

Ontologies are increasingly used in various fields such as knowledge management, information extraction, and the semantic web. Ontology evaluation is the problem of assessing a given ontology from the point of view of a particular criterion of application, typically in order to determine which of several ontologies would best suit a particular purpose [4].

Modern information systems are moving from "data processing" towards "concept processing", meaning that the basic unit of processing is less and less an atomic

[7]www.foaf-project.org/

[8]www.dbpedia.org

[9]www.dublincore.org/

[10]www.sioc-project.org

[11]www.gopubmed.org/web/gopubmed/

[12]www.lov.okfn.org/dataset/lov/index.html

[13]www.vocab.deri.ie/

[14]http://semanticweb.org/wiki/Ontology

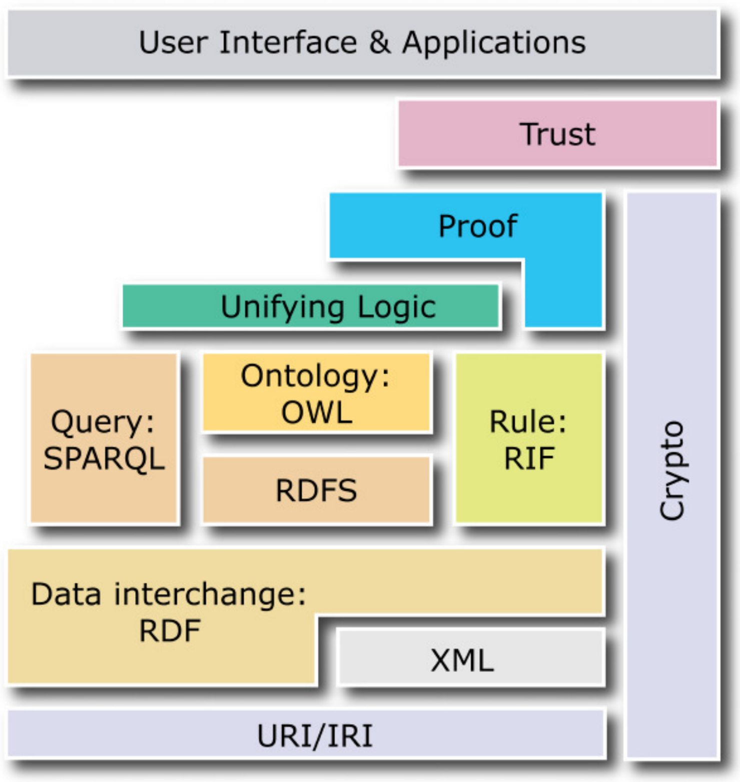


Figure 3.1. Semantic Web stack

piece of data and is becoming more a semantic concept which carries an interpretation and exists in a context with other concepts. Ontologies are used as a structure capturing knowledge about a certain area via providing relevant concepts and relations between them. Ontology evaluation is an important issue that must be addressed if ontologies are to be widely adopted in the semantic web and other semantics-aware applications. They may be evaluated over several level:

- Lexical, vocabulary, or data layer: the focus is on which concepts, instances, facts, etc. have been included in the ontology, and the vocabulary used to represent or identify these concepts;

- Hierarchy or taxonomy: ontologies typically includes hierarchical is-a relation between concepts. Such relationship is often important and may be the focus of specific evaluations;

- Syntactic level: ontologies are described in a formal language such as OWL and must match the syntactic requirements of that language;

- Structure, architecture, design: evaluation regards the compliance to certain pre-defined design principles or criteria such as the organization of the ontology and its suitability for further development;
- Context or application level: looks at how the results of the application are affected by the use of the ontology.

Type of ontology evaluations may vary with respect to the level for which the evaluation is considered, but they may be classified as the following:

- evaluation based on the comparison between the ontology with a "golden standard" that is the comparison with a well known and accepted reference ontology;
- evaluation based on the comparison of ontologies in a specific application, by taking into account results;
- evaluation based on the domain to be covered by the ontology;
- evaluation based on how well the ontology meets a set of predefined criteria, standards, requirements (e.g. number of classes, imported ontologies).

Evaluation type and the level to which the evaluation is performed depends on the purpose of the evaluation and the application where the ontology is used. Most of the evaluations, especially the ones at the context level, are performed by expert humans, due to the lack of automatic tools for evaluation. Researches on this field may lead to new applications for ontologies evaluation.

3.4 Ontology-based applications

Several projects apply ontologies as a central concept for modeling context information. One of the first projects was CoBrA [6], a broker-centric agent architecture for supporting context-aware systems in smart spaces. Central to the CoBrA architecture is the presence of an intelligent agent called the context broker that is a specialized server which role is to maintain a shared model of context on the behalf of a community of agents and devices in the space. Moreover, it reasons about contextual information that cannot be directly acquired from the sensors (e.g., intentions, roles, temporal and spatial relations) and detect and resolve inconsistent knowledge that is stored in the shared model of context. Agents and device get information from the context broker in order to provide some service.

Context-Driven Adaptation of Mobile Services (CoDAMoS) [23] defines a generic ontology to model context in Ambient Intelligence infrastructures that suits the requirements of mobile computing. This ontology is based on four general entities:

- The user is the central entity, including the user's profile, preferences, mood and current activity. The rest of the entities should adapt to the user, not vice versa.
- The environment in which the user interacts, including information such as temperature and lighting.
- The platform that describes the hardware and software of a device, including device resources such as memory and bandwidth.
- The service that provides specific functionality to the user.

The Service-Oriented Context-Aware Middleware (SOCAM)[14], is an architecture for building context-aware services based on a two-level context model. This middleware acquires context information from different sources and interprets it. The context ontology is divided into a two-level hierarchy, distinguishing between common and specific context information. The upper level describes global concepts of the ontology and captures general knowledge about location, type of entity, person or activity. On the other hand, the lower level is divided into several pervasive computing sub-domains, each one of which defines specific details and properties for each scenario. Depending on the situation and the available devices, an appropriate sub-domain is selected from the lower level. When environment changes are detected, the lower level ontology can be dynamically plugged into and unplugged from the upper ontology, thus dynamically changing this association. This mechanism appears to be very reasonable also with respect to resource limited devices. An ontology resulting from the extension of the top-level ontology with a domain-specific ontology can be kept quite small in comparison with a single huge ontology capturing all potentially involved concepts.

The representation of data with an ontological approach, is able to provide a consistent and formal data model on which several inferences can be performed. There are different ways in which an ontology can be included in applications, but we focus mainly on two frameworks related to the java world.

The first framework is OWL API. Its focus is on providing Java interfaces and classes for OWL language constructs for all three standard dialects. As interfaces and implementations are strictly separated, developers can add their own specific implementations to support particular features. Although the OWL API does not contain a reasoner, it foresees extension points for adding external reasoners. The basic version of OWL API supports in-memory computation only, which poses certain limitations on scalability when working with large ontologies.

The second well-known ontology programming framework is Apache Jena. Instead of directly working with OWL constructs, it adds a level of abstraction and works on general graphs, which allow for the processing of ontologies in RDF(S), all

OWL dialects, and their predecessor DAML. Jena supports in-memory processing as well as processing of persistent ontologies with a lazy loading mechanism and is thus suitable for implementing highly scalable applications working with large-scale ontologies. It contains a simple set of built-in reasoner, like Pellet, Racer or FaCT, but doesn't support external reasoners (DIG interface is no longer available) [22]. By using Jena, it is possible to perform SPARQL querying, while it isn't possible with the OWL API. On the other hand, Jena lacks in terms of compatibility with external reasoners; the most important ones, like HermiT, are not supported, while they are supported by the OWL API. The choice of a framework instead of another, thus, depends on the type of inferences the reasoner has to perform. OWL API is suitable for ontologies requiring complex inferences by the reasoners, but a more complex management of information gathering is required. Jena is suitable for ontologies that don't require advanced inferences by the reasoner and provides a more simple mechanism for information extraction. An example of an ontology-based Application is given by the Dog gateway.

3.4.1 The Dog Gateway

With the aim of solving interoperability problems and bringing more intelligence in the home, the ed-Lite research group of Politecnico di Torino, is working on Dog, a smart home gateway, able to manage different home automation networks as a single system.

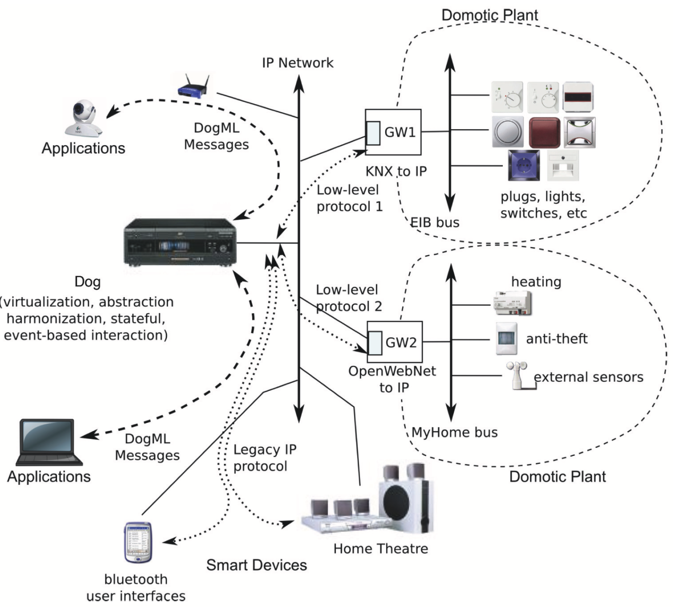


Figure 3.2. Architecture with Dog inclusion

By adding this gateway over existing home automation systems, a more complex and advanced management of the house is possible.

Dog exploits the OSGi (Open Service Gateway initiative) as a framework coordination for supporting dynamic module activation, hot-plugging of new components and reaction to module failures. Such basic features are integrated with an ontology model of domotic systems and environments named DogOnt.

DogOnt is an OWL2 model for the domotics domain describing where a domotic device is located, the set of its capabilities, the technology-specific features needed to interface it and the possible configurations it can assume.

It allows the house structure modelling, contained domotic components, their states and functionalities. Moreover, it models how the home environment is composed and what kind of architectural elements and furniture are placed inside the home [2].

Dog uses the DogOnt ontology for implementing several functionalities encompassing: command validation at run-time, using information encoded in functionalities, stateful operation and using the state instances associated to each device.

With its last release (ver. 3.0), in 2013, Dog is organized in a layered architecture with 4 layers, each dealing with different tasks and goals, ranging from low-level interconnection issues to high-level modelling and interfacing. Each layer includes several OSGi bundles, corresponding to the functional modules of the system.

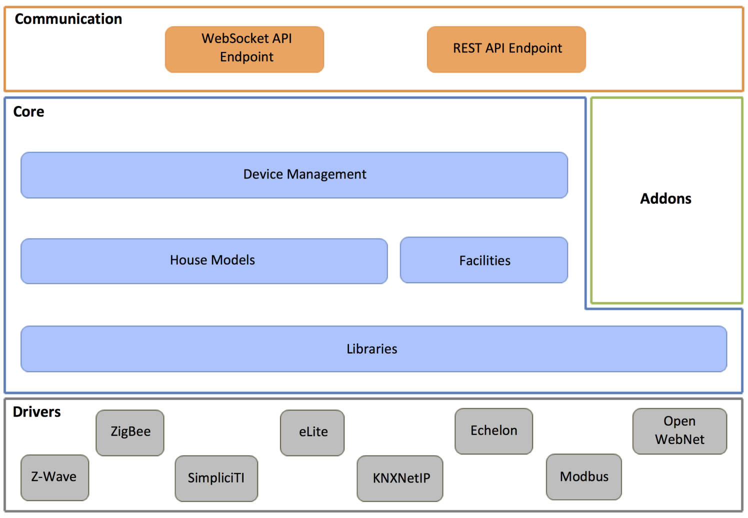


Figure 3.3. Dog stack

- **Drivers**: encompasses the Dog bundles that provide an interface to the various home and building automation networks connected to Dog. Each network technology is managed by a set of dedicated drivers, one per each different technology (e.g., KNX NetIP, Z-Wave, etc.), which abstract network-specific protocols into a common, high-level representation that allows to drive different devices in a transparent way. Each driver implements a "self-configuration" phase, in which it interacts with the OSGi framework to retrieve all the needed low-level information, according to the specific technology, e.g., the device address(es) or its ID in the network. For each technology, a Network Driver, a Gateway Driver and at least one Device Driver should exist;

- **Core**: hosts the core intelligence of Dog, based on the DogOnt ontology. Moreover, it provides a set of common libraries and services useful to the entire system. The Core library is the most important part of the Dog stack: it contains all the possible devices, functionalities, states and state values, as defined in DogOnt All these classes and interfaces are programmatically generated from DogOnt, thus ensuring a formal and full compliance with the ontology representations. The core provides core-level notifications, as well as common data structures needed by the other Dog bundles. Moreover, it provides utility classes, to avoid code duplications and repetitions;
- **Addons**: includes additional bundles for injecting further capabilities or more intelligence to the "core" part of the system, such as data storage, stream processing, rule engine, power model, event storage, etc;
- **Communication**: provides the bundles offering access to external applications, either by means of a REST Endpoint or via WebSocket. Both APIs use the same, basic, message structures and expose the same functionalities and information: they help retrieving the building configuration, sending commands to devices managed by Dog, handling the building structural information (rooms, flats, etc.), getting the devices status, etc. The only exception concerns asynchronous events (i.e. notifications) that may come from any device handled by Dog: they are provided only by the WebSocket endpoint due to the client-server architecture of HTTP.

Dog has been proposed as the home gateway in the Notont-based Intelligent Notification System project: it manages all the house plants, devices and sensors, providing information that is processed by the Notont software for the message generation and its delivery to users.

Chapter 4

Notont: the Notification ontology

4.1 Introduction

To create a system able to interact with users and to communicate some information from the house in a proper way, information about users, their activity, location and state, their devices with their capabilities have to be known. Moreover, the house, its plants and their current state have to be known as well, as they are the source of all the information routed to the users. The Notont ontology puts together the relevant information needed for the routing of messages coming from or generated by the smart home, to the most suitable end user device. In this chapter, the description of the ontology is presented, focusing on the imported ontologies and the most relevant classes used in the thesis work.

4.2 Related Works

The definition of a new ontology may lay on some already existing ontologies. The research community has shown a lot of interest in defining ontologies for context modelling in smart environments. Over the last decade, several ontologies have been released leading the basis for the definition of the Notont ontology.

One of the first ontologies developed for context modelling in smart environment is Cobra-Ont. Cobra-Ont Ontology is categorized into four distinctive but related themes: ontologies about physical places, agents (both human and software agents), agents location context and agents activity context. It is used in the CoBrA project as explained in 3.4

Another ontology related to the CoBrA project is SOUPA (Standard Ontology for Ubiquitous and Pervasive Applications). It consists of two sets of ontology documents: SOUPA Core and SOUPA Extension. The SOUPA core ontology consists of nine ontology documents that define vocabularies for describing person, contact information, beliefs, desires and intentions of an agent, actions, policies, time, space and events [7]. The SOUPA Extension ontologies, extended from the core ontologies, define additional vocabularies for supporting specific types of applications and provide examples for defining new ontology extensions.

CONON [29] is an ontology for context modelling in pervasive computing environments, and for supporting logic-based context reasoning. CONON divides the context model into upper ontology and specific ontology. The upper ontology is a high-level ontology which captures general features of basic contextual entities such as Person, Location and Device. Specific ontology is a collection of ontology set which define the details of general concepts and their features in each sub-domain.

DomoML-env [26] describes a smart environment in terms of Building-Equipment for house appliances, Component for simple elements as switches, valves, sensors; Core-Foundation for technical or basic elements that a Component needs, Building-Environment for house infrastructural components as kitchen, dining room and Location, which defines the location of each element or object within the domestic environment, allowing establishing spatial relationships.

Several other ontologies have been defined focusing on particular domain such as people, devices, houses or services. We explored on-line repositories, searching for suitable ontologies to import and extend in order to realize Notont.

FOAF[1] (Friend of a friend) is an RDF based schema to describe people and their social network in a semantic way. It is widely adopted in many ontologies for people description and is included in online resources for annotating user pages, or describing articles about people.

CEO[2] (Consumer Electronics Ontology) is a vocabulary for describing typical consumer electronics products, services and their features, e.g. digital cameras, camcorders, etc. It can be used in combination with GoodRelations[3], for better describing devices and such combination is used for e-commerce purposes.

mIO![4] is a complex ontology for describing ubiquitous services in an intelligent environment and it is used in the mIO! project that aims to provide technologies able to adapt themselves to everybody and to the context. mIO! is made by several ontologies, each one for a specific concept. There are ontologies describing services, users with they relations (imported from FOAF), environment and locations. An interesting ontology used by mIO! is Device[5], the ontology used for devices description. Such ontology describes in detail devices, their features and capabilities

[1]http://www.foaf-project.org/

[2]http://www.ebusiness-unibw.org/ontologies/consumerelectronics/v1

[3]http://purl.org/goodrelations/

[4]http://www.cenitmio.es/

[5]http://www.cenitmio.es/ontologies/Device.owl

ranging from mobile devices to printers and sensors.

BOnSAI[6] describes a smart environment by means of concepts like location, devices, operations and services. It is based on CoDAMoS from which imports some of the definitions like user profile. The ontology covers the representation of several concepts for a smart environment, although they are described in a general manner, with a poor subclasses definition.

Locont[7] is an ontology describing the concept of person, activity, location and artifact with aim of classify the user current activity in relation with is posture and interaction with artifacts. Several activity subclasses are defined, making this ontology interesting for the user description.

4.3 The Notont Ontology

Notont is an ontology for the context modelling in a smart environment, with the aim of maintaining information to provide smart notifications to the users. The ontology models information about the users, their devices, the house where these users live and the type of message to send. In order to obtain such model, several ontologies have been used, for an overall of 27 imported ontologies, and a final number of classes equal to 1385. By following the reusing principle suggested by the W3C, Notont defines those classes that have not been found in other ontologies; classes that have been found compliant with the specifications have been imported in the ontology. Notont connects and put together all the relevant imported classes, set the equivalences between classes and object properties of such ontologies.

Ontologies selection is based on their evaluation at the application and context level. Ontologies have been compared to meet as more as possible to the interested domain. Domain specification is guided by the following test case scenario where a family lives in a smart home equipped with a smart notification system:

Giulia works as employee and owns a smartphone and a tablet. She is interested in maintaining her home comfortable and being informed about the appliances usage to optimize her time while she is at home.

Luca works as a manager and spends a lot of time out of his home. He owns a smartphone and is mainly interested in saving energy and being informed about appliances or security alert.

While Giulia is cooking, the dryer completes its cycle. Since the activity keeps her hands busy, devices are not currently in use and the kitchen is equipped with a controllable HiFi device, the system infers that the best device to use is the Hifi

[6]http://lpis.csd.auth.gr/ontologies/bonsai/BOnSAI.owl

[7]http://webmind.dico.unimi.it/CARE/locont.owl

system, so the message about the dryer is sent as vocal message to the kitchen speakers. She puts the dinner in the oven, set a timer to 30 minutes and decides to read an e-book from her tablet, relaxing in the living room couch. After 30 minutes, the oven turns off and, by taking into account that Giulia is reading using her tablet, the system decides to send her a message about the oven directly on her tablet. Luca comes back home and starts to set the table in the living room while the window sensor in the upper floor signal an open window. Since no presence is detected in the floor and the weather isn't particularly windy, the system infers that it could be an intrusion, so an alarm is sent to luca's smartphone and to the living room smart tv, due the high priority of the message.

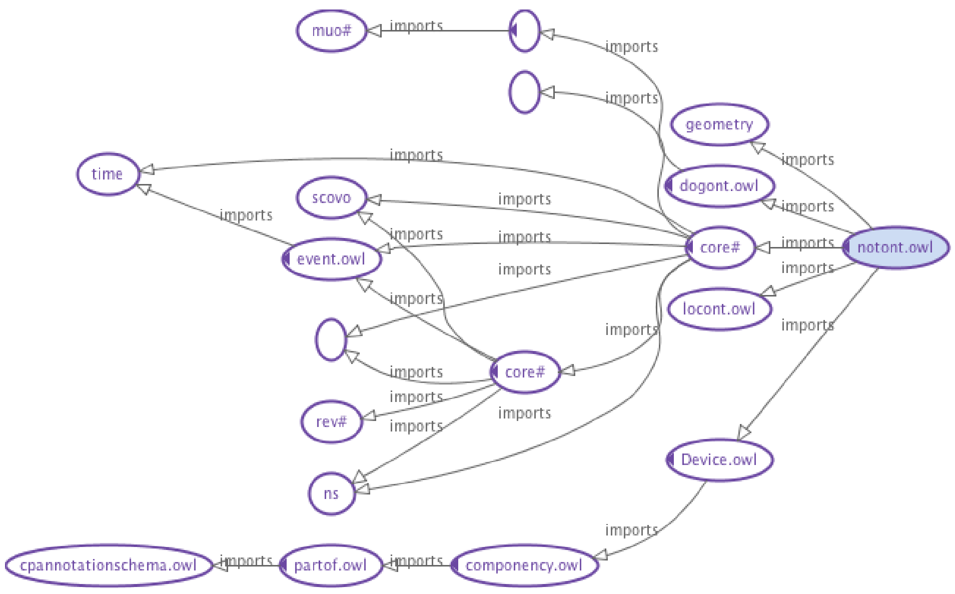


Figure 4.1. Notont import tree

By taking into account the defined use case scenario, Notont focuses on four main entities: the user, the device, the house and the message category. The user is the most important element to model since, by knowing as more as possible about him, it is possible to tailor the system behaviour to his needs. Devices are the means of interaction between the house and the users; it is important to know how devices interact with the user, so it is useful to model their capabilities. The house is the generator of all messages routed to users. These messages are the consequences of particular events related to the house, its plants or some external service managed by the central unit (as the weather forecast). Messages are divided by category, so that users can decide which ones they are likely to receive.

4.3.1 User Modelling

The most relevant information about the users is modelled by means of the Locont[8], Foaf[9] and Core[10] ontologies. The main class used is Person, actually available in more ontologies and originally defined in Foaf. Locont extends it by adding subclasses; so a user can below to one of the classes shown in figure 4.2. It can be seen that the definition of Person is used in three other ontologies (Locont, Schema and Contact). The original subclasses definition has't been modified, but they can be easily extended and new subclasses of Person can be defined from Notont ontology.

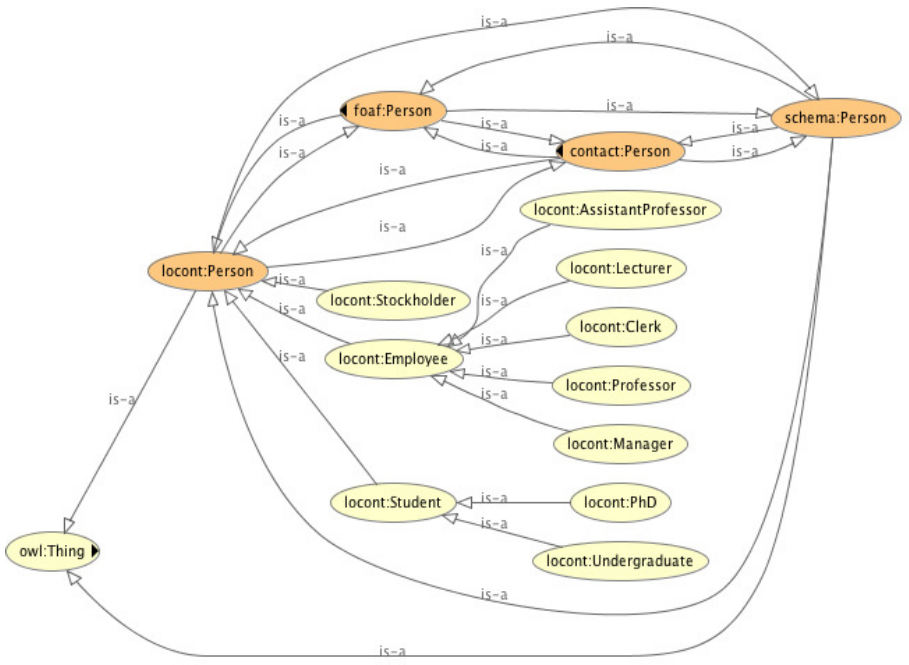


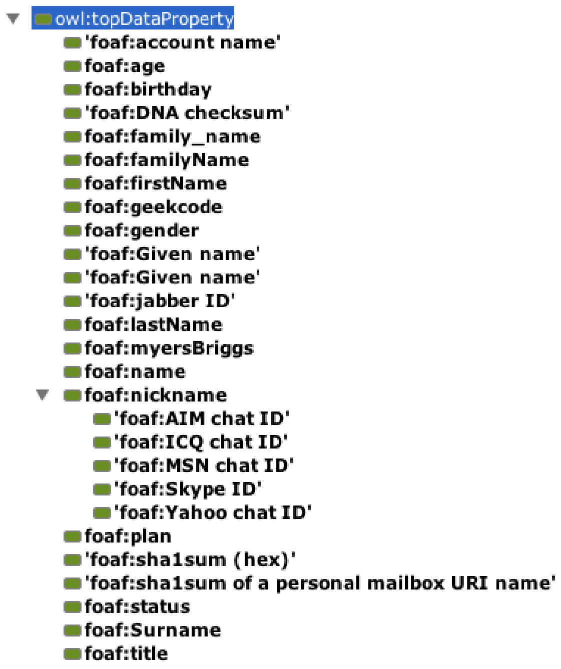
Figure 4.2. Notont: Person specification

From the Foaf ontology it is possible to specify several data properties such as name, family name, gender and birthday (figure 4.3). These information are used to customize the interaction with the users, for example by calling them with their nickname. A Person can be located in a particular place through the object property “hasCurrentSymbolicLocation” defined in Locont, that connects the class Person to the class SymbolicLocation. The SymbolicLocation is the generic class that specifies

[8]http://webmind.dico.unimi.it/CARE/locont.owl

[9]http://xmlns.com/foaf/0.1/

[10]http://purl.org/ontology/wi/core



```
▼ owl:topDataProperty
    'foaf:account name'
    foaf:age
    foaf:birthday
    'foaf:DNA checksum'
    foaf:family_name
    foaf:familyName
    foaf:firstName
    foaf:geekcode
    foaf:gender
    'foaf:Given name'
    'foaf:Given name'
    'foaf:jabber ID'
    foaf:lastName
    foaf:myersBriggs
    foaf:name
  ▼ foaf:nickname
        'foaf:AIM chat ID'
        'foaf:ICQ chat ID'
        'foaf:MSN chat ID'
        'foaf:Skype ID'
        'foaf:Yahoo chat ID'
    foaf:plan
    'foaf:sha1sum (hex)'
    'foaf:sha1sum of a personal mailbox URI name'
    foaf:status
    foaf:Surname
    foaf:title
```

Figure 4.3. Foaf data property specifications

all the house elements like rooms, buildings, artifacts and their specializations. Its subclasses are connected to the Dogont ontology, as better explained in section 4.3.3. Knowledge of the user location let us infer the knowledge about appliances in that particular place, thus, we can infer if other means to communicate with users are available and how this communication may happen.

For the class Person, it is possible to know the actual performed activity through the object property "hasCurrentActivity" defined in Locont that connects a Person to an Activity instance. The available activities are divided in individual and social and several subclasses are defined, as shown in figure 4.4. In particular, for the individual activities, the figure 4.5 shows the tree of the defined subclasses.

Notont, through the Locont ontology, defines several rules to infer the user activity. For example the figure 4.6 shows the rule to infer the Sleeping activity. This rule asserts that a Person sleeps if he/she is located in a Bedroom (that contain the Person), light sensors measure a light intensity of less than 40 lux and sound sensors measure a sound level of less than 30 db. Due to the complexity of the activity recognition that requires a big effort to be accomplished with good results, it hasn't be explored in this work so, as for the location, this information is assumed to be known and provided to the system by some external source.

The knowledge of the current activity is an important element to determine the

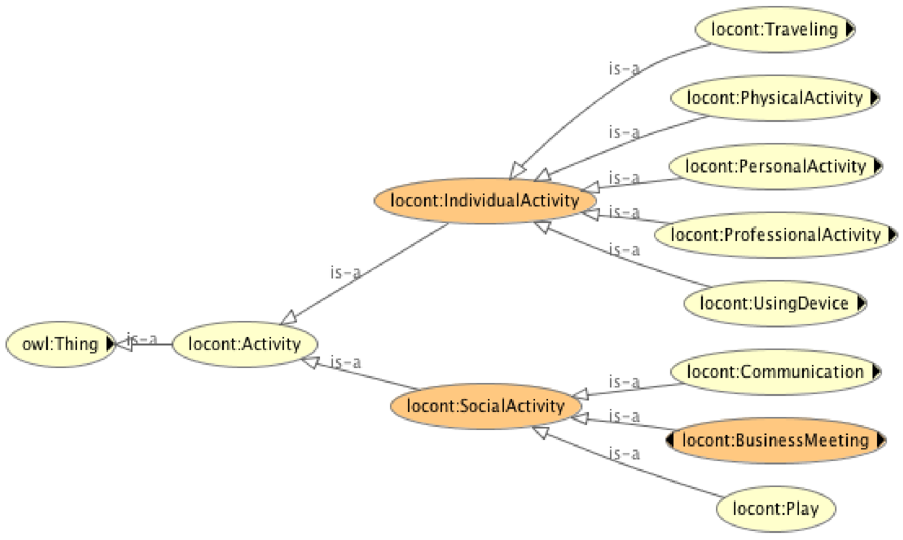


Figure 4.4. Activity tree in Notont

availability of the user in receiving some information. For each activity an Accessibility instance is connected through the “hasAccessibilityLevel” object property. Four instances (individuals) of this class describe to which accessibility level the user is when performing a specific activity. These four instances are:

- accessibility_free, when the activity lets the user free to perform every type of interaction. Activities with this accessibility level are, for example, StandingStill or MovingByTrain.
- accessibility_freeableHands, when the activity requires that the Person uses its hands but they can be used to perform some other action in the same time. Examples of such activities are Eating or WorkingAtPc.
- accessibility_freeableWrist, when the activity requires that the Person uses its hands but he can still take a look to a smartwhatch placed on the wrist. Example of these activities are GivingClass or WritingOnBlackBoard.
- accessibility_notFree, when the activity requires that the Person uses its hands and they can’t be used to do something else. Examples are Showering, MovingByCar or Sleeping.

Each activity can be connected to a particular location through the “canTakePlaceOn” object property and may happen in a particular moment of the day by connecting it to a TimeGranularity through the “canTakePlaceDuring” object property.

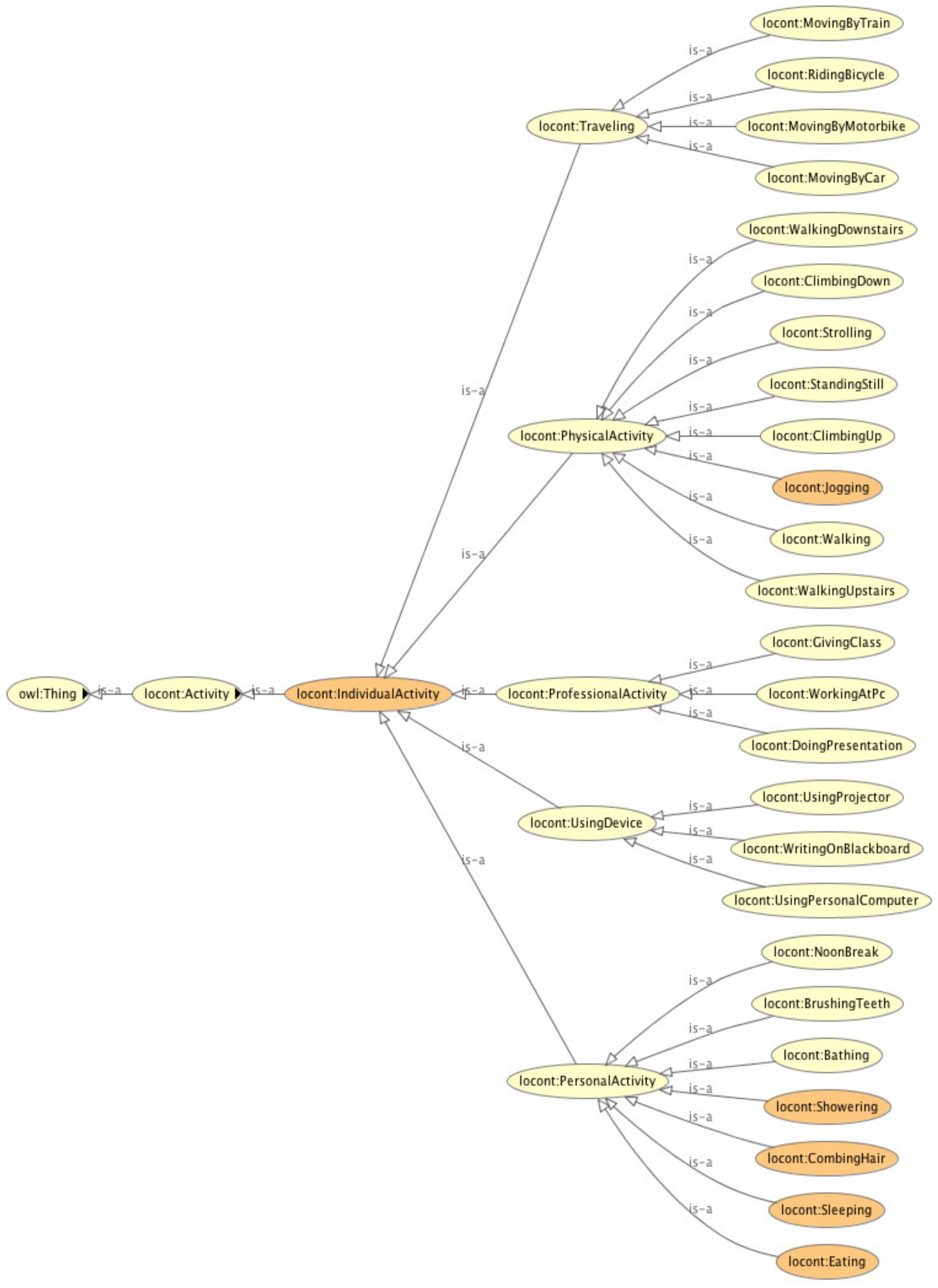
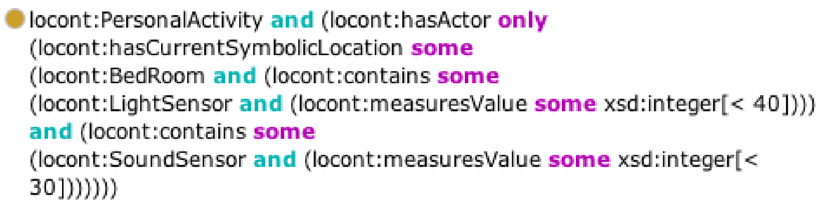


Figure 4.5. IndividualActivity tree in Notont



```
locont:PersonalActivity and (locont:hasActor only
(locont:hasCurrentSymbolicLocation some
(locont:BedRoom and (locont:contains some
(locont:LightSensor and (locont:measuresValue some xsd:integer[< 40]))
and (locont:contains some
(locont:SoundSensor and (locont:measuresValue some xsd:integer[<
30]))))))
```

Figure 4.6. Rule to infer the Sleeping activity

In addition to the Accessibility for the Activity, each Person can define its own state of Obtrusiveness through the object property "hasObtrusivenessLevel" that connects an instance of the class Person to an instance of the Obtrusiveness class, defined in the Notont ontology. This class define which type of interaction the user is willing to accept among all the possible. For this work we considered three main type of notifications: audio from the HiFi system, video from smart TV and notifications from mobile devices. Users can choice their obtrusiveness level among these eight:

- obtrusiveness_available: all the possible means of communication can be used to notify a message to a particular user;
- obtrusiveness_notAvailable: no means of communication have to be used since the user doesn't want to receive any notification;
- obtrusiveness_houseOnly: just house devices can be used to notify a message. Mobile and personal devices will not be taken into account for the message delivery;
- obtrusiveness_mobileOnly: just mobile and personal devices can be used to notify a message. House devices will not be taken into account for the message delivery;
- obtrusiveness_audioOnly: just the HiFi system (if any in the user location) can be used to notify a message. All the other devices are not used;
- obtrusiveness_videoOnly: just the TV (if any in the user location) can be used to notify a message. All the other devices are not used;
- obtrusiveness_noAudio: All mobile and personal devices and all available house devices except for the HiFi system can be used to notify the message;
- obtrusiveness_noVideo: All mobile and personal devices and all available house devices except for the video system can be used to notify the message.

4.3.2 Device Modelling

The modelling of the devices is achieved mainly by means the the Device[11] ontology. This vocabulary has born with the aim of describe devices and all their features. The main ontology class is Device that define several subclasses as shown in the figure 4.7.

In particular, MobileDevices define all mobile devices as shown in figure 4.8. This is the main class used to describe all the devices owned by the users and their features. It can be seen as the "Mobile phone" class (from Device ontology) is set in equivalence with the Smartphone class (from Locont). For each device it is possible to specify several information such as screen size, touch screen technology, type of camera speakers and microphone. All these features are described by means of classes and are connected to the device instances through the "hasDeviceTechnology" object property that connect a Device to another. House artifacts may be modelled by means of this ontology as well. A subclass of Device is Sensor that defines some type of sensors such as TemperatureSensor or CO2Sensor. These classes have been set in equivalence with the same classes in the Dogont ontology which describes the house and is better discussed in the next section. Each Person can have one or more devices and the association is realized by means of the "usingArtifact" object property defined in Locont. For the purpose of this work we are interested in modelling device physical features such as screen size or the loudspeakers, that are useful to understand how a message can be delivered to such device. These features are modelled by means of classes like Display, LoudSpeaker and GPSReceiver. Thanks to these information we can infer for each owned device, which type of interaction they can perform. For example, by knowing that a device has a loudspeaker, we can assume that an audio message may be delivered to such device.

4.3.3 House Modelling

Notont is able to model the house, its devices and appliances with their functionalities, notifications and states. The modelling is achieved by means of Dogont[12] ontology, developed by the e-Lite team of the Politecnico di Torino and Locont[13] ontology. The environment is defined by the subclasses of BuildingEnvironment as shown in figure 4.9. It can be seen the integration of the Dogont ontology with Locont that extends the availability of classes for the environmental description. Each BuildingEnvironment is connected to another through the "contains" object

[11]http://www.cenitmio.es/ontologies/Device.owl

[12]http://elite.polito.it/ontologies/dogont.owl

[13]http://webmind.dico.unimi.it/CARE/locont.owl

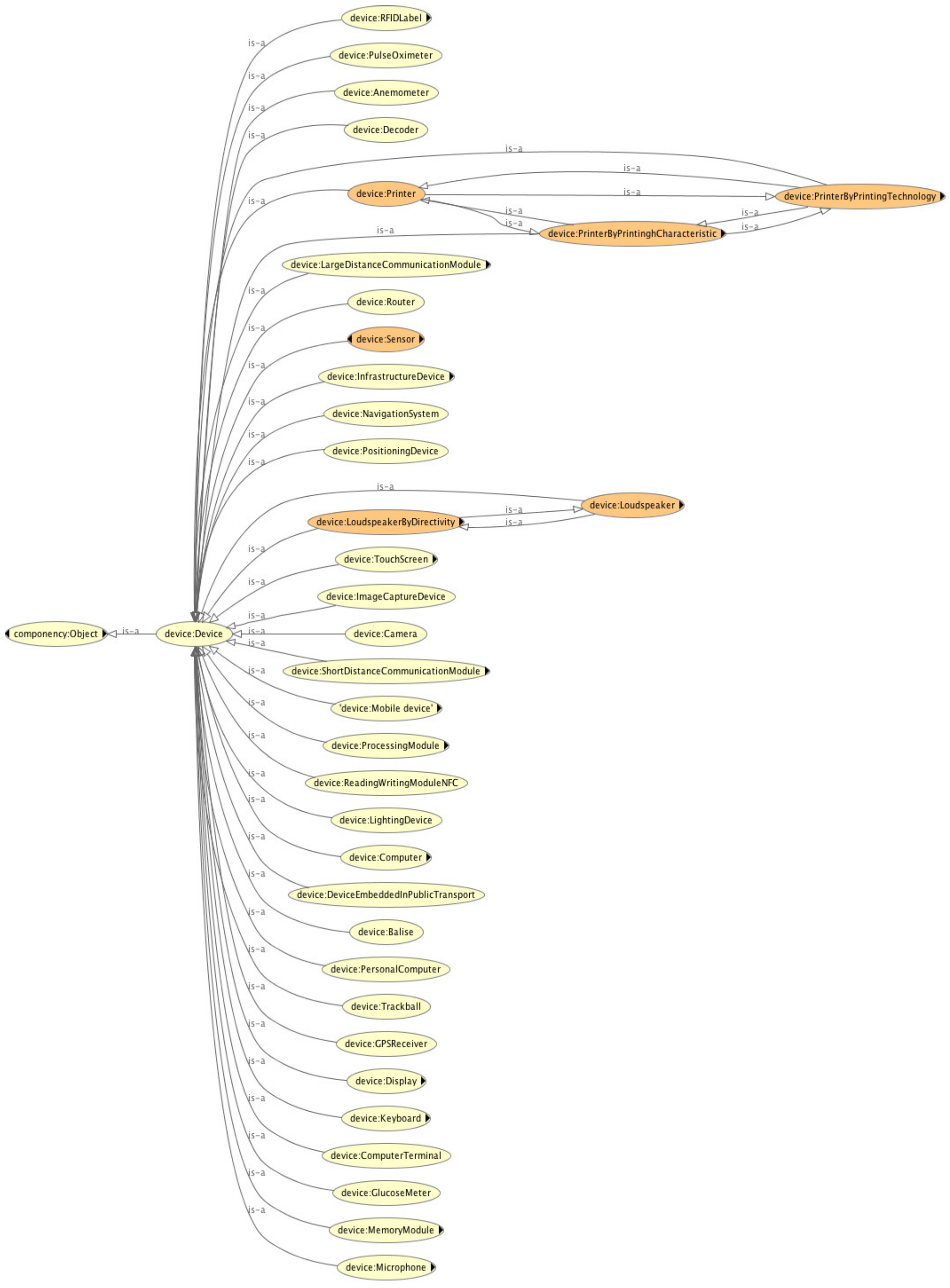


Figure 4.7. Devices tree in Notont

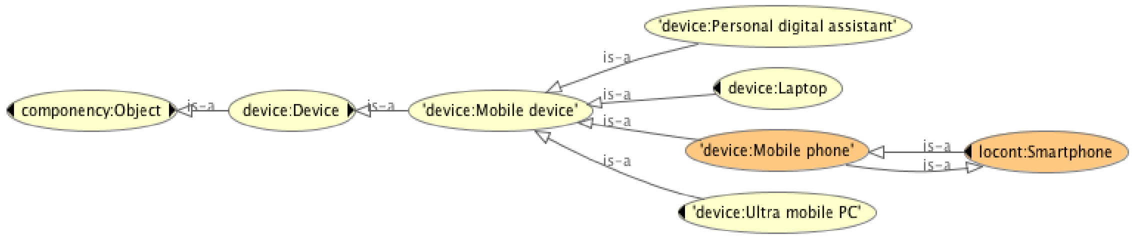


Figure 4.8. Mobile Devices tree in Notont

property and its inverse which is “isIn”. Moreover, the object property “hasArtifact” from Locont has been put in equivalence with “contains”, making available the extension of Dogont.

BuildingEnvironment subclasses can be either described in terms of their architectural elements such as ceiling, wall or windows, as shown in figure 4.10. This is an important aspect since, by knowing the architectural description, we have information on the devices positioning relatively to the environment they are placed in. These information are essential for the device mapping inside the house and in a specific location.

Domotic devices, smart appliances and house furniture are described by means of the BuildingThing class. This class defines two sublasses: Controllable and UnControllable. The first subclass is used to model all devices and appliances that can be controlled by a central unit such as a pc and for which are available information about their states and functionalities. The second subclass is used to model the house architectural aspect (as already presented on figure 4.10) and its furniture (Furniture).

The Controllable class, from Dogont, is one of the most complex class in Notont. It is set in equivalence with the Device class from Locont and Device ontologies, creating a class with several subclasses for modelling house plants, appliances and devices. Three subclasses are defined:

- Appliances: describes all the appliances of the house like oven, fridge or TV. Is divided in White and Brown Goods as shown in 4.12

- HousePlants: describes all house appliances, plants and facilities as HVAC system, sensors, electric components as shown in fig 4.13

- NetworkComponent: is the main class that defines all gateways for the communication with different type of available networks such as KNX, ZWave, or ModBus. Figure 4.14 shows this tree.

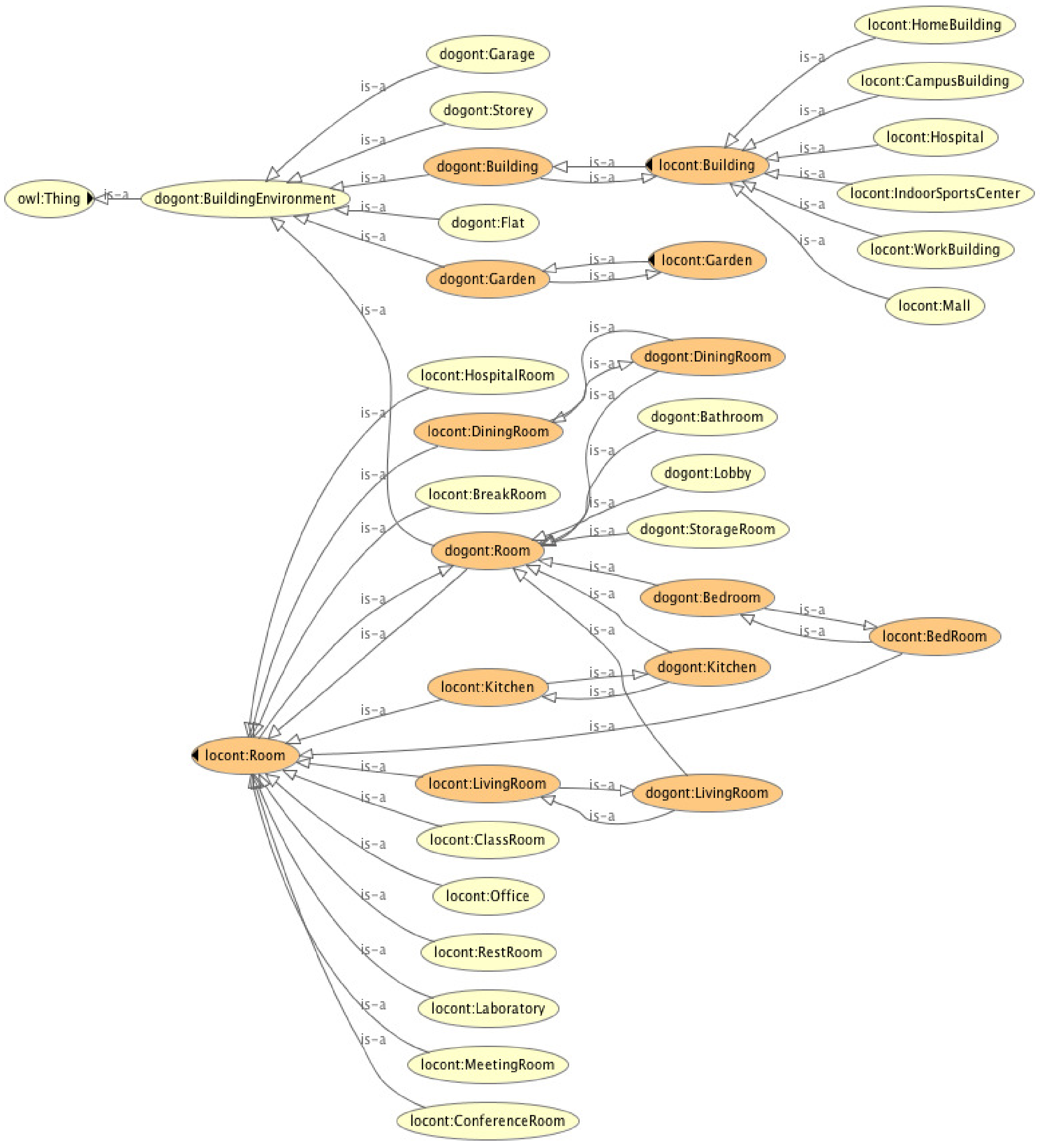


Figure 4.9. BuildingEnvironment tree in Notont

Controllable devices are modelled in terms of functionalities and states. Functionalities describe how a given device can be controlled, queried and whether it can autonomously generate notification events. Each functionality defines the commands to modify a given device property (e.g., light intensity) and the values they can assume. Functionalities are divided in different classes on the basis of their goals:

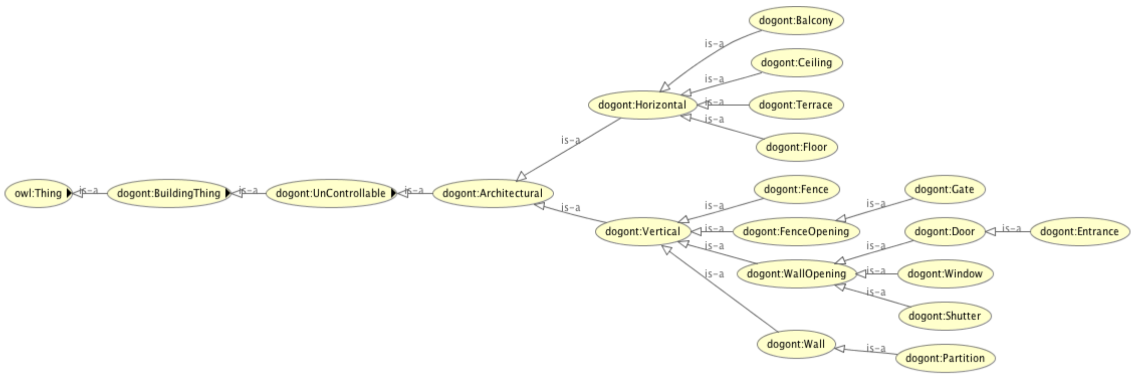


Figure 4.10. Architectural tree in Notont

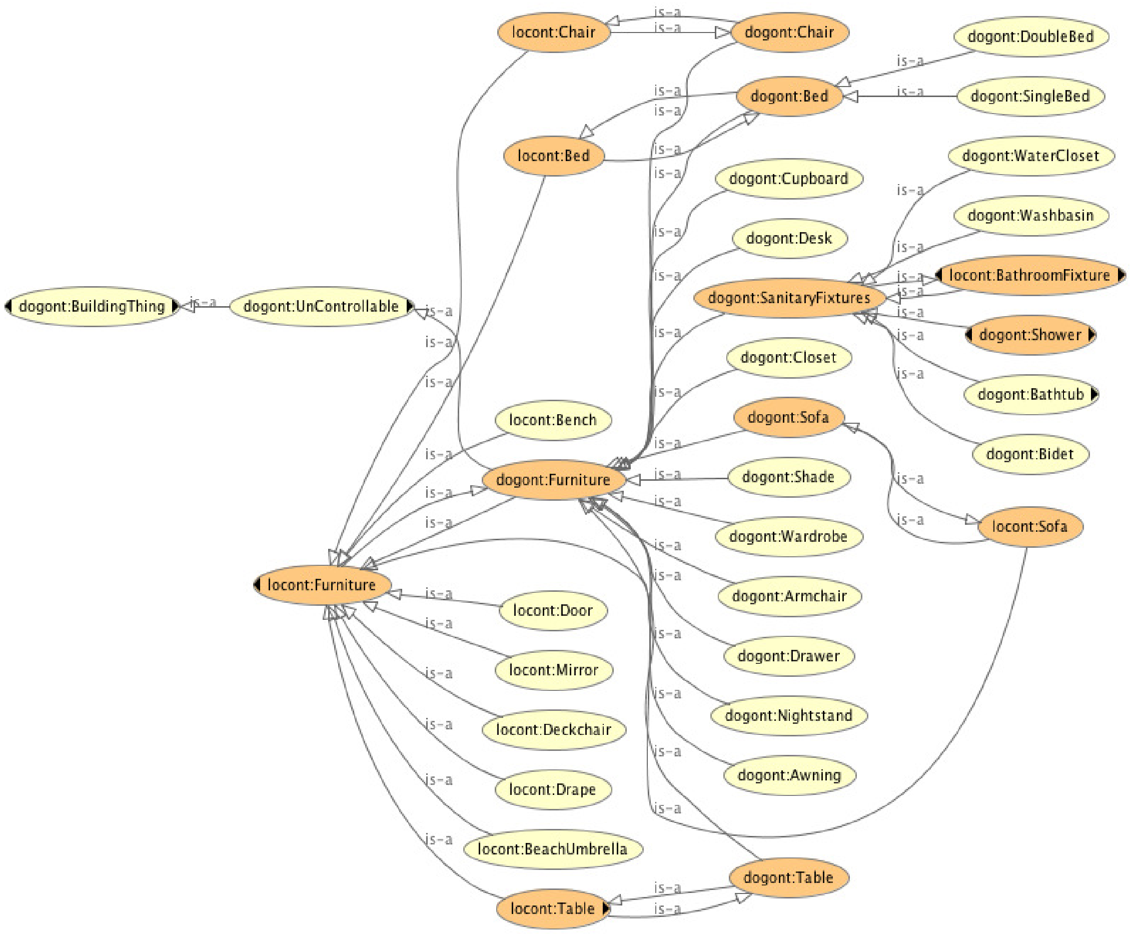


Figure 4.11. Furniture tree in Notont

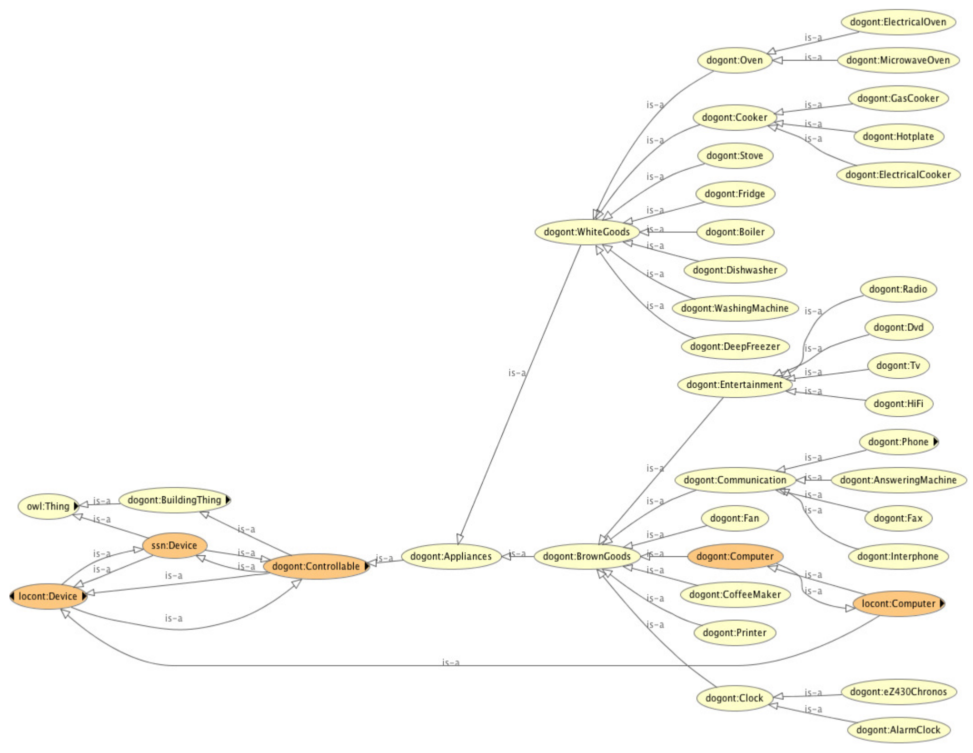


Figure 4.12. Appliances tree in Notont

- Control Functionalities: model the ability to control a device or a part of it, e.g., to open up a shutter;
- Notification Functionalities: represent the ability of a device to autonomously advertise its internal state and in particular the ability of detecting and signalling state changes;
- Query Functionalities encompass the capabilities of a device to be queried, or polled, about its condition, e.g., failure, internal state values, etc.

Device interconnections are modeled by the "controlledObject" object property and the same device can be involved in different connections with different roles, i.e., as either a controller or a controlled device.

Each Device can have one or more states depending on the device type. For example a sensor may have a temperature state while a ventilation unit may have a CO_2 measurement state; these are modelled by means of the State class. The state value is defined by means of the StateValue class to which each State is connected through the "hasStateValue" object property. The current state of a device is therefore defined by a list containing one StateValue per each State.

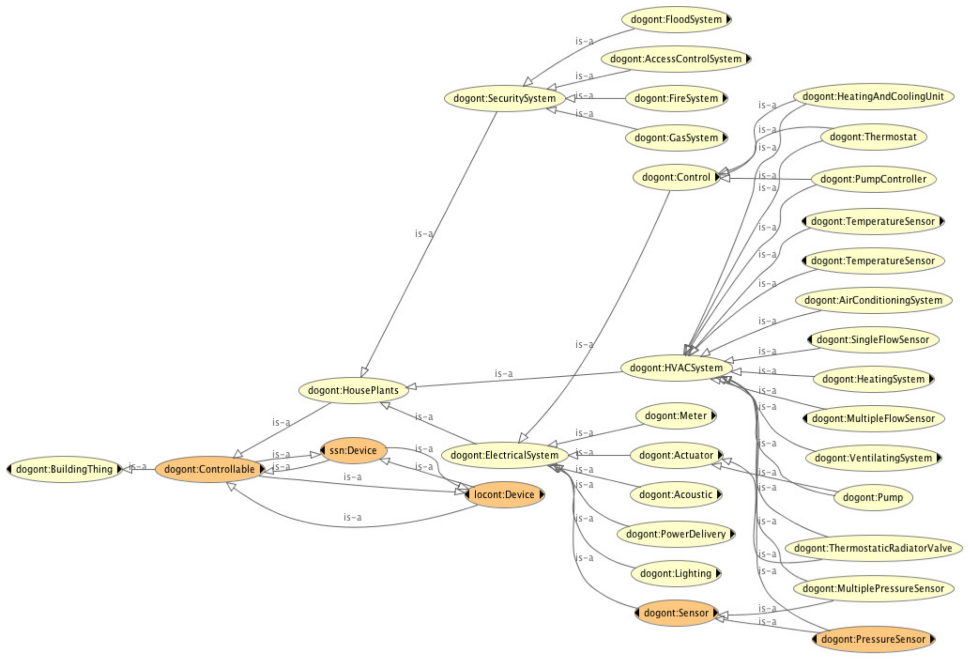


Figure 4.13. HousePlants tree in Notont

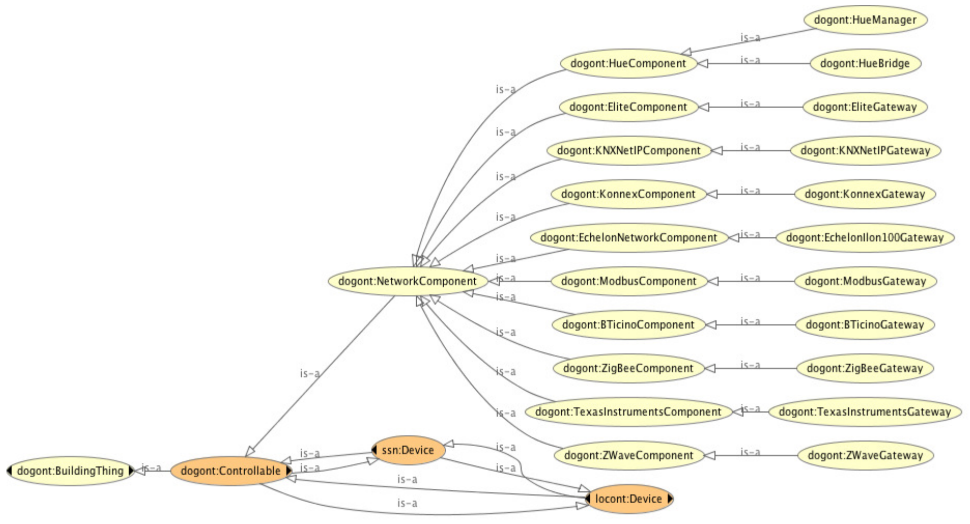


Figure 4.14. NetworkComponent tree in Notont

In a Home Automation System modelled with Dogont, devices may belong to different subnetworks. The intercommunication is grant through the NetworkComponents instances that provide an interface to communicate with those devices within the subnetwork. This feature, modelled by Dogont, is exploited by the Dog gateway (see chapter 3.4.1) that is able to communicate with different protocols in a transparent way. Devices are seen as belonging to a unique network and they are able to communicate each other to perform intelligent actions.

4.3.4 Message Category Modelling

The last important element to consider is the message type the user is willing to receive. Message classification is obtained through the class WeightedInterest defined in the Core[14] ontology. Several classifications are possible and users decide to which are interested. Each preference has a weight that expresses the priority of the message. The weight is obtained through the Weight class (Core ontology) and connected to the WeightedInterest by means of the "hasWeight" object property. Moreover, each message category may define some data property to better describe the message category, e.g. the temperature range that defines an environment comfortable. Examples of such classification may include the messages related to the house consumption, plants usage or alarms.

[14]http://purl.org/ontology/wi/core

Chapter 5

Notont-based Intelligent Notification System

5.1 Introduction

Notont provides a formal data representation for information related to a smart environment. Data models, defined accordingly with the ontology specification, contain information about the smart environment, users and devices defining the context.

Notont is exploited by a software, the Notont-based Intelligent Notification System (NINS), that generates and delivers context based messages to end user devices. House related information are used to interface the smart environment and get data accordingly to the specific message to generate; the message category is used to specify the interested data. Moreover, it is used to find out the interested users that have to be reached through the most suitable device. Devices are selected by means of the information about them. The most important data used by NINS are presented on figure 5.1. NINS architecture is represented in figure 5.2. The software is made up by four main components, each one with a specific role in the message delivery process:

- Notont Manager: takes care of the data model management providing action such as insert, update, delete and query upon the model;
- House Message Manager: encompasses the gathering of information from the smart home to generate messages;
- Device Manager: is in charge of the message dispatching toward the selected user device;

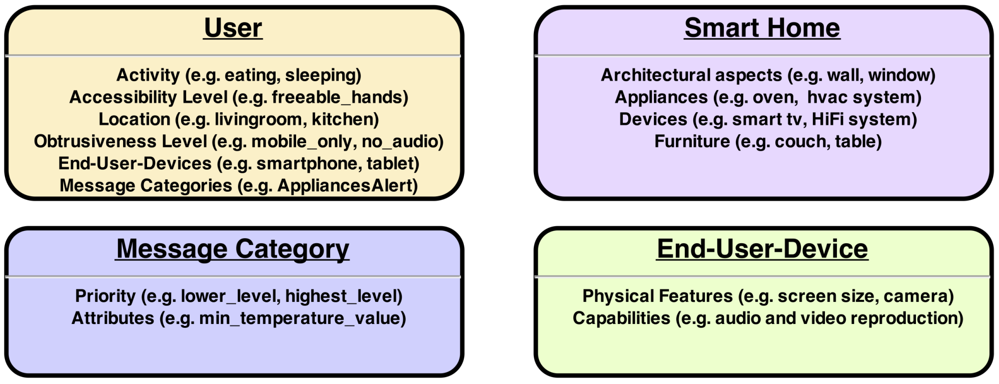

User
Activity (e.g. eating, sleeping)
Accessibility Level (e.g. freeable_hands)
Location (e.g. livingroom, kitchen)
Obtrusiveness Level (e.g. mobile_only, no_audio)
End-User-Devices (e.g. smartphone, tablet)
Message Categories (e.g. AppliancesAlert)
Smart Home
Architectural aspects (e.g. wall, window)
Appliances (e.g. oven, hvac system)
Devices (e.g. smart tv, HiFi system)
Furniture (e.g. couch, table)
Message Category
Priority (e.g. lower_level, highest_level)
Attributes (e.g. min_temperature_value)
End-User-Device
Physical Features (e.g. screen size, camera)
Capabilities (e.g. audio and video reproduction)

Figure 5.1. Most important information used in NINS

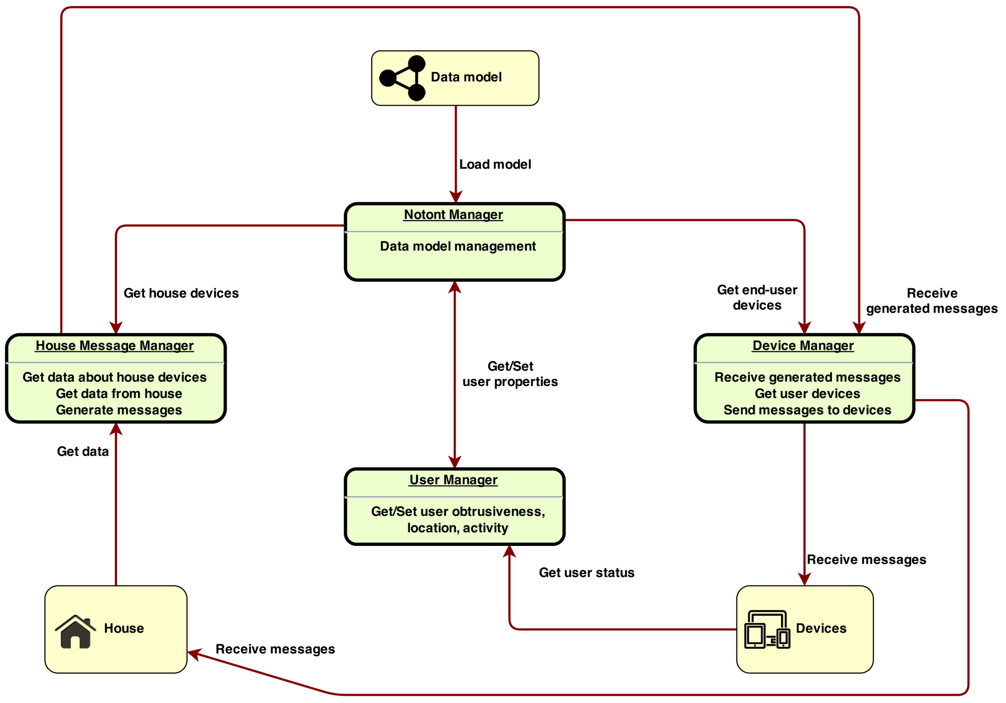


Figure 5.2. NINS architecture

- The User Manager takes care of updating user state, such as activities, locations or obtrusiveness level.

5.2 Notont Manager

The main NINS component is Notont Manager, in charge of the ontology and data model management. This is achieved by means of the OWL (Web Ontology Language) API that provides classes and methods to load and save OWL files, to query and manipulate OWL data models and to perform reasoning based on Description Logic engines. To infer information, Notont Manager relies on a reasoner, able to deeply explore the model and find not explicitly defined information. In this work, we used HermiT, a reasoner that can determine whether or not the ontology is consistent and identify subsumption relationships between classes. The choice of HermiT among several semantic reasoners is due to its good performances in checking ontologies consistency and reason on them, even with complex ontologies, like Notont is.

Notont Manager implements the needed functionalities to obtain information, set new data, update or delete old ones from the model: for instance, it is possible to get all users, their devices, or the location where a user is. It is possible to update the user location or its obtrusiveness level, delete a message category or define a new one. Moreover, it is in charge of verifying that the model definition is consistent, that is each definition of individuals is compliant with the Notont definition. For example, define an individual of the class Person and assign it more than one instance of the Obtrusiveness would lead to an inconsistency and such model wouldn't be used in the application.

When the application starts, an instance of Notont Manager is created. As first operation, Notont Ontology references are loaded: an XML configuration file contains the mapping between the IRI definition of Notont (with its imported ontologies) and the URL of the resource (such file or on-line resource). Notont Manager loads the model whose URL is given as parameter. Then, the model is checked for consistency. If the model is consistent, the Manager is ready to reply to incoming data requests.

One of the most important role of Notont Manager is the inferring of end user devices for each incoming message to dispatch. When a message is generated and the model is queried for recipient devices, Notont Manager first search for interested users: it get all the users and filters out those that aren't interested to that message category. Then, user obtrusiveness level is used to understand if a user is reachable somehow. For reachable users, their activity and accessibility level are used to understand if the user can interact with some device while performing the activity. User and house devices that respect the previews constraints are gathered and information about them are used to select the most suitable one that will receive the incoming message.

5.3 House Message Manager

House message manager is in charge of interfacing the smart home, retrieving data and generating messages. The model contains all the information about house, appliances, their location and their functionalities, needed for the message generation.

House Message Manager is made up by two submodules: the first has the role of generating messages starting from smart home data and information about message categories. The second sub-module has the role of interfacing the smart home and get needed data for message generation.

5.3.1 Message Generation

Generated messages belong to one of the defined message categories. Each category specifies a name, a priority and can have some data property defined. When the application starts, the House Message Manager gets the message categories and, in priority order, iterates through all of them recursively. For each category, needed data is retrieved form the smart home and, if the conditions are satisfied, a message is generated. For example, a message category may be “PowerOverThreshold” indicating that power usage exceeded the defined threshold. To generate this message, the House Message Manager get the the max power value associated to the message category by means of a data property. Then it gets the overall power from the smart home and, if it exceeds the defined value, a message is generated. Information about smart plugs are defined in the model: each one will be an instance of a Dog class; in this example a subclass of PowerMeter such as EnergyAndPowerMeter. Generated messages are sent to the Device Manager, that will provide to the dispatching.

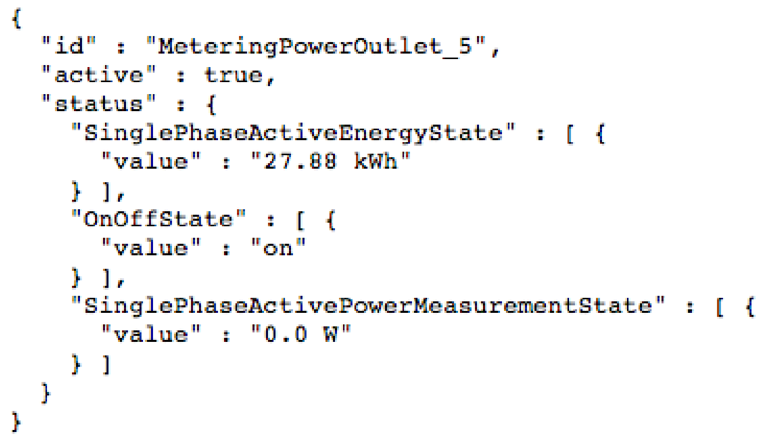
5.3.2 Smart House Interfacing

The communication between NINS and the smart house is technology dependent. For each smart home, the type of communication may change, so this module has to be adapted to the used technology. For the purpose of this work, the used smart home is recreated in the e-Lite research group laboratory. This smart environment is controlled by means of Dog (see 3.4.1), so the developed interface between NINS and Dog is presented.

Dog makes available information about the smart home through REST API and data are transferred as JSON objects or XML files. Information about devices are available through an XML file at the URL “<address>/api/devices”. Information retrieved from this URL have to be compliant with the ones defined in the data model; this file is used to update the model, for example to update a device location. Each device is available at the URL “<address>/api/devices/<deviceName>/”.

The device name is maintained in the data model and it possible to access its state through a JSON object at the URL "<address>/api/devices/<deviceName>/status".

In example showed before, to get the total power, plugs state are gathered as the one in figure 5.3, where the plug state shows a power value equal to zero.



```
{
  "id" : "MeteringPowerOutlet_5",
  "active" : true,
  "status" : {
    "SinglePhaseActiveEnergyState" : [ {
      "value" : "27.88 kWh"
    } ],
    "OnOffState" : [ {
      "value" : "on"
    } ],
    "SinglePhaseActivePowerMeasurementState" : [ {
      "value" : "0.0 W"
    } ]
  }
}
```

Figure 5.3. State of the meteringPowerOutlet5

5.4 Device Manager

Device manager is in charge of sending generated messages to the end user devices. When a message is received from the House Message Manager, Device Manager queries the model by means of Notont Manager and get the list of user devices to deliver the message.

For the purpose of this work, devices were based on Android operating system. Communication between Device Manager and devices is performed by means of GCM (Google Cloud Messaging) Service, the google service to send notifications to and Android-based devices. Device Manager contacts the GCM server and send it the message to be delivered. The server will notify the message to the device (figure 5.4).

5.5 User Manager

User Manager is in charge of the user-to-system communication to update user related information. This module gather data from the user regarding its current

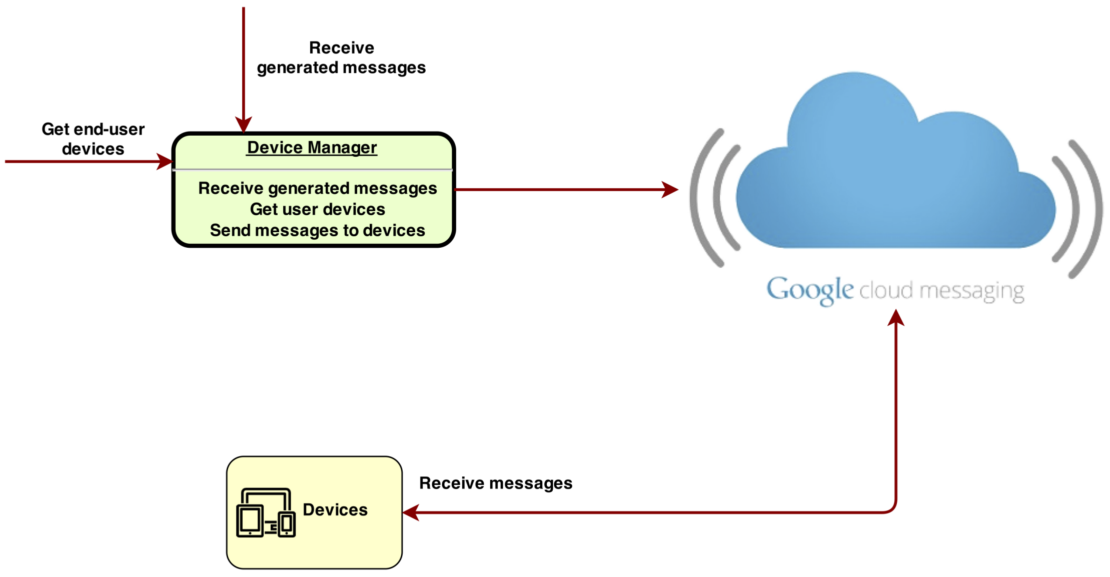


Figure 5.4. GCM usage for device reaching

activity, location or obtrusiveness level and propagate such data to the Notont Manager that will update the data model. Users communicate with the system through the mobile application installed in their mobile devices. As already explained in section 3.2, the gathering of information such as user current activity or location requires an effort to be completed that go beyond the purpose of this work. Assumptions about the knowledge of the user current activity and location lead to a simple User Manager whose aim is getting information about users directly from them.

5.6 The test case scenario

To try both ontology and application, a test case scenario was developed. It includes information about users, their devices, the smart environment and the type of messages that users are interested in, as defined in the section 4.3. Information is modelled as instances of Notont classes and stored in a file that is loaded by the NINS to manage the message generation. Moreover, selected devices have to be prepared, installing the mobile application to receive data.

The data model is composed by:

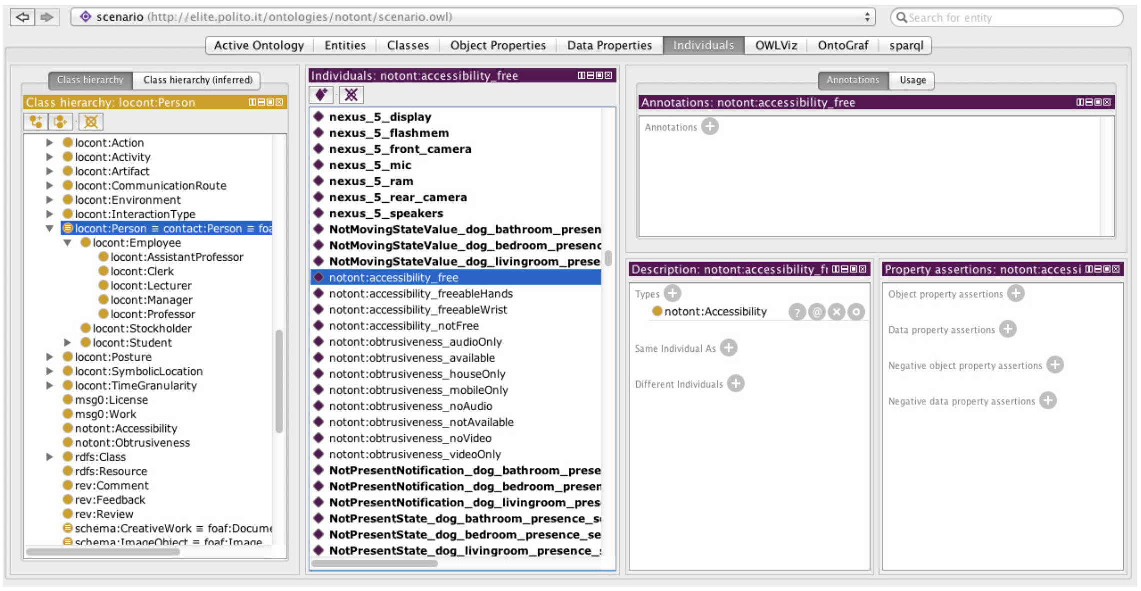
- two users: Luca, a 28 years old manager and Giulia, a 27 years old employee;
- three devices: samsung galaxy nexus, lg Nexus 5, samsung tab 10”;

- six message categories: EnvironmentalComfort, PowerOverThreshold, EnergyManagement, AppliancesUsage, AppliancesAlert, SecurityAlert;
- a smart environment made up by: temperature sensor, humidity sensor, three smart plugs, window sensor;

Luca is the owner of the LG Nexus 5 and he is interested in receiving messages about PowerOverThreshold, EnergyManagement, AppliancesAlert and SecurityAlert.

Giulia owns the other two devices and she is interested in receiving messages about EnvironmentalComfort, AppliancesUsage, AppliancesAlert and SecurityAlert.

The scenario modelling, as well as the ontology modelling is made by means of a software, Protégé[1]. This software, developed by the Stanford University, uses the OWL API to provide a GUI-based application for the ontology modelling and management (figure 5.5).


scenario (http://elite.polito.it/ontologies/notont/scenario.owl)
Search for entity
Active Ontology
Entities
Classes
Object Properties
Data Properties
Individuals
OWLViz
OntoGraf
sparql
Class hierarchy
Class hierarchy (inferred)
Class hierarchy: locont:Person
Individuals: notont:accessibility_free
Annotations
Usage
Annotations: notont:accessibility_free
Annotations
Description: notont:accessibility_fi
Types
notont:Accessibility
Same Individual As
Different Individuals
Property assertions: notont:accessi
Object property assertions
Data property assertions
Negative object property assertions
Negative data property assertions

```
locont:Action
locont:Activity
locont:Artifact
locont:CommunicationRoute
locont:Environment
locont:InteractionType
locont:Person ≡ contact:Person ≡ foa
  locont:Employee
    locont:AssistantProfessor
    locont:Clerk
    locont:Lecturer
    locont:Manager
    locont:Professor
  locont:Stockholder
  locont:Student
locont:Posture
locont:SymbolicLocation
locont:TimeGranularity
msg0:License
msg0:Work
notont:Accessibility
notont:Obtrusiveness
rdfs:Class
rdfs:Resource
rev:Comment
rev:Feedback
rev:Review
schema:CreativeWork ≡ foaf:Docume
schema:ImageObject ≡ foaf:Image
```

```
nexus_5_display
nexus_5_flashmem
nexus_5_front_camera
nexus_5_mic
nexus_5_ram
nexus_5_rear_camera
nexus_5_speakers
NotMovingStateValue_dog_bathroom_presen
NotMovingStateValue_dog_bedroom_presenc
NotMovingStateValue_dog_livingroom_prese
notont:accessibility_free
notont:accessibility_freeableHands
notont:accessibility_freeableWrist
notont:accessibility_notFree
notont:obtrusiveness_audioOnly
notont:obtrusiveness_available
notont:obtrusiveness_houseOnly
notont:obtrusiveness_mobileOnly
notont:obtrusiveness_noAudio
notont:obtrusiveness_notAvailable
notont:obtrusiveness_noVideo
notont:obtrusiveness_videoOnly
NotPresentNotification_dog_bathroom_prese
NotPresentNotification_dog_bedroom_presen
NotPresentNotification_dog_livingroom_pres
NotPresentState_dog_bathroom_presence_s
NotPresentState_dog_bedroom_presence_se
NotPresentState_dog_livingroom_presence_
```

Figure 5.5. Protégé application GUI

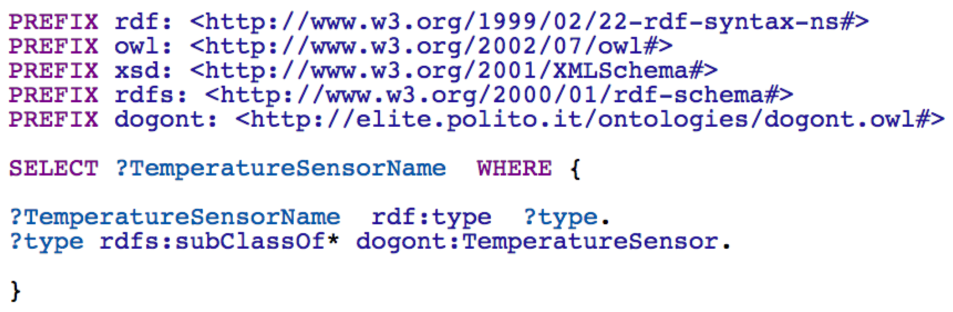
Protégé is developed in Java and any developer may implement and install his own plugins to improve the software functionalities. By using Protégé, it was possible to create all the individuals related to users or devices, connect them and check the consistency of the data model before use it. Check consistency and reasoning

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

upon ontologies is possible with protégé as well, since it includes reasoners like HermiT or FaCT++. The data model is stored as a unique .owl file: such file also includes the import of the Notont ontology.

The modelled scenario is given as input to NINS that manages and queries it to generate messages.

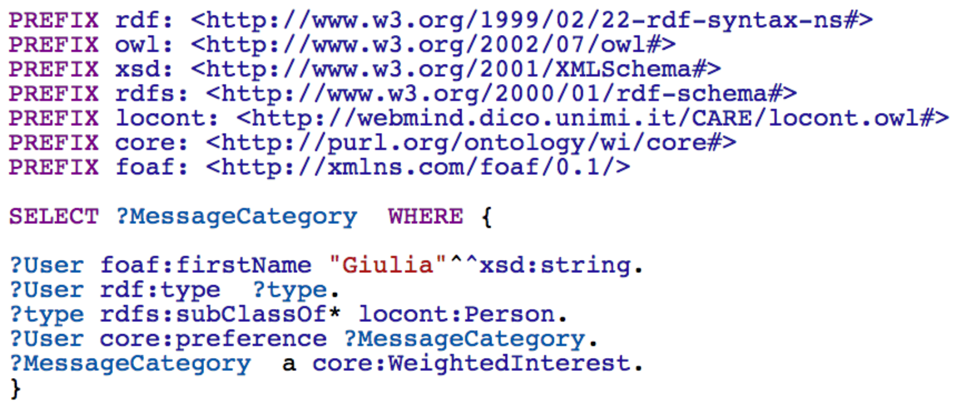
An example is given by the EnvironmentalComfort message generation; the House Message Manager, through the Notont Manager, search for temperature and humidity sensors and gets their name from the model. Figure 5.6 shows the SPARQL query to get the temperature sensor.



```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dogont: <http://elite.polito.it/ontologies/dogont.owl#>

SELECT ?TemperatureSensorName  WHERE {

?TemperatureSensorName  rdf:type  ?type.
?type rdfs:subClassOf* dogont:TemperatureSensor.

}
```

Figure 5.6. SPARQL query to get temperature sensor

Then it connects to the Dog and retrieves data about temperature and humidity. Minimum and maximum values are specified in data properties related to message category. The system keeps polling each 5 seconds checking that environmental variables are in the proper range. If temperature or humidity values exceeds the fixed ones, a message is generated and sent to the Device Manager. Figure 5.7 shows the query to get the message categories for the user named “Giulia”.



```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX locont: <http://webmind.dico.unimi.it/CARE/locont.owl#>
PREFIX core: <http://purl.org/ontology/wi/core#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?MessageCategory  WHERE {

?User foaf:firstName "Giulia"^^xsd:string.
?User rdf:type  ?type.
?type rdfs:subClassOf* locont:Person.
?User core:preference ?MessageCategory.
?MessageCategory  a core:WeightedInterest.
}
```

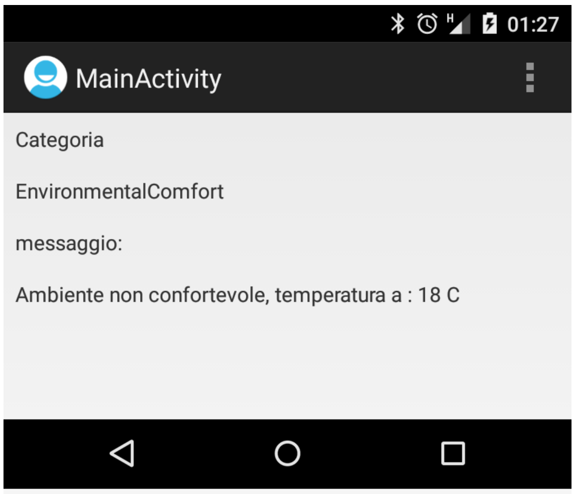
Figure 5.7. SPARQL query to get user message categories

The Device manager uses the message category and get the proper devices description from the Notont Manager. Then it creates a JSON object containing the message category and the message text (figure 5.8).

```
{
"message":"Ambiente non confortevole,temperatura a : 18 °C",
"category":"EnvironmentalComfort"
}
```

Figure 5.8. Message JSON Object

This message is sent to all the devices through the GCM Service. Each device will receive the message and will show it through the installed mobile application (fig 5.9).


01:27

MainActivity

Categoria

EnvironmentalComfort

messaggio:

Ambiente non confortevole, temperatura a : 18 C

Figure 5.9. Message visualization on Android-based devices

This process is repeated for all the message categories and reiterated continuously.

Chapter 6

Conclusions and future works

6.1 Discussion

The field of Smart Environment is rapidly evolving toward a reality that could pervade and influence our daily life, helping people in their activities, supporting the ones with disabilities and making people more aware of their surrounding.

We explored how smart environments and in particular smart homes are contributing in improving the life of their inhabitant by means of automation and intelligent systems able to autonomously manage house plants, optimizing their usage and communicating about their states in a more active way. We saw how smart environments are able to adapt their behaviour accordingly with the user needs and the surrounding context to provides a better place to live in. Researches in the last few years has certainly matured smart environment technology beyond their deployment in experimental situations. Researches encompass not only the supporting technologies areas such as physical components and middleware, but also the modelling and decision making capabilities of entire automated environments, leading to more intelligent systems.

We saw how different data modelling approaches are possible and we focused in particular on the usage of ontologies for context modelling. Ontologies proved to be an interesting solution for context modelling thanks to their capabilities to describe general concepts, connecting and making assumption on them. The availability of several ontologies for context modelling in smart environments shows how the research on this field is trying to properly use ontologies to support making decision processes, infer user activities and provide context aware services.

Focusing on the system-to-user interaction, we proposed a new intelligent notification system for smart homes, able to take into account users, devices and the surrounding environment to deliver house related messages to the most suitable devices. The work was divided in two steps: the realization of the data model infrastructure and the making of a software for its management and for the message delivery.

The first step was completed realizing an ontology, called Notont. This step was the fundamental part of this thesis and we focused on four main entities: User, Device, Message Category and Smart Home. To realize Notont, several already defined ontology for context modelling were explored. Among the multitude of available ontologies, we chose four of them as the base of Notont. The ontology was refined adding missing classes, equivalences statements and restriction on classes to complete the modelling infrastructure. The ontology is able to model information about the users, their activity, location, their availability in interacting with device while performing an activity (Accessibility) and their availability in receiving messages through some channels (Obtrusiveness). Devices are modelled by means of their physical features and the smart home is defined in terms of architectural description, its appliances with their functionalities and state. These information are used to evaluate the user state and infer which is the most appropriate device to use in the message delivery.

The second step implied the development of a software able to manage the ontology and use it to generate and deliver context based messages. The software is made of four main modules: Notont Manager, User Manager, House Message Manager and Device Manager. The Notont Manager is in charge of managing the ontology providing the needed functionalities to operate upon the model (query, insert, update and delete). The User Manager encompasses the update of the user state such as the current location. The House Message Manager is in charge of interfacing the smart home to retrieve the needed data for the message generation. Data changes with respect to the type of message to generate and may vary from the current temperature to the state of window sensors. By using such data, the House Message Manager generate messages to be delivered to users. The Device manager is in charge of the message delivery to the end user devices. Devices receive messages by means of an installed mobile application.

6.2 Preliminary results

The implemented solution was tested on a scenario test case. The e-Lite laboratory was used where a smart home is recreated. Information about users, devices and information about the smart environment was modelled and put in a data model to test overall system behaviour. The environment was alterated creating conditions for messages generation: plugs were connected to heavy loads such as heater or irons, temperature sensors where moved to heat sources and humidity sensors to near steam sources. Stimuli were captured by sensors so that the House Message

Manager could get them and generate the proper message. Messages were finally delivered to end user devices.

The datamodel management, due to the usage of the reasoner, heavily impacts on software performances. Starting from a message generated by the House Message Manager, the system infers the best end users device in a mean time of 3.1 s (SD 2.2 s). The overall delivery process, from the message generation to its reception on the end users device, takes a mean time of 5.7 s (SD 3.3 s).

6.3 Future work

The implemented software has the main role of testing the Notont ontology capabilities in providing a complete data representation for context based message generation. The prototype could be improved adding a higher number of message category to be generated. Moreover, the usage of the reasoner could be optimized reducing the overall execution time. The ontology may be improved removing useless classes and data properties derived from indirect ontology imports. A reduced number of classes would affect system performances as well, since the reasoning process upon the model would work on a reduced set of assertions. The number and type of devices could be incremented and the overall system could be tried by candidates to get important feedbacks from final users.

One of the most important future work regards the usage of Notont not only to infer the most suitable end user device, but also to infer the way a message is delivered. Information modelled in Notont enables us to explore the context and understand if a message has to be delivered, for example, via audio message or video message. This improvement doesn't requires any change in the ontology; it requires an improved management of the data model to take care of such information.

Bibliography

[1] Juan Carlos Augusto and Paul McCullagh, *Ambient intelligence: Concepts and applications*, **4**, no. 1, 1–26.

[2] Dario Bonino and Fulvio Corno, *Dogont-ontology modeling for intelligent domotic environments*, Springer.

[3] ———, *What would you ask to your home if it were intelligent? exploring user expectations about next-generation homes*, **3**, no. 2, 111–126.

[4] Janez Brank, Marko Grobelnik, and Dunja Mladenić, *A survey of ontology evaluation techniques.*

[5] Kevin Brooks, *The context quintet: narrative elements applied to context awareness*, Human Computer Interaction International Proceedings, vol. 2003.

[6] Harry Chen, Tim Finin, and Anupam Joshi, *An ontology for context-aware pervasive computing environments*, **18**, no. 3, 197–207.

[7] ———, *The SOUPA ontology for pervasive computing*, Ontologies for agents: Theory and experiences, Springer, pp. 233–258.

[8] Diane J. Cook and Sajal K. Das, *How smart are our environments? an updated look at the state of the art*, **3**, no. 2, 53–73.

[9] Luigi De Russis, *Interacting with smart environments: Users, interfaces, and devices.*

[10] Anind K. Dey, *Understanding and using context*, **5**, no. 1, 4–7.

[11] Colin Dixon, Ratul Mahajan, Sharad Agarwal, AJ Bernheim Brush, Bongshin Lee, Stefan Saroiu, and Paramvir Bahl, *An operating system for the home.*, NSDI, vol. 12, pp. 337–352.

[12] David Fono and Roel Vertegaal, *EyeWindows: evaluation of eye-controlled zooming windows for focus selection*, Proceedings of the SIGCHI conference on Human factors in computing systems, ACM, pp. 151–160.

[13] Thomas R. Gruber, *Toward principles for the design of ontologies used for knowledge sharing?*, **43**, no. 5, 907–928.

[14] Tao Gu, Xiao Hang Wang, Hung Keng Pung, and Da Qing Zhang, *An ontology-based context model in intelligent environments*, Proceedings of communication networks and distributed systems modeling and simulation conference, vol. 2004, pp. 270–275.

[15] Ian Horrocks, Peter F. Patel-Schneider, and Frank Van Harmelen, *From SHIQ and RDF to OWL: The making of a web ontology language*, **1**, no. 1, 7–26.

[16] Oguz Icoglu, KlausA. Brunner, Ardeshir Mahdavi, and Georg Suter, *A distributed location sensing platform for dynamic building models*, **3295** (2004), 124–135 (English).

[17] Kin Fun Li, *Smart home technology for telemedicine and emergency management*, **4**, no. 5, 535–546.

[18] Nan Li and Burcin Becerik-Gerber, *Performance-based evaluation of RFID-based indoor location sensing solutions for the built environment*, **25**, no. 3, 535–546.

[19] Paul P. Maglio, Rob Barrett, Christopher S. Campbell, and Ted Selker, *SUITOR: An attentive information system*, Proceedings of the 5th international conference on Intelligent user interfaces, ACM, pp. 169–176.

[20] Eladio Martin, Oriol Vinyals, Gerald Friedland, and Ruzena Bajcsy, *Precise indoor localization using smart phones*, Proceedings of the International Conference on Multimedia (New York, NY, USA), MM '10, ACM, 2010, pp. 787–790.

[21] Hamid Medjahed, Dan Istrate, Jérôme Boudy, J.-L. Baldinger, and Bernadette Dorizzi, *A pervasive multi-sensor data fusion for smart home healthcare monitoring*, Fuzzy Systems (FUZZ), 2011 IEEE International Conference on, IEEE, pp. 1466–1473.

[22] Heiko Paulheim, *Ontology-based application integration*, Springer New York.

[23] Davy Preuveneers, Jan Van den Bergh, Dennis Wagelaar, Andy Georges, Peter Rigole, Tim Clerckx, Yolande Berbers, Karin Coninx, Viviane Jonckers, and Koen De Bosschere, *Towards an extensible context ontology for ambient intelligence*, Ambient intelligence, Springer, pp. 148–159.

[24] Albrecht Schmidt, *Interactive context-aware systems interacting with ambient intelligence*, 159–178.

[25] Jeffrey S. Shell, Roel Vertegaal, and Alexander W. Skaburskis, *Eyepliances: Attention-seeking devices that respond to visual attention*, CHI '03 Extended Abstracts on Human Factors in Computing Systems (New York, NY, USA), CHI EA '03, ACM, 2003, pp. 770–771.

[26] Lorenzo Sommaruga, Antonio Perri, and Francesco Furfari, *DomoML-env: an ontology for human home interaction.*, SWAP, vol. 166, Citeseer.

[27] Roel Vertegaal, *Attentive user interfaces*, **46**, no. 3, 30–33.

[28] Roel Vertegaal, Jeffrey S. Shell, Daniel Chen, and Aadil Mamuji, *Designing for augmented attention: Towards a framework for attentive user interfaces*, **22**, no. 4, 771–789.

[29] 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, 2004. Proceedings of the Second IEEE Annual Conference on, Ieee, pp. 18–22.

[30] Florian Weingarten, Marco Blumendorf, and Sahin Albayrak, *Towards multimodal interaction in smart home environments: the home operating system*, Proceedings of the 8th ACM Conference on Designing Interactive Systems, ACM, pp. 430–433.

[31] Rayoung Yang and Mark W. Newman, *Learning from a learning thermostat: lessons for intelligent systems for the home*, ACM Press, p. 93.

Ringraziamenti

Vorrei anzitutto ringraziare i miei relatori, Fulvio Corno, Dario Bonino e Luigi De Russis per la loro ammirevole guida in questo percorso di tesi. Grazie a loro, ho avuto la possibilità di approfondire un argomento al quale sono legato da molto, acquisendo conoscenze e competenze che spero di approfondire nel prossimo futuro.

Un profondo e sentito ringraziamento va alla mia famiglia e alla mia fidanzata Jessica per il continuo e incondizionato supporto che mi ha dato la forza di affrontare le difficoltà incontrate in questo percorso ricco di avventurosi ostacoli, non sempre facili da superare. E' solo grazie a voi se oggi posso scrivere (di nuovo) queste poche righe.

Un grande ringraziamento a tutti gli amici, vicini e lontani, con i quali ho condiviso questa bellissimima esperienza. Amici entrati a far parte della mia vita e con i quali spero di condividere ancora tante altre avventure. Amici storici come il gruppo di PB e il fratello acquisito Fabrizio, o gli amici dell'indimenticabile CappelVerde (e relativi acquisiti) con cui ho mosso i primi passi qui a Torino. Tra queste non posso non menzionare Michela, sfortunata amica e terapista acquisita. Ringrazio il gruppo di "astrofisici ma non solo", Daria, Lorenzo e Sabrina e le loro improbabili discussioni che continuano ad alimentare l'accesa diatriba tra fisici e ingegneri.

Non posso non ringraziare tutto il team Sunslice e i suoi ingegneri e architetti, con i quali ho vissuto bellissime esperienze. Citarvi tutti sarebbe lungo e tedioso, per cui ve lo risparmio.

Vorrei concludere questa pagina ringraziando ancora una volta chi, con la sua mancanza, ha alimentato ambizioni e obiettivi, dandomi la forza di non arrendermi e di affrontare ogni avversità a testa alta. Grazie papà.