Semantic Modelling of Human Behaviour in Smart Spaces: A case study on public buildings and office domain

Natalia Díaz Rodríguez

Master on Soft Computing and Intelligent Systems -Master's Thesis-

Department of Computer Science and Artificial Intelligence
University of Granada (Spain)

Department of Information Technologies
Åbo Akademi University (Turku, Finland)

July 2012

Directors:
Manuel PEGALAJAR CUÉLLAR (University of Granada)
Miguel DELGADO CALVO-FLORES (University of Granada)
Johan LILIUS (Åbo Akademi University)

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this work we study systems that allow detection and recognition of human activities. We present a model to represent human behaviour in Smart Spaces with semantics; more concretely, with ontologies. The domain is focused within a working environment, such as the office domain. This model will be used to learn, recognize and monitor behaviours, and will be simple enough for those behaviours to be easily programmed not only by a programmer, but also by a non expert. In order to achieve this, the model is graphical, with the aim of equipping it with a broader usability in the end-user application. Context-aware reasoning allows to infer novel information from atomic data. The proposal also contains mechanisms to face the uncertainty that the environment around the user may encounter. Our research topic falls under the context of Ambient Intelligence (AmI), which plays an essential role on integration and treatment of information from sensors that describe the user activity in the environment for a later transfer and processing, in real-time, by Artificial Intelligence (AI) models. The final aim consists of tracking humans and events or behaviours, such as their location or activity, to generate reminders or alarms reacting to, for instance, forgotten actions or potentially hazardous situations.

1.1 Ubiquitous and Pervasive Computing

The idea of ubiquitous space was proposed as an ideal world where humans and surrounding devices interact effortlessly. People would be surrounded by intelligent intuitive interfaces embedded in all kinds of objects and the environment would be capable of recognizing and responding to the presence of different individuals in a seamless, unobtrusive and often invisible way [6]. Thus, a transparent technology in the environment facilitates humans an easier everyday life [117, 74]. According to his father Mark Weiser [145], Ubiquitous or Pervasive Computing is the method of enhancing computer use by making many computers available throughout the physical environment, but making them effectively invisible to the user.

1.2 Ambient Intelligence

The term of Ambient Intelligence (AmI) was introduced by the European Commission in 2001 [67] and originated, in part, from the technology evolution, computer costs reduction and computer networks development [118, 34]. In the decade of the 80s the personal computer revolution turned the ratio from one computer per group of expert people to one user per computer. Furthermore, multiple objects and tools of everyday life integrate embedded microprocessors (domestic appliances, vehicles, mobile phones, etc.). One of the latest definitions [16] describes AmI as digital and proactive environments, with capacity to sense the environment and assist users in their daily lives. However, the concept of Ambient Intelligence [6, 74] in Information Society goes a step beyond Ubiquitous Computing emphasizing on greater user-friendliness, user-empowerment, human assistance [109] and easy interaction with efficient services.

From the previous definitions we can intuit that AmI involves the integration of diverse areas of knowledge within Computer Science, related with the environment perception, the actuation with the environment, reasoning on the perceived information, human computer interaction (HCI) and management of privacy and security.

In the context of perception of the environment, AmI systems tend to use different sensors to perceive the environment [118]: pressure, position, movement, distance, light and sound sensors, image, RFID, etc. In order to act in the ambient, AmI considers actuators to manage the light, window blinds, locks, heating, air conditioning, electricity, etc. The type of actuators depends in great part of the type of implemented application.

Another key aspect to consider in AmI is reasoning and treatment of information, needed to monitor and detect the user's actions. Therefore, it is essential to develop human activity models for prediction and recognition, connecting the semantics of the user activities with the signals obtained from the sensors. In this line, the most used models in the literature are Hidden Markov Models [134, 23, 97], Naive Bayes classifiers [24], Bayesian Networks [141, 138], data mining [70] or rule based systems [9]. These activity models need to be integrated with inference methods to allow classification and inferring of activities from a data stream, provided by sensors. Temporal logic and temporal clustering appear at this point as a possible alternative for this aim [12, 69]. However, the complexity and the quantity of possible complex behaviours, the temporal interdependences among actions, the relevance of a behaviour and its associated semantics make these tasks non trivial and clear challenges in AmI research. Nevertheless, simple proposals on taxonomies [1] of activities in Ambient Assisted Living exist [69] for temporal relations representation within the *CASAS* project, as figure 1.1 shows.

Another challenge in AmI is the modelling and detection of behaviours

[1]A taxonomy is a hierarchical structure or classification scheme that allows to characterize a domain.

when there exists more than one user in the controlled environment, since it is often complex to infer which user is realizing each action, without using invasive solutions, such as video cameras. In these cases there also exist some proposals with Hidden Markov Models to detect users and to model shared activities [134, 35].

Despite the short existence and the current open problems, the application areas of AmI are very extensive: smart homes, transport, emergency services, education, work spaces and eHealth are some of the examples. Next, we proceed to detail these domains by separate.

In relation with smart homes, the main objectives have been security at home, comfort or energy efficiency. There exist a large variety of international projects that study these aspects, for example, [110, 81, 36, 60] among others. All these proposals focus on the detection of activities of daily living, with emphasis on applications to improve the autonomy of the elderly or people with special needs. When it comes to transport, harbours, airports, stations and vehicles, these can be enhanced with technology capable of providing knowledge about the state of the environment in each moment. Based on this information, an AmI system can act in every instant to improve the security and efficiency of the system or augment the comfort of the user. Examples of projects in this line are I-VAIT [108], Nissan Cambridge Basic Research [101] or Microsoft's route planner [82, 68]. The first of these systems provides the driver information about different elements of the vehicle such as pressure in the breaks, takes into account the driver's mood and acts to prevent dangerous situations such as difficult manoeuvres. The second model predicts what actions the driver will perform to optimize the vehicle's performance in concrete situations, and warn the user over possible hazards. The third system provides route plans customized to each driver.

In the domain of emergency services, AmI systems can be useful to manage efficiently the traffic, detect risks and accidents, fires, etc. and elaborate plans for the service. An example is the project *e-Road* [40], where the access is prepared to security personnel when emergencies occur. Another example is the project *PRISMATICA* [142]. It uses cameras to monitor public transport locations and detect forbidden actions in traffic, intrusions, or to guarantee public transport security.

In the area of education, AmI is able to assist students in learning through PDAs and elements of human computer interaction (ambient noise, cameras, etc.). Some examples are preparing material automatically, promoting work in group, inferring future actions of the professor, etc. Projects related with AmI in education are [8, 133, 52, 150, 84]

Regarding AmI in offices or work environments, HCI applications focus on easing the work in group and optimize the office space. Examples in this line are the project *AIRE* [10], the interactive *iRoom* project at Stanford [51] and other similar ones [135, 90]. Other work environments such as industrial plants have been used as target of AmI, e.g. the *MOSES* system [136] localizes work staff and identifies the tasks they are doing at any moment, being able in this way, to advice in the remaining tasks to do and warn about potential oversights or forgotten actions. Another example is *iShopFloor* [132], a multi-agent architecture to plan and control industry processes.

In the eHealth area [73], the evolution of IT and medical technologies, research, culture and society have led health care systems to require new solutions for efficient, effective and accessible in any moment health care. EHealth can be considered as a broad term that, among others, includes telemedicine, and is one of the largest application areas of AmI systems. Knowledge-based recommendation systems, illness detection and diagnostic systems are some examples of applications.

As a consequence of the longer life expectancy, the need for social assistance at home, the ageing of the population and the reticence of the elderly to abandon their residence, **Ambient Assisted Living (AAL)** arises as the area of Ambient Intelligence applied to homes or elderly residences, with the aim of assisting the user in his daily tasks. Some international projects in this area are *CARE* (Context Awareness in Residence for Elders) [81], *CASAS* [114], *iDorm* [45] and *TigerPlace* [110], among others [46]. They focus on modelling, learning, detecting and recognizing daily activities, to identify anomalous behaviours and assist when needed. Application domains on eHealth and AmI go from individualized medicine to preventive health to improve life quality. Some use mobile assistants [28], such as diet personal managers [155], sport assistants [140] or rehabilitation systems [154]. Other area of eHealth and AmI uses sensor networks to monitor patients, e.g., video techniques to detect the emotional state of a patient [103] or sensors to obtain blood pressure, electrocardiograms and glucose levels [80]. The project *PERSONA* [13] studies diverse sensor types and how they can be used to monitor daily activities. [13] reviews different techniques for intercommunication and interoperability systems.

In spite of the great advances in AmI work, many open problems remain. One of them is knowing when, how and how often, an AmI system should communicate with the user to alert of dangerous situations, since it can become disturbing and non operative, if it happens too frequent. The distribution of sensors, its time of use and battery life are problems that require long time to design a solution for, depending on the application to deploy. In regard to the activities to monitor, a taxonomy of the types of activities that can be realized in each context is needed. Other issues are identifying user profiles, the model design when several people are involved and other ethical aspects, related with privacy and the reduction of social activity, implied by this kind of technology advances. In this project, we will focus on AmI applications on the domain of the work office and public buildings and develop a detection system independent of the sensors to use.

1.3 Smart Spaces

Within a ubiquitous space two issues are to be taken into consideration. The first one is the underlying infrastructure, i.e., the technology enabling interaction with the space. Secondly, from the user point of view, the user experience has to be considered, i.e. how the Smart Space technology can be put around to serve the user without demanding too much interaction from him, but helping in his daily activities. As an answer to the underlying infrastructure, the idea of **Smart Space** (SS) [131] comes up as a physical space which encapsulates its information allowing devices to join and leave the space. The Smart Space becomes a dynamic environment changing over time while entities interact with it, to share information between them. Application examples using this Smart Space paradigm can be seen in [75, 124] using M3 implementation [63]. The first work details the framework for Smart Space application development with an example in home automation. The second case applies Smart Space development tools for semantic programming of biomedical imaging applications.

1.3.1 Tagged World

A **Tagged World** [41, 129] is an example of Smart Space which is able to recognize user behaviours to warn when some action, belonging to certain behaviour, has been forgotten. The concept of Tagged World [5] is born from a project involving, among others, the University of Ritsumeikan, that makes users' lives safer and simpler by means of human behaviour reasoning. User interaction is made through a computer or Pocket Assistant with an RFID reader, which compares the RFID tag registries with the patterns that determine the human activities. A Bayesian network checks the probability of occurrence of a behaviour from the set of events that are produced during the studied time. A Tagged World tracks its entities (user, objects...) by sensors or RFID tags, assuming that in few time all objects will include a RFID tag. Works related to a Tagged World [41] show an inductive learning model whose input is a series of user actions (detected by sensors), and the output is a behaviour database containing detected user routines. The reasoning system uses the behaviour database to identify recognized behaviours and wrong actions producing, in the latter case, alarms to the user. In [149] the user can employ objects as switches of a touching interface of services offered by the Tagged World.

In relation with Smart Spaces, the umbrella term *everywhere computing* is the paradigm to which ubiquitous and pervasive computing are turning into [7], including mobile computing, sensor networks, HCI and artificial intelligence under pervasive computing. Since different scenarios and metrics are deployed within ubiquitous computing, the diversity of systems prevents the use of well-formed hierarchical classifications, making a consensus in evaluation tools difficult.

1.4 Context awareness

Context consists of any information that can be used to characterize the situation or state of an entity [43]. Entities can include a person, an object, an application or device that is used to interact with the user. In order to model human activity and behaviours of the Smart Space, context needs to be modelled. There are different ways of modelling context: key-value models, graphical models, object oriented or logic based models [17]. However, ontology-based context modelling overcomes other models' problems regarding simplicity, flexibility, extensibility, generality, expressiveness and automatic code generation [131]. An interoperability solution based on ontology model can benefit from ontology reasoning, since ontologies are the most promising and expressive models fulfilling requirements for modelling context information [17].

The survey [18] shows advanced ontology-based context models in a good compendium of design architectures, pros and cons. Examples of OWL-based approaches for context modelling are *CoBrA* and *SOCAM*. *CoBrA* [30] is an agent-based infrastructure for context modelling, reasoning and knowledge sharing using context acquisition components. *Soupa* and *CoBrA-Ont* Ontologies are some of the related tools. User's privacy control is also included. *SOCAM* (Service Oriented Context Aware Middleware)[55] introduces a server based architecture for building context-aware services focused on information sensing and context providers. Other project providing an OWL encoded context ontology (*CONON*) is [147]. Part of *CONON*, for the home domain can be seen in figure 1.2. *CoDAMos* (Context-Driven Adaptation of Mobile Services) [105] is another ontology, around four concepts: User, Environment, Platform and Service. An excerpt of the Environment ontology that serves us as inspiration can be seen in figure 1.3.

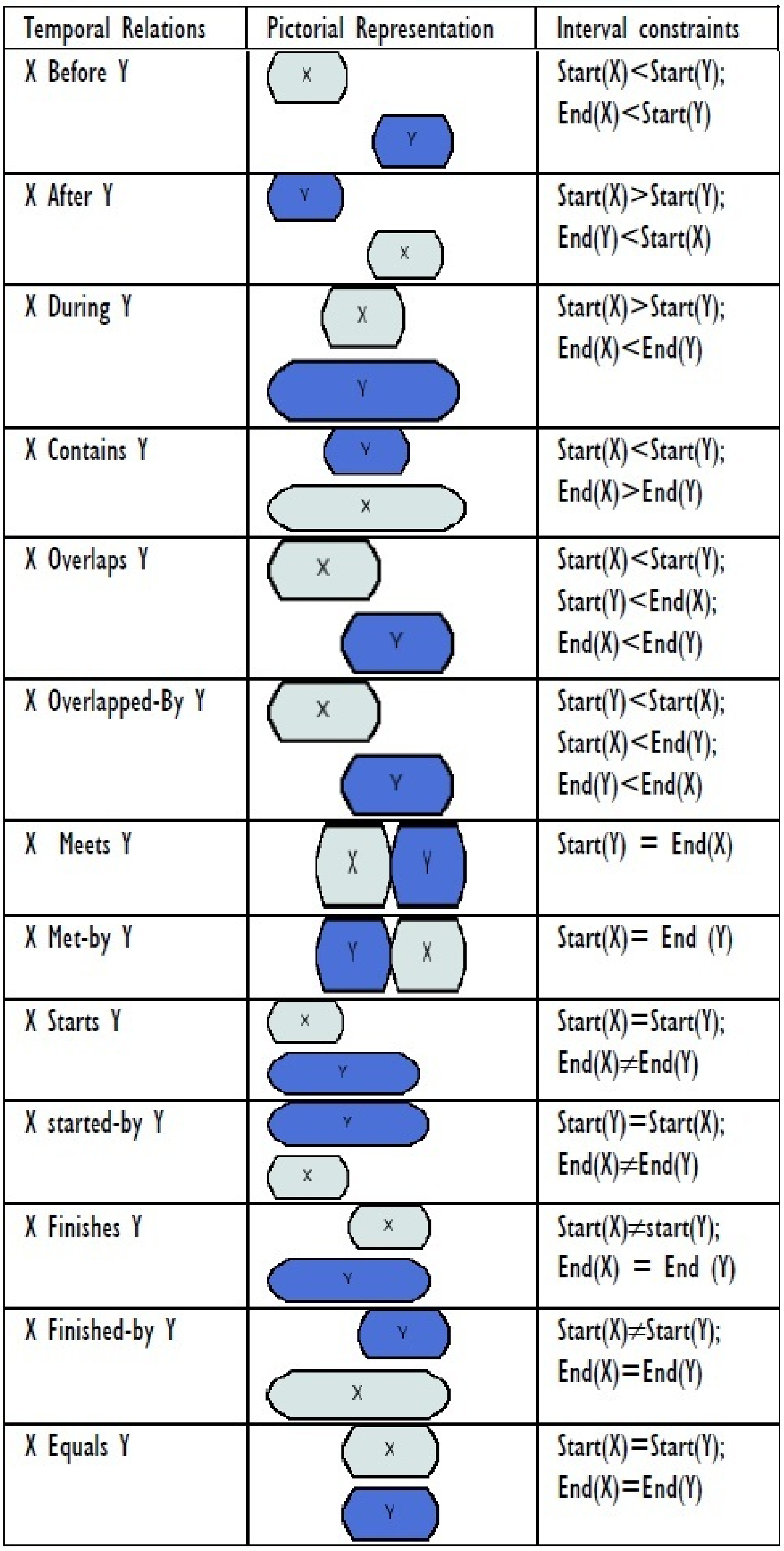
In *Gaia* [125], a metaoperative system is extended to include context-awareness. Instead of using RDF Triples, *Gaia* uses 4-ary predicates (being the 4th one context-type), first order logic and not OWL but DAML + OIL. Gaia's MVC model differs from other architectures, such as e.g., the blackboard architecture in Smart-M3 [63, 99]. Other architectures for smart phone context-aware frameworks can be found in [86, 152].

Context Toolkit [43] presents an approach to enable application development by using reusable components. However, a restriction in meaningfulness exists due to its attribute-value tuples, in contrast with RDF. Many systems use SQL as query language, which makes them unsupported for latest standards (i.e., SPARQL).

Other example is *HIPPIE* [100], which utilizes existing users' information to distribute context information to the users' devices. *NESSIE* [106] focuses, on the other hand, on event based awareness. In [19], the Context Aggregation and REasoning (CARE) middleware interacts with the COSAR [121] system to recognize human activities through hybrid ontological/statistical reasoners executed on personal mobile devices. They use a general human activity recognition ontology [2] and a smart-home 28 days dataset.

Summarizing, we can observe diverse frameworks to facilitate the creation of context-aware services. Our objectives try to go further, as it will be shown, in order to provide support for imprecision, as well as a functional programming tool on top of a context-aware activity representation framework. In the next chapter we proceed to further detail the state of the art on human activity recognition.

[2]PalSPOT project activity recognition ontology: http://everywarelab.di.unimi.it/palspot



Temporal Relations	Pictorial Representation	Interval constraints
X Before Y	X Y	Start(X)<Start(Y); End(X)<Start(Y)
X After Y	Y X	Start(X)>Start(Y); End(Y)<Start(X)
X During Y	X Y	Start(X)>Start(Y); End(X)<End(Y)
X Contains Y	Y X	Start(X)<Start(Y); End(X)>End(Y)
X Overlaps Y	X Y	Start(X)<Start(Y); Start(Y)<End(X); End(X)<End(Y)
X Overlapped-By Y	X Y	Start(Y)<Start(X); Start(X)<End(Y); End(Y)<End(X)
X Meets Y	X Y	Start(Y) = End(X)
X Met-by Y	Y X	Start(X)= End (Y)
X Starts Y	X Y	Start(X)=Start(Y); End(X)≠End(Y)
X started-by Y	Y X	Start(Y)=Start(X); End(X)≠End(Y)
X Finishes Y	X Y	Start(X)≠start(Y); End(X) = End (Y)
X Finished-by Y	Y X	Start(X)≠Start(Y); End(X)=End(Y)
X Equals Y	X Y	Start(X)=Start(Y); End(X)=End(Y)

Figure 1.1: Temporal relations representation for a multi-inhabitant smart environment, used in CASAS project [69]

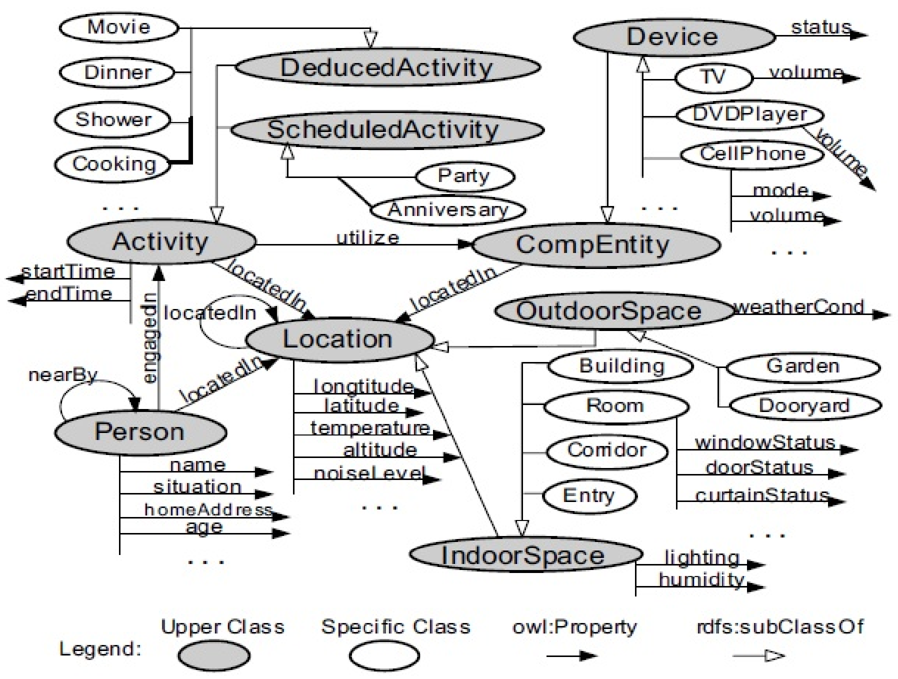


Figure 1.2: OWL encoded context ontology (CONON) [147] (partial)

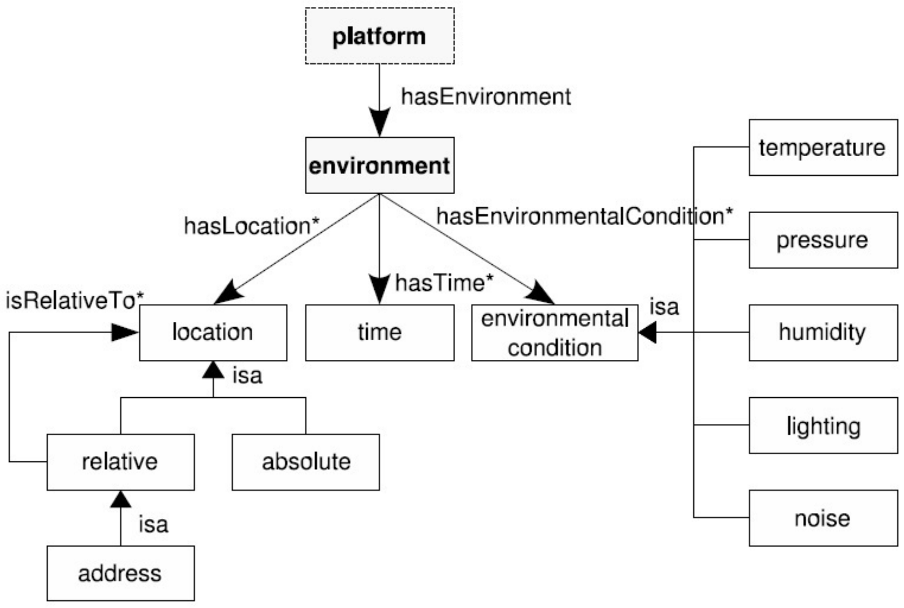


Figure 1.3: Environment ontology concepts within the CoDAMos ontology [105]

Chapter 2

State-of-the-Art

In this chapter we detail the state-of-the-art on modelling behaviour knowledge, current solutions and how they are developed. With special interest, we study existent proposals on modelling activity recognition and social interaction due to its relevance with respect to our approach.

2.1 Human activity monitoring

Human activity study is a key aspect in the development of AmI systems, since its intrinsic aim is oriented to user assistance. One of the areas in which complex behaviours are studied is in **Ambient Assisted Living**, where **Activities of Daily Living (ADL)** are modelled. Projects that stand out in this area are *TigerPlace* [110], *iDorm* [45, 34], *CASAS* [114, 134] or [102]. They aim at combining the elderly independence with their constant supervision and required assistance. The technologies employed to acquire information depend strongly on the objective and activity to monitor. *iDorm* and *CASAS* employ embedded sensors and consider the hypothesis that, it is not needed to identify the actor of each of the actions in real time, but learn the track of the behaviours through the detection of temporal moments and the areas where the actions happen [118, 134]. On the other hand, other research lines use wearable sensors, mobile devices, etc. [66, 151, 57]. [15] offers a broad study on sensor technologies on both approaches, embedded and wearable sensors. Finally, another path recently open is the use of non intrusive video sensors, such as Microsoft's device Kinect, for indoor identification and tracking of multiple persons. Promising results have been shown when following several persons at the same time [33], although there exist limitation when the number of users rises or the distance of the user to the sensor is superior to about 3 meters.

Technology advances on telecommunications, sensor networks and the miniaturization of electronic devices, have propitiated the monitoring of user activity at any time and place, with the aim of improving life quality. Accelerometer sensors are being currently studied in a broad number of studies [83, 107, 76, 47], e.g., in fall detection. The main advantages are the possibility of real time trajectory tracking and the 3D, lateral and vertical, signals. Common drawbacks are finding an adequate positioning of these sensors for a quality signal, the noise removal and proper feature selection. Alternative ways embed sensors on clothes, and more details can be found on [94, 49, 95, 38, 56, 29]. Another trend recently is the use of smart phones, equipped with accelerometer, gyroscope, magnetometer and GPS, to study motor and ergonomic variables. Some examples can be seen in [137, 87]. Some software architectures [71] propose WBAN (Wireless Body Area Networks) such as Bluetooth or ZigBee to transmit user information.

In the following section, concrete Artificial Intelligence proposals are studied for exhaustive learning and classification of behaviours.

2.2 Human activity learning and detection

Recognition of human activities and detection of anomalies, in support of the user, has been previously studied with different probabilistic models and AI techniques that try to abstract the concept of behaviour. Probabilistic systems provide great flexibility when controlling different alternatives in the realization of behaviours and when applying to different environments. Regarding probabilistic and statistical techniques, [79] collects a study on advantages and drawbacks on the use of stochastic techniques for human activity recognition. Although Bayesian Networks have been used in several cases [141, 138], the most used tool is Hidden Markov Models (HMM) [23, 97]. Even if probabilistic models are the most used models in the past, data mining techniques [70] or fuzzy logic [9] are also being developed nowadays.

Applications on different techniques can be seen on naîve Bayes classifiers [141], decision trees [91] or conditional random fields [139] (all tested on single user settings). A common approach is using Hidden Markov Models (HMM) [92, 153, 23] and Multiple Behavioural Hidden Markov Models (MBHMM) [97], which outperform the commonly used Viterbi algorithm [50]. In home task identification Viterbi determines the most likely sequence, while MBHMM determines the probability of being in a final state of a model, given the window of observations being considered. This approach (modelled with the idea of profile HMM) solves the problem of missing sensor readings and allows unexpected sensor event occurrences between expected events, having the ability to determine which tasks are currently active, even if the activity has not been concluded by the user. This can be due to having several ways to finish the activity correctly, or having actions that can be executed in any order. Another work dealing with lost sensor events is [98], in which the authors use a size and relevance based Hierarchy of Activities of Daily Life (HADL). In [49] Ermes et al. use acceleration, temperature, illumination and humidity sensors to infer if the user is rowing, biking, playing football, walking, running, sitting, or hiking. Training and test data are validated with decision trees and artificial neural networks.

As an attempt to avoid body markers to acquire information from the user, the latest approaches consider video sensors, such as in [88], to analyse silhouettes and walking rhythm, with an HMM per user. [116] uses infrared cameras for hand gesture monitoring, and [33, 148] use Kinect depth images to recognize people indoors. More on reviews of vision based activity recognition can be found in [154, 104, 27].

2.2.1 Human activity recognition

Due to the growing interest in designing smart environments that reason about residents [44], intensive data collection is becoming more common. The solution with HMM in [54] focuses on real-time recognition of activities interrupted and interleaved, among multiple residents in smart environments. Manually labelled data from 40 residents served to identify the most probable sequence of activities that corresponds to a sequence of sensor events. The result average accuracy was 60.60%. However, activities which take longer and generate more sensor events tend to be recognized with greater accuracy (e.g., reading a magazine, 94.38%), than others (like e.g., setting the table, 21.21% accuracy) that do not generate enough sensor events to be distinguished from other activities. Parallel activities and in cooperation have not been considered; and a strong constraint is their assumption of knowing the person ID for each sensor event.

In [35], an unsupervised learning algorithm is applied to detect social interaction in the *CASAS* smart environment [113] and monitor ADL. Activity and event density maps visualize sensor events during 15 days in a 2 resident apartment. Bayesian updating is used to detect and track individual interactions, while a supervised learning algorithm with two HMM (resident identifier model and activity identifier model) gave an average accuracy of 90%. Viterbi algorithm is used [35, 54] to compute the most likely sequence of hidden states. However, since not all interactions can be determined by physical proximity, [35] suggests to fuse the resident identification model and the activity identifier model into a multi-layer hierarchical model to improve the activity recognition task in multi-resident settings.

HMM have the limitation of being stationary; therefore, they can not be applied to dynamic environments. HMM assume order restriction among the actions that conform the behaviour and show limitations when activities can be carried out in different ways. In [97, 79] Skip Chain Conditional Random Fields are proposed, together with Multiple HMM to consider any behaviour variation. However, HMM have the added inconvenient of the first order assumption, which can difficult the detection of temporal restrictions among actions, when the user realizes multiple tasks simultaneously.

Ros et al. propose in [41, 129] inductive learning for detection of temporal relations applied to behaviour detection and recognition. In [126], human activity modelling is enhanced with the on line adaptation to habit changes, environmental variations and temporal information by means of learning automata and fuzzy temporal windows. This results in an independence of the on line learning and adaptation capability, respectively. This means that the proposed abstract behaviour is independent of the method used for learning. One advantage of a graphical model such as learning automata is the ease of user monitoring. The methods were validated on a real space, *iSpace* at University of Essex [127], and they were applied at the University of Missouri, where the process of rehabilitation in patients with degenerative diseases was followed up [128, 14]. Furthermore, the approach in [126] overcomes limitations of HMMs regarding first order assumption [92, 153, 23] because behaviour actions are structured in execution levels. The normal behaviour detection rate obtained is 92.50% versus a 69.51 % with HMM. As [126] suggests, the development of a semantic representation of the environment, behaviours and activity recognition system starts to be essential in this kind of problems.

In activity modelling we can find research on different challenges, for instance, on behaviour mining [114, 146, 23, 66], systems which tackle the presence of multiple users [39, 35], the migration of learnt behaviours to other spaces, their adaptation to other users [111, 112] and the adaptation of the models to changes in the user activity [72, 115, 138]. However, it does not exist a general solution that integrates all these aspects. There is no consensus on what kind of sensors should be used, what activity models or what information treatment methodologies should be employed in each case. This is due to the fact that it mostly depends on the type of application to model. However, as a conclusion on the state-of-the-art, we can confirm the greater attention earned by accelerometer sensors in the latest years, and recognize stochastic methods such as HMM, as the most extended ones on the literature to model a concrete activity.

Regarding the office domain and public buildings, we can find *EasyMeeting* [31], an intelligent meeting room system that builds on the design of *CoBrA* (Context Broker Architecture) [30]. An illustrating scenario is shown in an intelligent meeting room with RFID sensors embedded in the walls and furniture for detecting the presence of the users' devices and clothing. On receiving information about the user's context and intention, the broker sharing platform allows activation of projector, slide download and set lighting control. On [42], ontology-based interoperability is applied to Smart Spaces for a context-aware maintenance of large buildings, monitoring environmental variables, automatically detecting building-related faults and executing interventions in a multi-modal way when required. An architecture for ubiquitous group decision support system, *WebMeeting* [90], is able to consider the emotional factors of participants and their associated processes of argumentation. As intended for intelligent decision making, and part of an AmI environment, the distributed meeting involving people in different locations with access to different devices (e.g. computers, PDAs, mobile phones, or embedded systems) is also asynchronous (participants do not need to be involved all the time). The system shows available information to the participants, analyses the meeting trends and suggests arguments to be exchanged with others. The AIRE project [10] focuses on intelligent workspaces, conference rooms, and kiosks, using gaze-aware interfaces and multi-modal sketching to capture speech discussion semantics between co-workers and writing on a whiteboard. The *SmartOffice* or *Monica* project [53] identifies gestures and activities to retrieve needed workplace environment information. It has a speech recognition and internet agents. The latter serves, e.g., for not announcing a new e-mail message during a speech or a meeting. A user position tracker detects up to 3 persons simultaneously. Other components are *MagicBoard*, *MagicDesk*, finger, click and face recognition, phycons [1] and activity detection for low-level activities (walking, standing, sitting, falling, etc.). These serve to build higher-level pieces of context with an event-based scenario detector. Other project is the Interactive Room (*iRoom*) at Stanford [51], which enables easy retrieval and display of useful information. NIST Smart Space and Meeting Recognition projects develop tools for distributed processing that aid context-aware smart meeting rooms [135]. At last, MOSES system [136] infers where the staff is located and what task are performing using RFID technology. Having workers equipped with RFID readers, the system can track the development of activities and thus advise the employee with remaining tasks to be done.

2.3 Semantic human activity representation

Since different scenarios are deployed within heterogeneous ubiquitous spaces, the diversity of systems prevents the use of a well-formed hierarchical classification, making a consensus in evaluation tools difficult. [7] proposes a taxonomy to evaluate a wide variety of pervasive computing systems. The main criteria presented for evaluation consists of the concepts *Architecture, Application Purpose, Autonomy, Integration, Interaction, Intelligence* and *Service Availability.*

Generally, a wide expressibility model is needed to deal with the description of all possible features of a user and the functionalities provided by devices and systems [93]. [93] proposes an ontology to model AAL services for the elderly in a domotic domain. Their particular aim is to facilitate validation of accessibility (i.e., disability constraints) for users, starting from the design stage.

Since we will focus on human activity representation based on ontologies, W3C technologies and recent standards on Semantic Web should be followed. SPARQL is the W3C standard query language for RDF (Resource Description Language) since 2008. However, SPARQL query engines are not possible to run in resource-constrained devices. The work in [11] focuses on perva-

[1] Phycons are common physical objects whose presence or position has a special signification for the system. E.g., a Rubik's cube on the desktop, with the red face up, means to stop the image transmission[53].

sive applications in which semantic knowledge processing and reasoning is required in resource-constrained devices. They select a subset of SPARQL expressiveness, as an extended version of N-Triple pattern language [2], and use a subset of OWL-Lite. An inconvenient, therefore, is the limitation on OWL and complex SPARQL queries.

In [48], upper and lower level ontologies are presented for modelling context. Their platform query engine, however, relies on RDQL, a previous query language to the standard SPARQL.

The proposal in this work goes beyond the classical work, in which behaviour of a unique user is modelled, such as in AAL approaches. This is because in certain domains such as office or public domain, there can be multiple persons and behaviours can be shared by two or more people. The challenge is thus on recognizing activities in multiple user settings, since sensor readings for an activity are not contiguous if other users are performing other activities in parallel.

In order to model imprecision and uncertainty that different service providers, devices and sources of information can origin when modelling context, we propose a fuzzy ontology as methodology. One of the reasons is because fuzzy ontologies have shown to be useful tools for knowledge mobilisation [25]. Next chapter details our proposal.

[2]SCENT: Semantic Device Language for N-Triples

Chapter 3

Semantic modelling of human activity in Smart Spaces: A case study on public buildings and office domain

In this chapter our ontology-based proposal is presented. Having the inference and learning of human behaviours as our ultimate goal, both the environment and the user must be provided with semantics. There are many ways of obtaining context information from the environment in order to extract and establish semantics. Information sources can be heterogeneous: video, audio, sensors, mobile devices, positioning systems, etc.

Ontology-based activity recognition provides a number of advantages [32]: it supports incremental progressive activity recognition, state based modelling and a more robust reasoning since there is no fixed sequences for an activity, especially for ADLs. Other benefits are the ability to discriminate importance and urgency of activities through semantic descriptions, support for course-grained and fine-grained activity assistance and the possibility for data fusion and semantic reasoning, including activity recognition, activity learning and activity assistance.

On our case study we consider a work environment scenario showing examples of different types of activities, e.g. in public buildings such as a university department or regular offices. At this point, we focus on the representation of context-aware human activity.

3.1 Ontologies

The literature offers a wide range of ontology definitions [37]. An ontology is a "formal specification of a shared conceptualization" (W.N. Borst, 1997). It offers formalism to represent classes or concepts, individuals, relations, functions and attributes. The advantages of using ontologies is their way to represent and share knowledge by using a common vocabulary. As providers of a format for exchanging knowledge, they promote interoperability, knowledge reuse and information integration with automatic validation. They separate declarative and procedural knowledge, making easier the modularity of the knowledge base (KB)[20]. Ontologies allow information to become not only human but also machine-readable (by agents).

3.2 Existing ontologies and tools for context-aware modelling

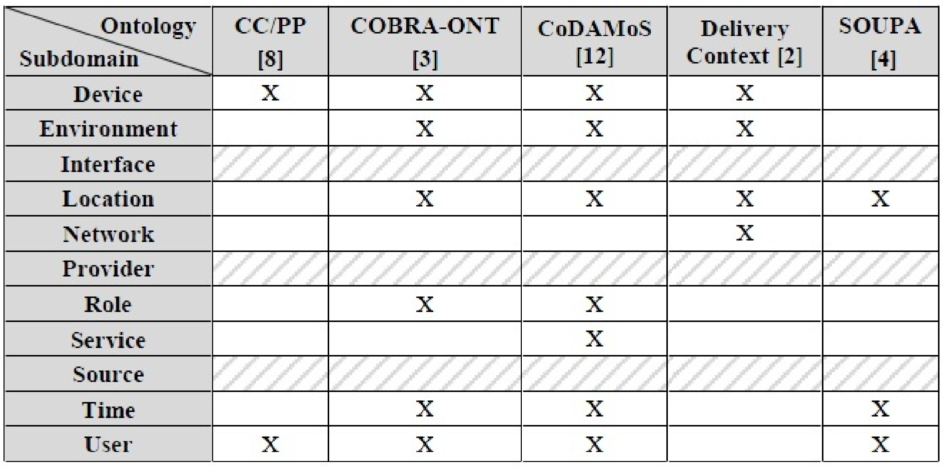
In order to find an ontology that satisfies our problem domain, with the aim of reusing existing work for expanding and scale our ontology, several tools assist on finding actual ontologies and metadata. Among others, there exist:

- *OMV*, Ontology Metadata Vocabulary [1].
- *Oyster*, a distributed ontology registry [2].
- *Swoogle*, Semantic Web search engine and metadata service provider [3].
- *Watson*, a kind of ontology search engine that allows to query for existing descriptions of selected entities for reuse [4].
- *Scarlet*, discovers semantic relations between concepts by exploring the entire Semantic Web as a source of background knowledge (relying on Semantic Web gateways such as *Watson* or *Swoogle*) to automatically find and combine knowledge provided by multiple on line ontologies [5].

When a candidate set of ontologies is found, criteria for assessing them needs to be identified with respect with the problem domain; in our case, the work environment. For instance, in [61], a formal context model composed by four independent ontologies (users, devices, environment and services) is presented for describing intelligent environments and enabling interoperability with external ontologies. Some modelled aspects of our interest are locations in time, devices for services and interfaces.

A more detailed state of the art in available context-aware ontologies is well summarized in [143] as Table 3.1 shows.

[1]http://omv.ontoware.org/
[2]http://oyster2.sourceforge.net
[3]http://swoogle.umbc.edu
[4]http://kmi-web05.open.ac.uk/WatsonWUI/
[5]http://scarlet.open.ac.uk/



Ontology / Subdomain	CC/PP [8]	COBRA-ONT [3]	CoDAMoS [12]	Delivery Context [2]	SOUPA [4]
Device	X	X	X	X	
Environment		X	X	X	
Interface	▨	▨	▨	▨	▨
Location		X	X	X	X
Network				X	
Provider	▨	▨	▨	▨	▨
Role		X	X		
Service			X		
Source	▨	▨	▨	▨	▨
Time		X	X		X
User	X	X	X		X

Figure 3.1: Subdomains addressed by available context ontologies [143].

3.3 Entities to consider for context modelling in Smart Spaces

A Smart Space represent an abstraction of a physical space, its (semantic) data repository and a protocol to access it. We propose an architecture to model context in a SS based on observations in the environment. Sub-domains needed for our domain specific applications are content extraction, on-line social networks and, in general, support for human activity representation. Some elements to be modelled on our ontology have been studied previously in other contexts, and thus, can be matched for reuse and/or extension of our ontology. These, and other concepts considered relevant in context-modelling, are detailed below.

- Activities in work environments: Activities can be extended with e.g. Meeting Ontology Knowledge Base within CALO[6] Project, the Office ontology [7], etc.

- User profile and preferences: People can be modelled with the FOAF ontology [8]. Composite Capabilities/Preference Profiles (CC/PP) W3C structure and vocabularies provide profiles to describe device capabilities and user preferences [9]. A W3C Delivery Context Ontology and a glossary of terms for device independence exists (with discontinued maintenance). Ontologies in the semantic desktop Gnowsis project focus on use cases such as tagging a file, e-mail or a web [10]. Nepomuk (Social Semantic Desktop) User Action Ontology describes desktop events

[6]Cognitive Agent that Learns and Organizes
[7]http://www.daml.org/ontologies/369
[8]Friend of a Friend Vocabulary: http://xmlns.com/foaf/spec/
[9]http://www.w3.org/TR/CCPP-struct-vocab/
[10]http://gnowsis.opendfki.de/

and their Calendar Ontology (NCAL) adapts the W3C ICALTZD ontology [11].

- Time: the OWL Time ontology, carried out by the Semantic Web Best Practices and Deployment Working Group (SWBPD) in W3C can be reused. Furthermore, the pattern extension in [143] allows representing an enumeration of different individuals (e.g., to model the days of the week).

- Online Behaviour, such as in social communities: The SIOC (Semantically-Interlinked Online Communities) [12] ontology describes information from online communities (e.g., message boards, wikis, weblogs, etc.) on the Semantic Web. E.g., *Foursquare* check-ins could be managed, by SIOC, to deduce location or companion. A natural extension to SIOC, for online behaviour is the Open University Behaviour Ontology (OUBO), which allows user behaviour to be captured over time and facilitates role inference [130].

- Content extraction: Image Annotation W3C [13] ontology for semantic image annotation and retrieval can be used for a deeper multimedia analysis (e.g. image-based context recognition). Nepomuk Multimedia Ontology (NMM) also defines metadata for multimedia files, as well as EXIF ontology describes digital camera images and image management software [14].

- Interfaces, devices and services: The *mIO!* context ontology is a network ontology that represents user context to be able to process and use it to configure, discover, execute and enhance different services that the user may be interested in [143].

- Location: For instance, *PlaceTime.com*[15](URIs for places and times) contains instants and intervals in the Gregorian calendar and points in the WGS 84 datum, utilizing the RDFIG Geo vocabulary. Other vocabularies useful in object and human location tracking are WGS84 Geo Positioning [16] or GeoNames [17].

These ontologies can serve to augment our ontology with a lower level of granularity by declaring equivalence assertions and constraints. Specifications for constructing ontology networks [89] can help combining different ontologies. Following a methodology to guide ontology development is shown useful [143], as well as reusing knowledge resources and attend to good practices in the

[11]http://oscaf.sourceforge.net/ncal.html
[12]http://rdfs.org/sioc/spec
[13]http://www.w3.org/2005/Incubator/mmsem/XGR-image-annotation/
[14]Shared-Desktop-Ontologies (SDO) http://oscaf.sourceforge.net/
[15]http://placetime.com
[16]http://schemapedia.com/schemas/geo
[17]http://www.geonames.org/

ontology development. However, reusing context ontologies can become difficult due to the different purposes and requirements for which the ontologies are designed. For the moment, in human activity representation, since we focus on public buildings and work office scenarios, we build our customized ontology, which can always be later extended.

3.4 An ontology for context-aware human activity representation

This section details the design principles of our ontology to represent human activity. When it comes to human behaviour, we can distinguish among three types of activities:

1. *Single User.* Some activities involve a unique user. Aspects to consider when learning about behaviours contain the user's location, calendar, role/position, etc.

2. *Multi-user/Generic User.* More generic activities involve or affect multiple users or a generic kind of user. Since this kind of generic user embraces a joint or shared behaviour among a group of people realizing the same action (e.g., meeting attendants, visitors, students, etc.), they should not be treated individually, but as an abstract generic user that acts in the environment. Special features of this group can consider users of a programmed activity, a schedule associated to the activity, role, access rights, etc. Some of these aspects are shared properties with single users actions and activities. However, the *GenericUser* class will be useful when personal data about a user(s) is not known but is relevant as registered or observed entity in the environment context.

The three main categories of concepts to model human behaviour are:

- *Environment.* An organized hierarchy of, e.g., locations, will model generic and specific features of each kind of space. For example, in the office/work environment, we will represent offices, meeting rooms, lecture rooms, auditoriums, kitchen, toilets, etc. Different levels to track, in this case, indoor positioning may include floor and room numbers, while outdoor locations may refer to open spaces or means of transport. A location can have associated measurements such as *Humidity, Temperature, Lighting, NoiseLevel* or *Pressure.* For a more fine grained environment, objects (e.g. doors, curtains, windows) can have an *Aperture* and rooms and locations a (seating) *Capacity.*

- *Activities and Actions.* Any user can perform from among a hierarchy of (atomic) actions and activities. Depending on the actors in the environment, will be individual or collective (social) activities.

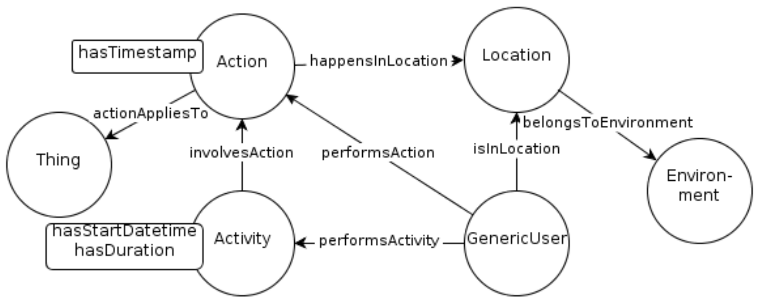


Figure 3.2: Main relationship among User, Activity and Environment in the modelled context ontology.

- *Relations.* Relations model the interaction among users and the environment, and specify behaviours happening on the latter. Possible relationships can be among single users, single user with a generic (multi)user, or among a user and the environment.

Part of the ontology reuses some hierarchy concepts from existing ontologies, e.g., the indoor/outdoor hierarchy for locations from the *CONON* ontology [147] or the environment features (temperature, pressure, etc.) from *CoDAMos* ontology [105]. The final combination of these aspects in a user context ontology represents a systematic relationship among activity, user and environment. This essential relation can be seen in figure 3.2.

Other features of interest modelled in the *HumanActivity* Ontology are the following:

1. *Activity Duration and Concurrency*: Several activities or the same one can be performed by different people at the same, or overlapping times. Some events, resultant of a user's actions, remain done for another user later, e.g., turn the light on or set the projector on, which are done by a user and remain done for the next one, within some time frame. The second user does not have to perform that action in order to complete the same activity as the first one, unless the light or projector has been set off. To model these situations, the boolean data property *remainDone* of the class *Action* indicates when an action required by an activity has not been done because it remains done or "active" by somebody else.

2. *Activity Characterization and Indispensable Actions*: Not all users may perform a given behaviour or activity on the same way. Therefore, an activity can be performed according to more than one behaviour model, but only one per user. Thus, the object relation *hasBehaviourModel (Activity OR Action, BehaviourModel)* is not functional, while *performsBehaviourModel (GenericUser, BehaviourModel)* is. However, in order to know, in an efficient manner, if certain action is absolutely needed in order to perform a given activity, the object property *isIndispensableForActivity (Action, Activity)* can be set once it is known, to

avoid further calculations during training period. Also *isIndispensable-ForBehaviourModel (Action OR Activity, BehaviourModel)* is a property to optimize obtaining a behaviour's required actions/activities (since different users might not execute all the actions that others require to perform the same activity).

3. *Messages/Alerts*: Activities such as *SendSMS* or *SendEmail* are associated with a message. Another use of a message is when an activity has been performed without realizing some of the required actions involved, and an alarm message must be generated, as a warning to the user, administrator or for security or logging reasons. Alerts are useful as reminders of forgotten actions or alarms of potentially hazardous situations. Types of messages are *Error, Alarm, Information* or *Suggestion*; modelling the message transmission happens through *SenderUser, SenderDevice, AddresseeUser* and *AddresseeDevice* classes.

The complete ontology can be seen in figure 3.3.

3.4.1 Semantic restrictions in the ontology

Any *GenericUser* (or *User*) can perform an *Activity* or *Action.* A *Behaviour* can be composed by:

a) *Actions*: atomic events, e.g.: *OpenDoor, MoveObject, TurnLightOff, WalkBy, BeObservedInLocation*, etc.

b) *Activities*, considered as more complex actions, actions with inherent "purpose", and involving (i.e., composed by) different actions, each possibly happening in different location. E.g.: *TakeCoffee, AttendConference, GroupMeeting, Videocall, SendEmail*, etc.

c) *A sequence of Activities and/or Actions*, since actions can have lower degree of granularity. E.g., the activity *CoffeeBreak* includes the Action *ExitOffice*, the Activities *MakeCoffee* or *TakeCoffee* and the Action *EnterOffice.*

It is important to note that the architecture presented in the ontology mainly represents event observations. Nevertheless, any *User* can have a *Calendar* with *Event*s he intends to attend. This does not mean that he will finally perform them, since more evidence will be required for affirming so. When this evidence in context will be certain, *(User, attendsEvent, Event)* will be asserted. Those activities that require to be represented as planned or scheduled, need to be modelled, however, only as *Event*s belonging to a certain *Calendar.* The rest of information in the knowledge base will not consist of plans, but only assertions of facts. At the same time, some *Events* can also involve some *Activity*, (which may involve a *GenericUser*). For example, the *Activity* subclasses: Symposium, Seminar, Workshop and Conference.

Therefore, an Activity including an Event is modelled through the relation *(Event, involvesActivity, Activity)*.

Once the ontology is populated, in order to detect certain behaviour performed, the order in which Actions are performed, within an Activity, can be used, by querying the timestamp of every Action and Activity. However, when representing behaviours, further semantic constraints need to be made explicit; these are detailed in the next section by using a regular automata model.

Value Partitions, Axioms and Relation constraints

- Classes like *DeviceStatus* are value partitions [65], meaning that the values they can adopt are mutually exclusive, e.g. something can not be *Available* and *Away*; or *On* and *Off* at the same time. They satisfy the covering axiom, i.e., the subclasses are covered by a covering class (in Protégé 4, this is manifested as a class that is equivalent to the union of the classes being covered [65]). Value Partition entities in our ontology are:
  - *DeviceStatus{On, Off}*
  - *BehaviourModel {RegularAutomaton, LearningAutomaton, HMM}*
  - *PersonalStatus {Available, Away, Busy, OnHoliday, OnLeave}*
- If a *User* performs an *Action* which happens in certain *Location*, we can affirm that the user is in the same *Location* as the *Action* is happening. This concept assertion can be expressed with a fuzzy RIA (Role Inclusion Axiom) [22]:
  $\langle performsAction\ happensInLocation \sqsubseteq isInLocation \geq 1\rangle$
  Since we aim at having an interoperable ontology as independent and modularized as possible, we have not modelled these rules, leaving them up to the user of the concrete application domain. Rules of these type can be modelled as e.g., SWRL [4] or SPIN [2, 3] rules.
- Other more elemental restrictions that can be modelled simply with OWL are, e.g., a *User* is a kind of *GenericUser* in which *hasNUsers* data property value is exactly 1: *GenericUser and hasNUsers exactly 1.*
- An *Event* must belong to at least one *Calendar*: *belongsToCalendar min 1.*

To finally demonstrate two examples that make explicit the use of the proposed ontology, we propose the following context rules:

```
  Activity: Conference
User: GenericUser
Rule: IF Conference for 25 people in the Auditorium at 14.00,
      THEN Lights.ON at 12.55 and Heating.ON to 23ºC at 12.30.

   Activity: Meeting
User: MeetingAttendant
Rule: IF more than 3 MeetingAttendants are in MeetingRoom and
      Lights.ON and Projector.ON,
      THEN SMS_Attendants("Please, silent your phone!")
```

3.5 Behaviour Model: Regular Automata

It is important to differentiate between behaviours, meaning more complex human activities, and the system behaviour, which is implemented as rules. It is necessary to model human activities to provide a higher level of abstraction to those rules. Not only writing simple expressions as *IF Conference for 25 people in the Auditorium at 14.00, THEN Lights.ON at 12.55 and Heating.ON to 23ºC at 12.30* should be able to be expressed, but also complex behaviours can be part of the rule's antecedent, as it could be, e.g. rules such as *IF User is performing/has performed a behaviour THEN do some actions.* For this reason, it is important to model complex behaviours.

We proceed to design AI behaviour models for behaviour representation, detection and disclosure. In general, activities are composed by a sequence of actions. However, this does not happen in practice all the time, e.g., some cycles can appear to complete "half done" actions, or forgotten things are made at the end. Other procedures are not followed in the logic order, but in the order in which the actions are actually needed, during its execution, without any specific sequence ordering. Therefore, a behaviour does not have an only way to be performed, since some tasks may be executed with different order constraints [126].

Although preliminary results suggest that existing ontological techniques slightly under-perform data-driven ones (mainly because they lack support for reasoning with temporal information), when extended with simple forms of temporal reasoning, their effectiveness can exceed that one of HMM's [122]. According to comparisons among OWL-based and HMM-based activity recognition, [122] shows the relevance and need to include temporal reasoning in ontological methods to improve accuracy and effectiveness. [122] suggests two approaches, using temporal description logics (which do not support OWL DL operators), or using loosely-coupled external temporal reasoners to treat time intervals (i.e. activity overlapping). However, our approach aims at a more abstract, interoperable solution, embedding and modelling the time-handling technique, in this case regular automata, in the ontology itself.

We propose a non deterministic regular automaton (RA) or regular grammar as the model to represent and abstract a behaviour, i.e., a set of activities and/or actions.

Definition 1 A finite non-deterministic automaton (NFA) is a quintuple *M = (Q, A, δ, q_0, F)* in which:

- *Q is a finite set of states.*
- *A is an alphabet of symbols called input alphabet.*
- *δ is an application called transition function*

$$\delta : Q \times A \to \wp(Q)$$

  where $\wp(Q)$ are subsets of Q.

- *q_0 is an element of Q, called initial state.*
- *F is a subset of Q, called set of finite states.*

When the automaton gets an entry, and a given state, it can evolve to several possible states (including an unique state, or none if $\delta(q, a) = \emptyset$). Therefore, it acts as an algorithm that, in a given moment, let us choose several options, or even none at all. A word is said to be accepted by a NFA if, following in each moment any of the available options, we arrive to a final state.

The transition diagram of a finite state automaton is a graph in which the vertices represent the different states, and the arcs, the transitions between states. Each arc is labelled with the symbol that corresponds to such transition. The initial and final states come labelled in a special way (e.g, with an angle/arrow in the initial state and with a double circle the final one). Transition diagrams for NFA can have a vertex where none, one or several transition arrows can exit with the same label.

Since all languages generated by a regular grammar can be recognized by a finite state machine, a reasoning system based on a regular grammar allows to define an automaton to represent a user behaviour. Its route allows to know if the user activity corresponds with a known user behaviour [64, 41].

Definition 2 Let C = $\{c_1, c_2, ..., c_n\}$ be the set of possible user's actions in some situation or domain. A human activity A is a finite set of actions:

$$Activity\ A = \{\alpha_1, \alpha_2, ..., \alpha_n\}$$

with $\alpha_i \in C \forall i$, where α_i is performed before α_k, if and only if $i \leq k$.

A human behaviour B is a finite set of actions and/or activities:

$$Behaviour\ B = A \cup C = (\beta_1, \beta_2, ..., \beta_m)$$

with $\beta_j \in A \cup C \forall j$, where β_j is performed before β_k, if and only if $j \leq k$.

In the ontology, the *BehaviourModel* class is composed by a set of Actions and/or Activities, through the object property *(Action OR Activity, hasBehaviourModel, BehaviourModel).* At least one of the states is an *InitialState*, and at least one is a *FinalState* (subclasses of *State* in the ontology). A state can be an origin or destination state, representing any *Activity* or *Action* class in our Human Activity ontology. A behaviour can be composed by a single action, a single activity, or a set of these both in any specified order. Thus, the RA is represented in the ontology through the classes *State, InitialState* and *FinalState*. The object property *happensBefore* represents temporal dependency requirements among *Activities* or *Action*s composing a determined *Behaviour* (thus, it has as domain and range *Activity OR Action* classes). The general regular automaton representation in the ontology can be seen in figure 3.4. The data property *hasTimeInBtw* of the left side state of the object property *happensBefore* indicates the time that passes in between the occurrence of those two actions/activities. This duration can be, e.g., an average calculated empirically, out of history data, or set by the user, in minutes. Later on, on section 3.7, we see that fuzzy labels (representing a fuzzy membership function) will be allowed, becoming *hasTimeInBtw* a fuzzy data property expressing duration: *{shortDuration, mediumDuration, longDuration}*. Note that each action has a *hasTimestamp(datetime)* data property.

Since in general, we need to know not only the order of the actions happening, but also who realizes them, and in occasions, which objects/persons they act over, all these aspects can be inferable through ontology reasoning. The requirement for behaviour reasoning is to annotate each executed *Activity* or *Action* with the user who realizes it.

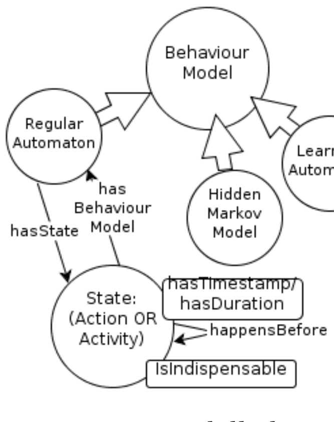


Figure 3.4: Regular Automaton as modelled in the Human Activity Ontology

Since temporal dependency requirements are imposed, any combination of actions that does not contradict the imposed order in actions/activities by *happensBefore* relations, will be recognized as a valid behaviour. Some actions can be done in any order within the activity, e.g. take milk from fridge in a coffee break, or take sugar. However, they may always be included within the required actions to complete a certain behaviour model. To represent the different variations on activities, *indispensable* actions or activities are those essential to happen in a given *BehaviourModel*.

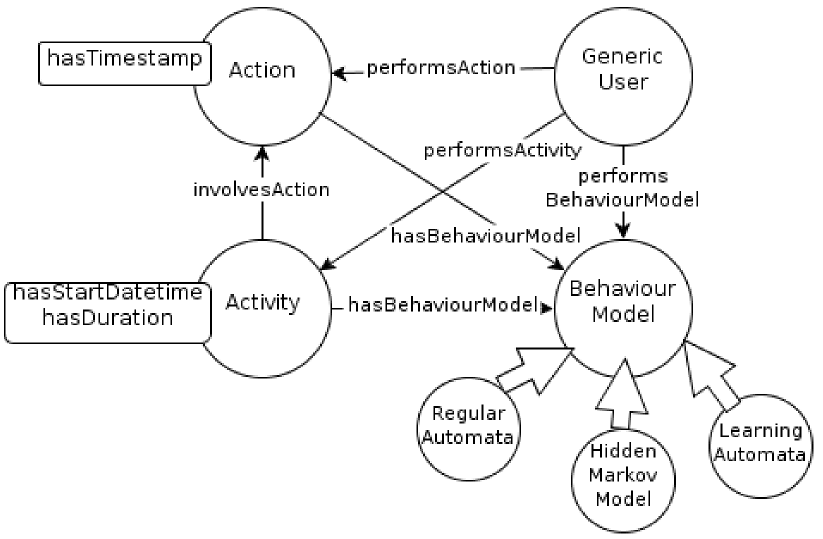


Figure 3.5: Activity Models and their relationships within the Human Activity Ontology

It can happen that certain users do the same activity in a simpler way, such that only one action needs to be recognized in order to infer the accomplished activity and its behaviour model. E.g., real coffee drinkers, which use no sugar nor milk, and therefore, *OpenFridge* (for milk) or *TakeSugar* actions are skipped, and not included in the behaviour model of these users. Since every user can have a different (max. 1) *BehaviourModel* way of performing behaviours, different subclasses of *BehaviourModel* are the classes HMM, Learning Automaton, etc. Its relationships with *User* and *Activity* classes within the Human Activity ontology are depicted in figure 3.5.

With the purpose of making behaviour, activity and action levels explicit, we detail an example with each part of the automaton, for a given behaviour, and also the equivalent ontological entities involved for its realization. Example 1 models, with regular expressions, a user getting out of his office, walking into the department kitchen and taking coffee or making it, if there is none available. Then he returns to his office, and the behaviour *To have a coffee break* is completed. The *involvesAction(s)* section represents the series of input symbols of the alphabet, i.e., sequential set of actions, needed for that activity. The behaviour *To have a coffee break* is related with the activities and actions that compose it, through the object property *(Activity OR Action, hasBehaviourModel, BehaviourModel)* (or inverse property *(BehaviourModel, hasState, Activity OR Action)*. In the same way, every action that composes an activity is related with it through the object property *(Activity, involvesAction, Action)*. Classes in parenthesis right after the activity or action represent the object over which the action applies to, e.g., *WalkBy (Corridor)* notation abbreviates the RDF triple *(WalkBy, actionAppliesTo, Corridor)*, as well as *BeObservedInLocation(Kitchen)* represents *(BeObservedInLocation, actionAppliesTo, Kitchen).*

Example 1:
Behaviour: *To have a coffee break*
hasState(s):
OpenDoor WalkBy(Corridor) BeObservedInLocation(Kitchen) (MakeCoffee + TakeCoffee) WalkBy(Corridor) OpenDoor.

-Activity 1: *MakeCoffee*
involvesAction(s): OpenCupboard TurnCoffeeMachineOn MoveObject(CoffeeJar) OpenFridge
-Activity 2: *TakeCoffee*
involvesAction(s): MoveObject(CoffeeJar) OpenFridge

Example 1 sequence diagram, illustrated in figure 3.6, shows the behaviour *To have a coffee break.* Activities are represented with white large nodes, while actions are represented in purple small nodes. Initial and final states are marked in turquoise colour with a start arrow and a darker border, respectively. Figure 3.7 shows, in more detail, example 1 at action level. Summarizing, the definition of the overall automaton for a behaviour happens through the composition of actions, through the transitive object property *happensBefore*, and the decomposition of each activity into one other automaton. E.g., reduction of the Example 1:

Behaviour: *To have a coffee break*
hasState(s):
OpenDoor WalkBy(Corridor) BeObservedInLocation(Kitchen) (OpenCupboard TurnCoffeeMachineOn MoveObject(CoffeeJar) OpenFridge)+ (MoveObject(CoffeeJar) OpenFridge) WalkBy(Corridor) OpenDoor

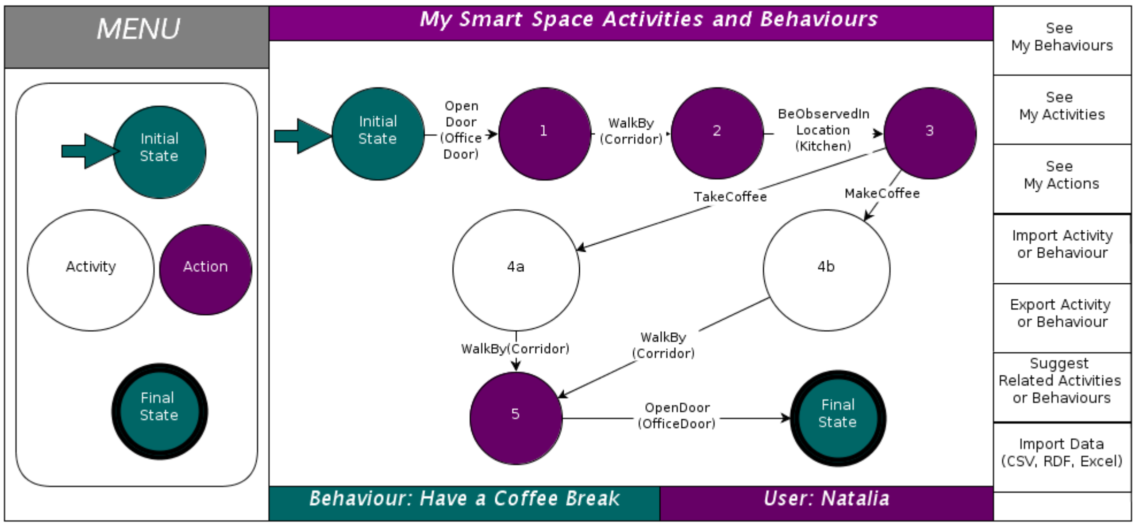

MENU
My Smart Space Activities and Behaviours
Initial State
Activity
Action
Final State
Open Door (Office Door)
1
WalkBy (Corridor)
2
BeObservedIn Location (Kitchen)
3
TakeCoffee
MakeCoffee
4a
4b
WalkBy(Corridor)
WalkBy (Corridor)
5
OpenDoor (OfficeDoor)
See My Behaviours
See My Activities
See My Actions
Import Activity or Behaviour
Export Activity or Behaviour
Suggest Related Activities or Behaviours
Import Data (CSV, RDF, Excel)
Behaviour: Have a Coffee Break
User: Natalia

Figure 3.6: Example 1 sequence diagram of the regular automaton for the behaviour *To have a coffee break*.

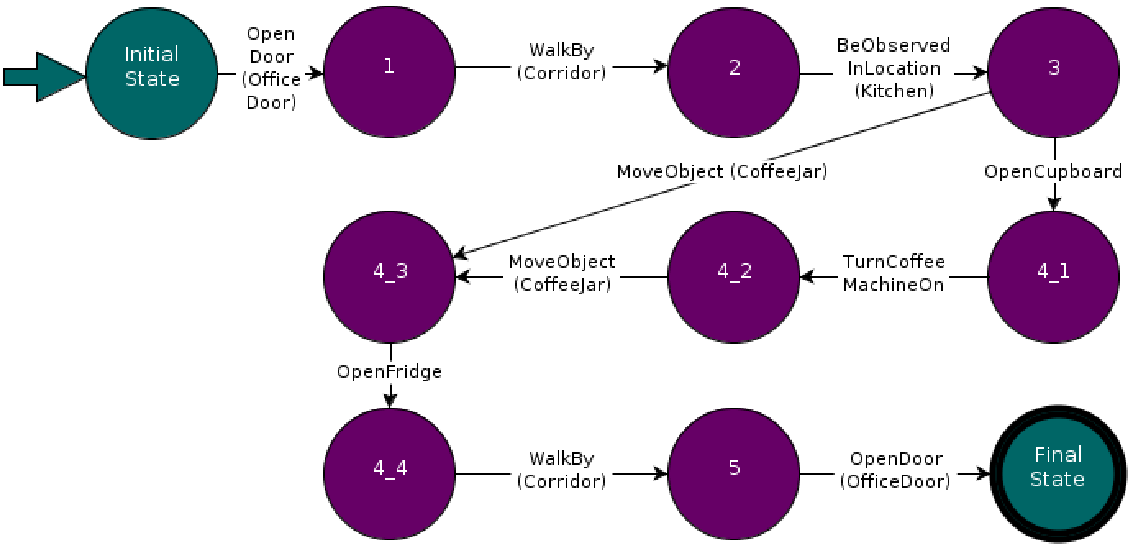


Figure 3.7: Example 1 sequence diagram, at lower Action-level, of the regular automaton for the behaviour *To have a coffee break*.

3.6 Modelling and implementing activity representation

In this section we provide two different approaches to allow two different kinds of end-user, to program the behaviour of a Smart Space, based on simple rules. The first approach, part of the Finnish project DIEM [18], focuses on a programmer end-user (i.e., assuming basic Object Oriented (OO) programming skills). The second approach presents a visual programming framework to specify semantic rules, going a step higher in abstraction and aiming to a broader end-user public, with no requirements on technical background. The common ideal underlying both approaches is to increase programming productivity and prototyping in heterogeneous spaces through adaptable and scalable ontology-based development of Smart Space applications.

3.6.1 Programming context-aware *PythonRules* for Smart Spaces

To allow the end-user to program the behaviour of a Smart Space, a toolchain was developed within our group [75, 131]. The framework uses Nokia's **Smart-M3**, a Multi part, Multi device and Multi vendor (M3) platform consisting of independent agents which communicate implicitly by inserting and querying information in the space. Smart-M3 is an open source, cross-domain architecture where the central repository of information, Semantic Information Broker (SIB), is responsible for information storage, sharing and management. Entities called Knowledge Processors (KPs) implement functionality and interact with the Smart Space by inserting/querying common information through the publish/subscribe Smart Space Access Protocol (SSAP)(figure 3.8). Communication happens not device to device but through the SIB. Entities and services are described with OWL (Web Ontology Language).

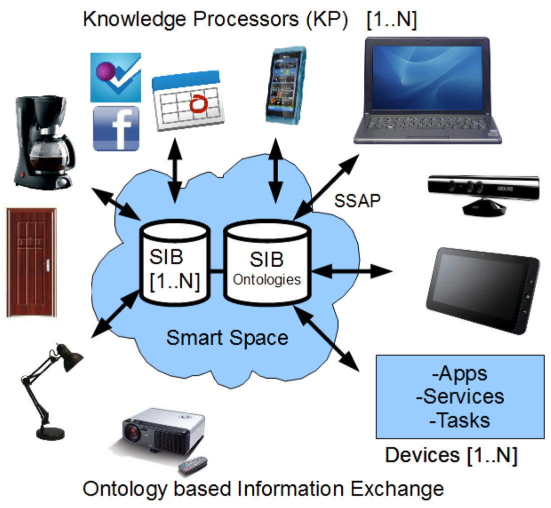


Figure 3.8: Smart-M3 Architecture for Smart Spaces

Benefits of publish/subscribe (or "push") semantic architectures, such as M3, include the inherent polling and a strong decoupling of the communication clients with respect to time, reference and data schema, increasing flexi-

[18]DIEM Project (Device and Interoperability Ecosystems) http://www.diem.fi

Protocol comparison for SPARQL and SSAP query languages	
SPARQL Query	**SSAP Query**
-	*Join* -Join a named Smart Space
-	*Leave* -Leave a Smart Space
INSERT [DATA] [INTO]	*Insert* -Insert information to Smart Space
SELECT	*Query* -Query for information in Smart Space
DELETE [DATA][FROM]	*Remove* -Remove information from Smart Space
UPDATE	*Update* -Update information in Smart Space
-	*Subscribe* -Set up subscription (persistent query) to receive notifications when data changes
-	*Unsubscribe* -Cancel an existing subscription

Table 3.1: Protocol comparison for SPARQL and pub/sub-based SSAP query languages

bility in application design and allowing for more autonomous system architectures [96]. Table 3.1 compares the different query operations for SPARQL and SSAP protocols. SSAP can use WQL [19] or RDF triple pattern queries. Although RDF is underlying both models, the main difference lays on extra functionality, in SSAP, for joining/leaving a confined SS data store, as well as the subscription advantage, for awareness of data changes. However, this capability easily becomes a performance bottleneck and efficient implementations are currently unavailable, or in early stage[20].

Development Framework

Our framework's main aim is the integration of ontology-based logic programming with regular programming languages, for rapid SS application development. The development framework consist of 1. *Smart-M3 Ontology to Python and C API Generators* [75, 1] to make more intuitive to the programmer the definition of applications by providing automatic generation of a Python API for every OWL class, as well as setters and getters, among other methods, to easier interact with the SIB. 2. *Middleware* layer to abstract the communication with the SIB. When KP templates (generated by the previous tool) are used, instance declarations automatically translate to RDF insertions/updates into the SIB. This allows other KP applications, connected to the SS, to know about those individuals' existence for interaction. 3. *Python-Rules for Smart Space programming*[131] to ease interaction by providing a higher abstraction for fast specification of the SS' behavior.

[19]Wilbur Query Language: http://wilbur-rdf.sourceforge.net/
[20]Redland Smart M3 v0.3.1-alpha: http://sourceforge.net/projects/smart-m3

PythonRules inference rule module is based on logic programming and Python's meta-programming features. It allows writing Python code including logic inference rules statements. Because of its versatility, meta programming opportunities and ease of prototyping (easy to learn and use) we chose Python. Thus, given a context, the programmer could define, in a simple way, the underlying rules that pervasively help the user in daily activities. Rules follow the Event-Condition-Action (ECA) paradigm, and its evolution/adaptation is caused by KPs taking part in the SS. Rules can define how to infer the user's activity using the active context information from multiple KPs, e.g., how to handle emergency calls while being at a meeting or when to set the projector on if a meeting activity is inferred. However, if the meeting attendant's caller is his wife, the user may want to have a rule that lets the call go through.

A **PythonRule** is defined with a 3-clauses pattern:

```
With() // When() >> Then()
```

The **With()** Clause represents *assertions* or *declarations* of individuals; the **When()** Clause represents *conditions* or *events* that must hold before the triggering, and the **Then()** Clause represents *actions* to trigger or *conclusions* to infer. The implementation approach is inspired by *Pythologic, Prolog syntax in Python*[21]. In this way, the application programmer does not deal with RDF triples, queries or namespaces directly, but mainly with logic Python expressions.

Programming Knowledge Processors in Python

An example of rule could state, "if the user is in the meeting room (B4050) at a time between 13:00 to 15:00, having more than one person there and the projector on, we infer the user's status as *Busy* in a meeting". Another *PythonRule* example would set the user's phone voice mail if he is busy and located on the meeting room:

```
condition1 = lambda: user.hasPersonalStatus('Busy')
condition2 = lambda: user.isLocatedIn(meetingRoom)
conditions = [condition1, condition2]
action =  lambda: usersPhone.setVoiceMail(True)
myRule = With([user, meetingRoom, usersPhone]) // When
    (conditions) >> Then(action)
diem.addRule(myRule)
```

Listing 3.1: Rule definition with PythonRules Module

The abstraction of the *PythonRules* resides on a mapping of Python statements to (*RDF* triple-based or *WQL*-based) queries in the SSAP protocol:

[21]http://code.activestate.com/recipes/303057-pythologic-prolog-syntax-in-python/

- **With()**: If instances in *With()* exist in the SIB (SIB-Query), proceeds to evaluate *When()*.
- **When()**: If *When()* is true (SIB-Query), executes *Then()*. If not, sets a SIB-Subscription to the attributes in *When* clause to re-evaluate them when these change its value. This avoids unnecessary infinite query loops.
- **Then()**: If *With()* & *When()* satisfy, executes *Then()*, which translates into SIB-Update/ SIB-Add/ SIB-Remove/ SIB-Unsubscribe of RDF triples, or other Python statement.

A Knowledge Processor can be located in any device, e.g., smart phones, such as an application for getting the local temperature on-line. A KP can be a projector, thermometer activator, etc. and can be created and connected to the Smart Space (called 'x' in this example) as follows:

```
app = QtGui.QApplication(sys.argv)
smartSpace = ('x', (TCPConnector, ('127.0.0.1', 10010)
   ))
KP = PhoneKP.create(smartSpace)
# Definition of Rules
sys.exit(app.exec_())
```

Listing 3.2: KP Programming and Connection to a Smart Space 'x'

After the KP is created, the user can define *PythonRules* involving existing KPs which are connected to the Smart Space. *PythonRules* Module just needs to be imported as library to be used in the KP Python classes where applications are developed. With this approach, learning OWL or query languages is not needed for achieving device interoperability and interaction in the Smart Space. However, the disadvantage is that Python knowledge is required and SPARQL standard is not fully supported by M3. The next section presents a model requiring no programming skills from the user and with underlying SPARQL query language.

3.6.2 A graphical model for human activity representation, detection and recognition

In this section we propose a graphical model for representing, visualizing and interacting with SSs information, for rapid application development. This model allows any end-user to model his own activities and the features of the environment wherein they are performed. Data gathering is possible by aggregation of different datasets and ontologies.

The proposed interface, for the user to interact with the SS, is based on simple IF-THEN rules. The IF clause represents conditions and the THEN clause consequent actions, or applications/services to execute when triggering.

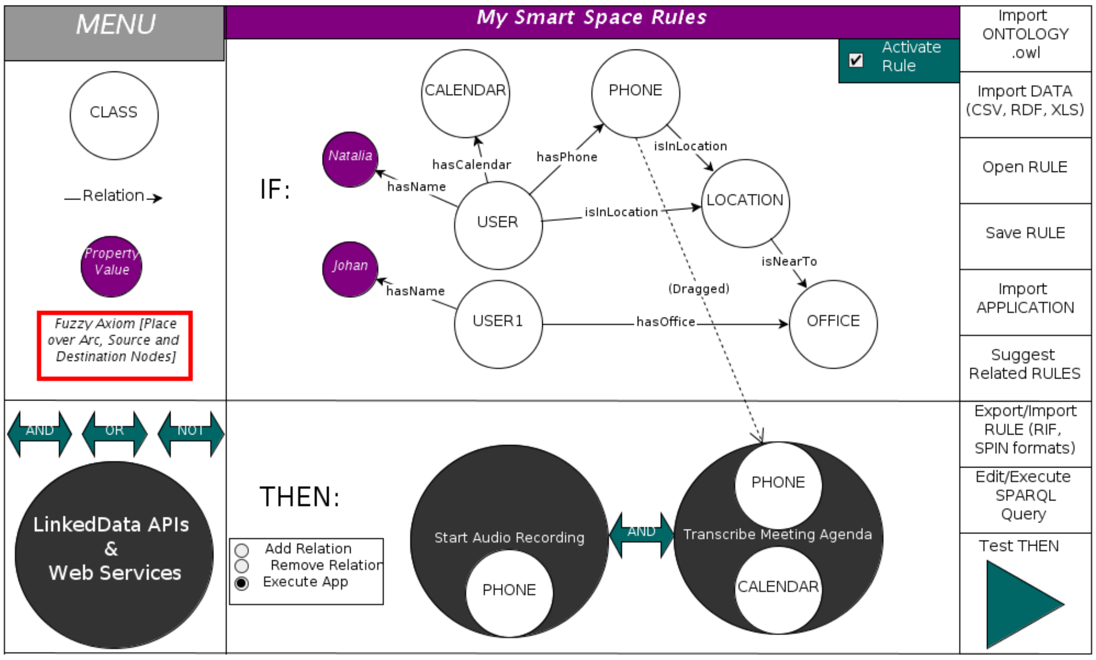


Figure 3.9: Example of user interface for semantic rule construction.

The visual model allows editing an IF-THEN rule at a time. The IF clause of the rule is composed by nodes and directed arcs which connect them. A node can be of two types, representing an OWL class (large and white) or a data property value (small and purple). An arc can be either a data property or object property, depending on the type of the destination node (*destNode*). If an individual is modelled, the destination node will be a data type value. However, if an object relation is modelled, both nodes united by an arc will be class nodes. If we want to model two different individuals of the same class, two separate nodes of the same class are required (e.g. see *User* individuals *Johan* and *Natalia* in fig. 3.9). All relation patterns are made by making explicit the corresponding nodes' connection. The THEN clause of the rule serves to 1) add, remove or update information in form of arcs and nodes (representing RDF triples) from the store, or 2) execute a registered browser-based application, possibly using concrete and well defined individuals or properties described in the IF clause and/or Linked Data APIs. This means that if the THEN clause view is set in *Execute App* mode, any class or property value (white or purple node) appearing within a grey large node, has to have been defined previously (above in the IF clause), in order to be identified, and serve as application parameter. Every registered web service or Linked Data service is represented in a large, grey node. Application grey nodes can be connected with logical operators for conjunctive, disjunctive or negative execution of the services. Green thick bidirectional arrows represent *Or, And & Not* operators. The features on case 1) build on top of SPARQL 1.1 [22] and

[22]SPARQL Update specification: http://www.w3.org/TR/sparql11-update/

for this purpose, arcs in the THEN clause can optionally be marked as *toAdd* or *toDelete*. To make the GUI simple, all relation arcs (i.e., data and object properties) and class nodes available in a (moderate size) store could be shown to the user in a hierarchy manner, on the left menu (by hovering). This would allow to visualize an ontologically organized data structure, to drag and drop the needed nodes and arcs to connect. Once selected a class node, data and object properties (with no apparent distinction at first to the user), are shown as available for selection. Once a property is selected, its range (data type or class node) is given to the user, as unique restriction, for him to fill in the node value to be connected. The graph-based and "puzzle"-like pieces to form rules are inspired by the successful *Scratch* programming framework [119] (in figure 3.10). Variable bindings are guaranteed to be correct, by construction of the user interface, through letting the user allocate pieces only in the positions in which corresponding data ranges and domains are allowed. Figure 3.9 shows the drag & drop SS interface proposal, for fast formulation of mash-up applications. The prototype example shows how the ontological model is projected into the end-user view.

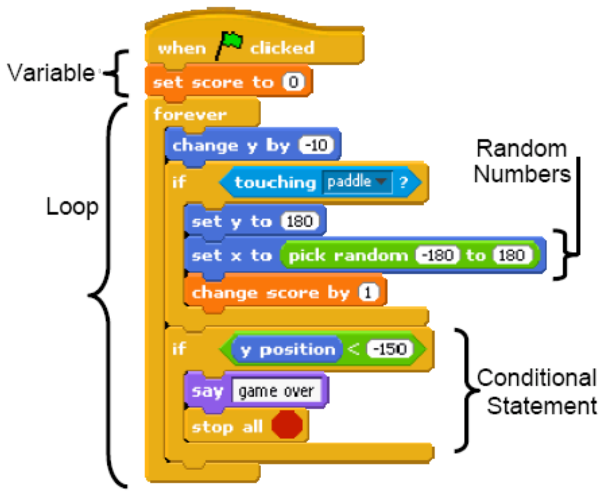


Figure 3.10: *Scratch, Programming for all*, example.

At last, useful functionality should be available to the end-user (right menu in fig. 3.9), e.g., to save and exchange rules (in SPIN [2, 3] or RIF format[23]), to test functionality before making a rule shareable and active in the SS, to check/edit the equivalent generated SPARQL query, as well as functionality to load a customized ontology, data (in different formats), or a given application.

This intuitive way of expressing a rule's IF condition, by dragging and joining compatible (data type-wise) nodes and arcs, can be easily translated into SPARQL query patterns (e.g., conditions in the *WHERE* field). Table 3.2 summarizes the visual model representation to be shown to the end-user,

[23]http://www.w3.org//TR/2010/NOTE-rif-overview-20100622/

as well as the mapping transformation applied to transform OWL 2 entities into *Fuzzy OWL 2* (detailed in next section).

A great power of visual languages is their ability of categorizations of certain primitives, and the graphical properties, to carry semantic information. Furthermore, elements of their syntax can intrinsically carry semantic information. To develop an effective visual language, i.e., a language that can be easily and readily interpreted and manipulated by the human reader, we followed guidelines for visual language design [59]. These can be summarized as: *morphology as types, properties of graphical elements, matching semantics to syntax, extrinsic imposition of structure, and pragmatics for diagrams.*

Visual programming constraints, in rule design time, are imposed on our GUI. E.g., when an instantiated class node in IF clause needs to be made concrete (through establishing property values), we show its (data and object) properties available (with that class as domain), when the user hovers that class node. Other design constraint that can be established for ontological cardinality restrictions is, e.g., not allowing more than one value for a given property, meaning that once a class node has that property set with a value, the UI will not show the possibility of setting that property twice. Since not all information may be captured directly through diagram syntax, the use of labelling languages, for labels which may potentially contain significant information, is necessary for most practical diagrammatic languages [58]. In our case, information to be obtained and shown when hovering, for an instantiated class node are, parent classes, descendant classes and all properties and semantic restrictions of given class.

SPARQL mapping to the end-user Graphical Model

Through structuring the edition of applications as simple IF-THEN rule statements, and by using an underlying graph based graphical structure, an end-user can model semantic behaviour, by means of available classes, relations and individuals, in a Smart Space. Objects or individuals can be selected from an RDF store which will reflect its content on the interface left side, by restricting the class with certain data properties and given values. In this way, general classes might not be required to be specified with details (e.g., Phone model or email of the Calendar), while the individuals in the KB to which the rule applies need to be explicit, if appearing as application parameter, on the THEN clause. Any constraint needed to make explicit a rule's IF condition needs to be identified by specifying properties and/or predicates of the selected class nodes, which will internally, and transparently to the user, translate to SPARQL query joins, that unify the RDF triples to retrieve. This retrieved data will presumably serve as input parameters to applications in the THEN clause of the rule, or to identify triples to add, remove or update data in the KB.

We show next some queries extracting the data to present in each view, when the end-user hovers the GUI left menu. The dataset may possibly be composed by diverse ontologies. Main categories to show as lists, when editing rules, are classes, object and data properties, data types and existing individuals. The following queries extract some of these:

Query I Show all classes (concepts) available. This query would return *WorkPosition, Action, Projector, EnterBuilding, Conference, SitDown, Projector*, etc.

```
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 ha: <http://users.abo.fi/ndiaz/public/
   HumanActivity.owl#>
SELECT DISTINCT  ?class
WHERE {
?class a owl:Class.}
```

Query II Show all individuals (i.e., instances) available. This query returns *Natalia, JohansOffice, NataliasCalendar, NataliasPhone*, etc.

```
SELECT DISTINCT  ?indiv
WHERE {
?indiv a owl:NamedIndividual.}
```

Query III Show all data types used in existing data properties. This query would show: *int, string, boolean, dateTime,* etc. and also defined fuzzy data types.

```
SELECT DISTINCT  ?object
WHERE {
?pred rdfs:range ?object.
?object a rdfs:Datatype.}
```

Query IV Given a class, show its data properties associated. E.g., data properties of the class *User*, would return *hasName, hasSurname, livesInMunicipality,* etc.

```
SELECT DISTINCT ?pred
WHERE {
?pred rdfs:domain ha:User.
?pred rdfs:range ?object.
?object a rdfs:Datatype.}
```

Query V Given a class, show its object properties (abstract roles) associated. E.g. object properties of the class *GenericUser* would be: *hasCalendar, hasPhone, hasPersonalStatus, performsActivity, performsAction, performsBehaviourModel, worksForProject, hasWorkPosition,* etc.

```
SELECT DISTINCT ?pred
WHERE {
?pred rdfs:domain ha:GenericUser.
?pred rdfs:range ?object.
?object a owl:Class.}
```

Query VI Given a class, show its associated individuals, e.g., instances of the class *User*. The query result for *User* is *Natalia, Johan.*

```
SELECT ?indiv
WHERE {
?indiv a ha:User.}
```

Query VII Retrieve the value of an individual's data property. E.g., this query retrieving the name of the *Project* individual *DIEM*, would return *Device and Interoperability Ecosystems.*

```
SELECT DISTINCT ?name
WHERE {
ha:DIEM ha:hasName ?name.}
```

Query VIII For a given class and data property, obtain its data type range. The result can assist the user in inputting a "filter" value when examining a concrete class' existing individuals. If the rule applies to certain individual, the user can choose data properties and its values to restrict the individual in his rule (and consequently, the SPARQL WHERE conditions). For example, to get the data type of the data property *hasName*, belonging to the Class *Project*, the query below would return *dateTime.*

```
SELECT DISTINCT ?type
WHERE {
?pred rdfs:domain ha:Project.
?type a rdfs:Datatype.
ha:hasStartDatetime rdfs:range ?type.}
```

SPARQL Query Plug-in 1.0.0 in Protégé 4.2 was used to demonstrate the viability of the user interface views. These views assist the user in editing context-aware scenarios through ontology-based rules composed by browsing existing data in the RDF store, as well as the structure of the loaded ontologies. The mapping of a visual IF-THEN rule is mapped to SPARQL with the following algorithm.

Visual Rule to SPARQL Query Mapping Algorithm

```
Initialize counter for ClassNode variables, n to 0.
Initialize processedNodes dictionary to empty.
<-IF CLAUSE MAPPING->
For each ClassNode in IFClause of the Rule:
  For each Arc leaving from ClassNode:
    If destNode is a Data Type: // Data Property triple
      Add patterns (?indiv_n a ClassName) and
           (?indiv_n dataProp destNodeDataValue) to WHERE
      Add originNode and its index n to processedNodes
      Increment variable index n
    Else: // The triple represents an Object Property
      If originNode is processed, obtain its index x
        If destNode is processed, get its index z
          Add pattern (?indiv_x objectProp ?indiv_z) to WHERE
        Else:
          Add pattern (?indiv_x objectProp ?indiv_n) to WHERE
          Add destNode and its index n to processedNodes
          Increment variable index n
      Else:
        If destNode is processed, get its index y
          Add pattern(?indiv_n objectProp ?indiv_y) to WHERE
          Add originNode and its index n to processedNodes
          Increment variable index n
        Else:
          Add pattern (?indiv_n objectProp ?indiv_n+1) to WHERE
          Add originNode and destNode to processedNodes
          Increment variable index n by 2
<-THEN CLAUSE MAPPING->
If THENClause.type is APP: // Execute external App
  For each ClassNode in THENClause:
    If ClassNode is processed, obtain its index w &
         add '?indiv_w' to SELECT
    Else: "ERROR: Class Nodes in APP parameters need to be defined in IFClause". Exit
  QueryResult = Run SPARQL Query with {SELECT, WHERE}
  Execute set of AppNodes with QueryResult as parameters
Else:
  If THENClause.type is ADD: // Add triples
```

```
38     For each Arc marked toAdd, add pattern to INSERT
39   Else:
40     If THENClause.type is REMOVE: // Remove triples
41       For each Arc marked toDelete, add pattern to
             DELETE
42   Run SPARQL query including {SELECT, WHERE, INSERT,
       DELETE}
```

The algorithm *parses* first the IF, followed by the THEN clause in the graphical model, to finalize running a SPARQL query with the parameters collected in some of the array structures SELECT, WHERE, INSERT and DELETE. Each *(origNode, Arc, destNode)* structure in the visual model corresponds to a triple pattern *(subject, predicate, object)*. The counter *n* keeps track of node indexes to keep unique naming (e.g., *?var1*) for each of their variables associated to use in the SPARQL query. Every arc is processed and, depending on the type of its destination node (line 6 & 11), the pattern is modelled as a) an individual's data property (asserting the belonging of an individual to a class, and a data property pattern) or b) an object property pattern (e.g., *(?indiv1 objectProp ?indiv2)*). Generated patterns are added to the WHERE field of the query [24]. However, if the arcs and nodes appear on the THEN clause, the visual model's equivalent patterns are added to the INSERT or DELETE fields of the SPARQL query, respectively, since these are triples that must have been marked as *toAdd, toRemove* or *toUpdate*. Finally, if the THEN class nodes are located enclosed in an application grey node, since they act as input parameters for the application, their variable is added to the SELECT field in the query, for a whole extraction of the node's information.

Rule Example Scenario

The model presented can serve to create any kind of ubiquitous domain application. To study its viability, we propose a location and context-aware scenario in a SS where positioning sensors are available through, e.g., each person's mobile phone. Let us suppose the end-user, with no programming skills, wants to create a rule which allows him to start recording audio of the meeting with his supervisor, automatically when he gets into his room, about every week. In natural language, the rule could be expressed as: "If *Natalia* enters the room of his supervisor *Johan*, start audio-recording the meeting agenda in her phone's calendar". The aim of this example is to keep track, for future reference, of the agenda points and brainstorming ideas discussed, on *Natalia*'s calendar. First of all, the user would select, from the GUI left menu, the needed classes (*User, Calendar, Phone* and *Office*), the data type values for identifying the individuals *Johan* and *Natalia* and finally, the relations which connect these with each other, and the needed artefacts, as figure 3.9 shows. A direct query for the previous rule example, could be:

[24]URIs and namespaces are omitted in the algorithm for simplification

```
SELECT ?phone ?calendar
WHERE{ ha:Natalia ha:hasPhone ?phone;
     ha:hasCalendar ?calendar;
     ha:isInLocation ?location.
  ?phone ha:isInLocation ?location.
  ha:Johan ha:hasOffice ?office.
  ?location ha:isNearTo ?office.}
```

which is shorter, but equivalent to the query obtained by our mapping algorithm:

```
SELECT  ?calendar1 ?phone2
WHERE{ ?user0 a ha:User.
     ?user0 ha:hasName "Natalia"^^xsd:string.
     ?user0 ha:hasCalendar ?calendar1.
     ?user0 ha:hasPhone ?phone2.
     ?user0 ha:isInLocation ?location3.
     ?phone2 ha:isInLocation ?location3.
     ?location3 ha:isNearTo ?office4.
     ?user5 a ha:User.
     ?user5 ha:hasName "Johan"^^xsd:string.
     ?user5 ha:hasOffice ?office4.}
```

The consequent of the rule is based on the conjunction of two simple applications that 1) would start a phone recording (*Natalia*'s), and 2) would transcribe the agenda stored in a phone to a calendar (in this case, also *Natalia*'s). Note the analogy with auxiliary variables in SPARQL, in the class node *Location* (fig. 3.9), used as auxiliary connector indicating that Johan's office location might be unknown, but must coincide with the proximities of Natalia's phone location. When completing the gaps that an application grey node has, i.e., its parameters, the user will drag the corresponding individuals or classes, from the IF clause, to fill the needed gaps (parameters) in the service. This results in a *pull* of information of the specific individual associated, and uniquely identified, in the IF clause. In figure 3.9's example, the user will drag (note discontinued arrow) class nodes *PHONE* and *CALENDAR* to fill the parameters in the applications in THEN clause. This represents a state change from data representation to data computation mode.

3.7 Treatment of uncertainty

Classical ontologies are not appropriate to deal with imprecise, vague and uncertain knowledge, inherent to real-worlds domains [20]. Fuzzy and possibilistic logics have proved to be suitable formalisms to handle imprecise/vague and uncertain knowledge respectively [20]. In the previous section we have developed the user context ontology to represent human activity modelling. However, there exist relations that induce to uncertainty. For example, "User isInLocation Room" will depend on the precision of the location sensor and of course, the sensor existence and proper functioning. Also, in "User hasEvent Event planned in Location at Time ", Time does not have to be exact in reality, etc. Therefore, uncertainty, imprecision and vagueness should be modelled as well. Thus, our previously defined ontology is adapted to support uncertainty; and imprecision can be present both in entities, properties and relations.

Fuzzy Description Logic (DL) is the most developed formalism to work with Fuzzy ontologies [20]. A fuzzy ontology is a relation on fuzzy sets, i.e. a relation associated with a membership function [62]. Formally, a fuzzy Knowledge Base (KB) or fuzzy ontology can be considered as a finite set of axioms [22]. In [20], up to 17 formal definitions can be found for *Fuzzy Ontology*. However, they believe they are not general enough and propose a more general definition: *A fuzzy ontology is an ontology which uses fuzzy logic to provide a natural representation of imprecise and vague knowledge, and eases reasoning over it.* It is to emphasize that the representation should be natural, since nowadays it is possible to represent some kind of fuzziness in crisp ontologies by means of ad-hoc solutions.

Fuzzy ontologies and fuzzy extensions of DL are appropriate to deal with vagueness or imprecision and have shown to be useful in applications from information retrieval and image interpretation to the Semantic Web and others [22]. In [62] a fuzzy keyword ontology serves to annotate and search event reports by superimposing a fuzzy partonomy [25] on fuzzy classifications. The fuzzy partonomy allows to find keywords which are partly the same for a query, regardless of where they are defined in the underlying keyword classification graph. With fuzzy reasoners, linguistic labels can be defined according to the context. In [26], fuzzy ontologies are shown to be effective in decision making and knowledge mobilisation [26]. [26] shows how amateur wine drinkers can become wine connoisseurs with support from knowledge mobilisation methods, such as approximate reasoning.

3.7.1 A fuzzy ontology for human activity representation

To represent our fuzzy entities in the already created Human Activity ontology, we use *Fuzzy OWL 2 2.1.1* plug-in [27] (in Protégé 4.1, jre 1.6) that provides support in creating *Fuzzy OWL 2* ontologies, without concerning about how to write the annotations. The plug-in makes transparent to the user the fuzzy representation part in OWL 2; however, it does not translate

[25]A fuzzy partonomy is a decomposition that fuzzifies the classical whole-part relationship.

[26]Knowledge Mobilization: "to make knowledge available for real time use in a form which is adapted to the context of use and to the needs and cognitive profile of the user" [77]

[27]`http://nemis.isti.cnr.it/~straccia/software/FuzzyOWL/#plug-in`

it in any way to OWL 2, but eases its representation. The user can define fuzzy elements in the ontology (fuzzy data types, fuzzy modified concepts, weighted concepts, weighted sum concepts, fuzzy nominals, fuzzy modifiers, fuzzy modified roles, fuzzy axioms, and fuzzy modified data types), and specify the fuzzy logic used in the ontology. By letting the user specify the name of the data type, and the type of the membership function, the plug-in asks for the corresponding necessary parameters [22]. The plug-in, also integrated with a *fuzzyDL* reasoner, allows to submit queries.

In *Fuzzy OWL 2*, three main alphabets of symbols are assumed: for concepts (fuzzy sets of individuals), roles and individuals [22]. These are represented in an ontology as classes, relations and individuals respectively. In [20], the degree of truth of a fuzzy assertion is equal to the proportion of observers who think that the crisp assertion is true.

Fuzzy Modifiers and Fuzzy Data Types

A fuzzy modifier is a function $f_{mod} : [0,1] \rightarrow [0,1]$ which applies to a fuzzy set to change its membership function, which can be *linear(c)* (fig. 3.11e) or *triangular(a, b, c)* (fig. 3.11b), where in linear modifiers we assume that $a = c/(c+1), b = 1/(c+1)$.

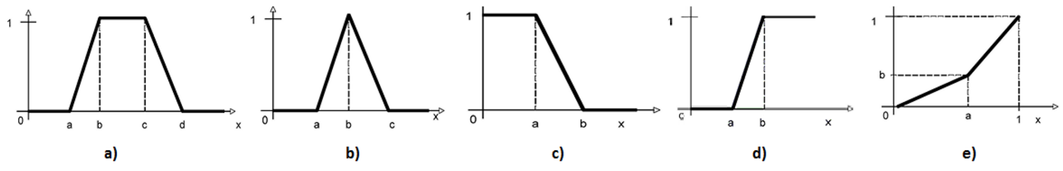


Figure 3.11: Membership functions for fuzzy data type definition (i.e., fuzzy concrete domains) and fuzzy modifier functions: a) Trapezoidal function; b) triangular function; c) left-shoulder function; d) right-shoulder function; and e) linear function.

A fuzzy concrete domain or fuzzy data type **D** is a pair $\langle \Delta_D, \Phi_D \rangle$ where Δ_D is a concrete interpretation domain, and Φ_D is a set of fuzzy concrete predicates **d** with an arity n and an interpretation $\mathbf{d}^{\mathcal{I}} : \Delta_D^n \rightarrow [0,1]$, which is an n-ary fuzzy relation over Δ_D[22]. For fuzzy data types, the functions allowed in Fuzzy OWL 2, defined over an interval $[k_1, k_2] \subseteq \mathbb{Q}$, are the following:

$\mathbf{d} \rightarrow$

$left(k_1, k_2, a, b)$ (fig. 3.11c) |

$right(k_1, k_2, a, b)$ (fig. 3.11d) |

$triangular(k_1, k_2, a, b, c)$ (fig. 3.11b) |

$trapezoidal(k_1, k_2, a, b, c, d)$ (fig. 3.11a) |

$linear(k_1, k_2, c)$ (fig. 3.11e) |

$mod(\boldsymbol{d})$

Considering the designed Human Activity ontology, we can identify concepts, data types, properties and relations that are susceptible of being fuzzy:

- Fuzzy Data Types. Some examples are:
  - *ShortDuration, MediumDuration, LongDuration*: Data types used to represent duration (*hasDuration*) of an *Activity*. In seconds.
  - *LowVolume, MediumVolume, HighVolume*: Data types representing the audio volume level (*hasVolume*) of a *Device*. In Decibels (dB).
  - *ClosedAperture, HalfAperture, OpenAperture*: Data types representing aperture angle (*hasAperture*) of a *Door, Window, Curtain* etc. In degrees.
  - *SmallCapacity, MediumCapacity, LargeCapacity*: Data types representing the capacity, in amount of people (*hasCapacity*), of a *Location*. An integer.
  - *LowHumidity, MediumHumidity, HighHumidity*: Data types representing humidity (*hasHumidity*) of a *Location*. In g/m^3 (IS).
  - *LowTemperature, MediumTemperature, HighTemperature*: Data types measuring centigrades degrees of temperature (*hasTemperature*) of a *Location, Environment, User, Object* or *Room*.
  - *LowLighting, MediumLighting, HighLighting*: Float-valued data types measuring lighting (*hasLighting*) of a *Location* or *Environment*, in lux (IS unit).
  - *LowNoiseLevel, MediumNoiseLevel, HighNoiseLevel*: Float-valued data types measuring noise level (*hasNoiseLevel*) of a *Location* or *Environment*, in dB.
  - *LowPressure, MediumPressure, HighPressure*: Float-valued data types measuring pressure (*hasPressure*) of a *Location* or *Environment*, in atmospheres (IS).
  - *FewPeople, MediumNPeople, ManyPeople*: Integer-valued data types measuring the number of attendants (*hasNAttendants*) to a *Conference, Seminar, Symposium, Workshop* or other *Event*.

The creation of a fuzzy data type in *Fuzzy OWL 2* plug-in visual editor results, e.g., for the data type *highTemperature*, in the following OWL 2 annotation:

```
<fuzzyOwl2 fuzzyType="datatype">
<Datatype type="rightshoulder" a="15.0" b="
    25.0" />
```

```
</fuzzyOwl2>
```

Listing 3.3: Annotation for the newly created fuzzy data type *highTemperature*

Classes in the original ontology can be transformed into fuzzy classes through fuzzy modifiers. Most of the environmental measurements become fuzzy data types. Their range is expressed through data range expressions such as *(double[>=-100.0] and double[<=100.0])* (e.g. for *hasTemperature* data property range). We use as range the referential set over which the fuzzy membership functions associated are defined. Examples of membership functions used are:

- *LowTemperature*: fuzzy data type with left shoulder membership function (a= -5, b = 5).
- *MediumTemperature*: fuzzy data type with trapezoidal membership function (a= 5, b = 10, c = 20, d = 25).
- *HighTemperature*: fuzzy data type with right shoulder membership function (a= 25, b = 30).

When creating a fuzzy role, an annotation property describing the type of the constructor and the value of their parameters are specified. Recursion is not allowed in the definition, and following the mapping in [22], only fuzzy modified roles are supported. The domain of the annotation will be any OWL 2 (object or data) property with the restriction that the modifier must be defined as a fuzzy modifier and the base fuzzy role has a different name than the annotated role.

- Fuzzy Concrete Roles (Fuzzy Data Properties):
  - Activity.*hasDuration*: Indicates the duration of an *Activity*, in seconds.
  - Activity.*hasTimeInBtw*: Indicates the time, in seconds, happening in between the associated *Activity* and the *Action/Activity* related to the first one through the relation *happensBefore*.
  - Action.*hasTimeInBtw*: Analogical to the previous case, for an *Action*.
  - Device.*hasVolume*: Indicates the volume, in dB, of a *Device* with audio capability (computer, radio, TV...).
  - (Window OR Curtain OR Door).*hasAperture*: Indicates the angle of aperture of a *Window, Curtain* or *Door*, in degrees.
  - Location.*hasCapacity*: Specifies the capacity, in amount of people, that a *Location* can host.
  - Location.*hasHumidity*: Indicates the humidity of a *Location*, in g/m^3 (IS).

  - Location.*hasTemperature*: Specifies the temperature, in centigrade degrees, of a *Location, Room, Environment, GenericUser* or *Object.*
  - Location.*hasLighting*: Light level or illuminance (luminous flux incident on a surface, per unit area) indicates the *hasLighting* property, in lux float-units, of a *Location.*
  - Building.*hasNoiseLevel*: Indicates the noise level of an *Environment* or *Location* (*Building, Room*, etc.)
  - Environment.*hasPressure*: Specifies the air pressure, in atmospheres, of a *Location* or *Environment.*
  - Conference.*hasNAttendants*: Specifies the number of attendants of an *Event*, *Conference, Symposium, Seminar, Meeting*, etc.

  A fuzzy concrete role is defined, in Fuzzy OWL 2, by setting its range data type to a previously defined fuzzy data type.

- Fuzzy Abstract Roles (Fuzzy Object Properties):
  - Thing-*isInLocation*-Location: Represents the location of anything (a *Thing*) in a given *Location.* A fuzzy degree can represent proximity.
  - User-*attendsEvent*-Event: Identifies a given *User* who attends an *Event*, which can belong to a *Calendar.*
  - Travel-*toLocation*-Location: Characterizes the activity *Travel* by specifying to what destiny *Location* it refers to.
  - Activity-*happensInLocation*-Location: Associates an *Activity* with a *Location* to indicate where it occurs.
  - Action-*actionAppliesTo*-Thing: Indicates the object over which an *Action* or *Activity* directly falls or acts over. E.g., *WalkBy* activity applies to *Corridor* if a user walks by a corridor.
  - GenericUser-*hasPersonalStatus*-PersonalStatus: Indicates a personal status of any *User* in certain moment. *PersonalStatus* class is specialized into: *Available, Away, Busy, OnHoliday, OnLeave.*
  - Activity-*involvesAction*-Action: Relates an *Activity* with the *Actions* that it involves or requires. E.g., *DoPresentation* activity involves actions *SetLaptopOn*, *SetProjectorOn*, *StandUp* and *Talk.*
  - GenericUser-*performsAction*-Action: Specifies which *User* realizes an *Action.* Fuzzy degree can represent that who performs the action is "unknown".
  - GenericUser-*performsActivity*-Activity: Specifies which *User* performs a given *Activity.*

  - Thing-*isNearTo*-Thing (e.g. User-*isNearTo*–Office): Specifies closeness of an entity (*Object, User, Location,...*) to another one.
  - User-*hasFace*-Face: Associates a *User* with a *Face* image. A user can have associated different faces, depending of the situation where they are acquired. Useful in face recognition techniques.
  - (Action OR Activity)-*happensBefore*-(Action OR Activity): Specifies temporal order constraints among *Actions* and/or *Activities*, associated to the *RegularAutomaton* class.

- Fuzzy Modifiers and Fuzzy Modified Data Types: We define the fuzzy modifiers *very* as *linear(0.85)* and *barely* as *linear(0.15)*. Given the object property *isNearTo(Thing, Thing)*, the fuzzy modifier *very* can be used to define new fuzzy properties such as *very(isNearTo)* (i.e., *isVeryNearTo*), to relate objects that are not close to each other, but very close. Other examples of fuzzy modified roles are *very(isFarFrom)* (i.e., *isVeryFarFrom*), *barely(hasDuration)* (i.e., *hasBarelyDuration*), or fuzzy modified data types such as *very(HighTemperature)* (i.e. *veryHighTemperature*), *very(LargeCapacity)* (i.e., *veryLargeCapacity*).

- Fuzzy Axioms or Assertions: In practice, expressing degrees of truth will be reflected on real time when assertions or axioms, in general, are expressed about concrete individuals (i.e., instances populating the ontology). Examples of fuzzy axioms can be assertions as: the *User* individual *Juan hasWorkPosition Professor* with degree 0.7 (if he is associate professor, but not titular professor).

In the graphical model, uncertainty is modelled in a way such that the user does not have to worry much about fuzzy degrees of truth, but rather choose data types that express different levels or degrees of truth. For instance, we model low, medium and high values of temperature (*lowTemperature*, *mediumTemperature* and *HighTemperature*) through underlying, and invisible to the end-user, membership functions (as explained in section 3.7.1). Linguistic labels make the editing of the rule closer to natural language and also closer to any kind of user, not requiring of him technical knowledge. An example using a linguistic label is in figure 3.12 to indicate, in a vague way, an office's temperature. A degree within the interval (0, 1] can, however, be also provided for more precise assertions or axioms. For example, when we would want to find if there exists any user (through his phone) which is close enough (with certainty >0.8) to the meeting room, we could express the antecedent IF of the rule as figure 3.13 shows. The user should be able to edit a red frame over any arc relation containing a source and destination nodes.

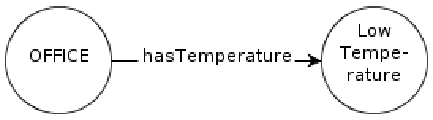


Figure 3.12: Example of linguistic label for a Fuzzy data type representing different degrees of Temperature.

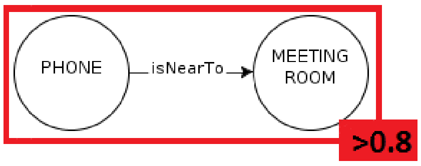


Figure 3.13: Example of axiom assertion to express how much degree of truth we need in our axiom for our rule to fire.

A summary of the OWL 2 to Fuzzy OWL 2 mapping is in table 3.2 together with equivalent graphical representations in the proposed GUI. Providing a fuzzy regular automaton, as part of the whole process of integration of the regular automaton behaviour model into the ontology, remains as part of the future work.

Once the fuzzy ontology is created in an ontology editor, Protégé 4.1 in our case, it needs to be translated into a language supported by a fuzzy ontology reasoner. Fuzzy OWL 2 parsers convert Fuzzy OWL 2 ontologies into *DeLorean*[28][20] and *FuzzyDL*[29][21] reasoners' syntax. *DeLorean* is a fuzzy rough DL reasoner that supports fuzzy rough extensions of the fuzzy DLs $\mathcal{SROIQ}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$ (equivalent to OWL and OWL 2) and it is based on a discretization of the fuzzy ontology using α-cuts [20]. *FuzzyDL* reasoner's main features are the extension of the classical description logic $\mathcal{SHIF}(\mathbf{D})$ to the fuzzy case, it allows fuzzy concepts with left-shoulder, right-shoulder, triangular and trapezoidal membership functions, supports concept modifiers and general inclusion axioms among other features. Since in our case, we do not specify degrees of truth initially on the ontology, due to the fact that these need to be asserted and introduced in execution time, α-cuts can not be calculated until that time. Furthermore, our fuzzy ontology is characterized mainly by the presence of fuzzy data types, which are specially tackled by *FuzzyDL* reasoner. Therefore, for our concrete fuzzy ontology, we use the reasoner *FuzzyDL*, for which some issues need to be addressed.

First of all, *FuzzyDL* discards asymmetry axioms and role irreflexivity. In practice, there is no problem since our roles are defined with domain and range for each property, and these are disjoint classes. Therefore, it will not exist symmetry nor reflexivity. Second, cardinality restrictions are discarded. This does not become a problem since: 1) In the class *Event*, *belongsToCalendar min 1 Thing* is equivalent to *belongsToCalendar some Thing.* 2) In the class

[28]DeLorean Reasoner: http://webdiis.unizar.es/ fbobillo/delorean

[29]FuzzyDL Reasoner: http://gaia.isti.cnr.it/ straccia/software/fuzzyDL/fuzzyDL.html

User, *hasNUsers exactly 1 int* can be substituted by *hasNUsers some int.* This fact, added to having *hasNUsers* as functional property, makes the number of relations *hasNUsers* to be exactly 1. Other features such as reasoning with property values of type *boolean* or *datetime*, not yet supported by *FuzzyDL*, can be adapted with crisp functions when needed. And modified properties require further development of the reasoner.

Gurobi 4.5.2 was required [30] to use *FuzzyDL.* Finally, *FuzzyDL* allows realizing query expressions such as the following:

- *(min-instance? NataliasNokiaN8 Phone)*
  Determines the minimal degree to which individual *NataliasNokiaN8* is an instance of concept *Phone.*

- *(all-instances? C)*
  Same as previous query, for every individual of class *C* in the KB.

- *(max-related? Natalia JohanLiliusOffice isInLocation)*
  Determines the maximal degree to which individual pair *(Natalia, JohanLiliusOffice)* is an instance of role name *isInLocation.*

- *(min-subs? VeryFullCapacity FullCapacity)*
  Determines the minimal degree to which concept *VeryFullCapacity* subsumes concept *FullCapacity.* Optionally, an individual and Lukasiewicz, Gödel or Kleene-Dienes implications can be used for this type of query [21].

[30]Through getting a trial/academic license first for *Gurobi 5.0*

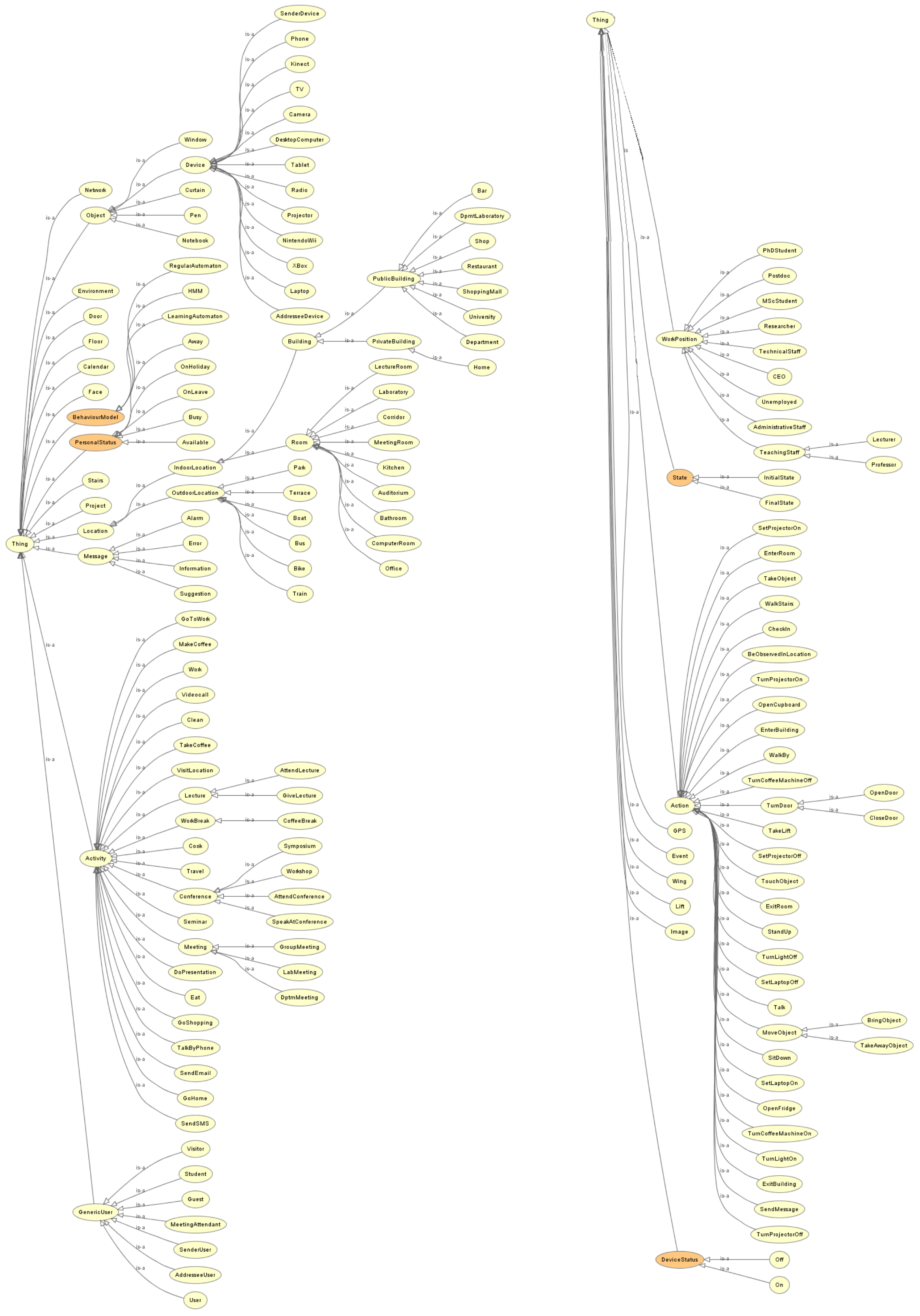


Figure 3.3: Complete Human Activity Ontology proposed (asserted model)

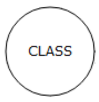
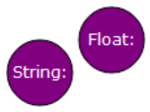
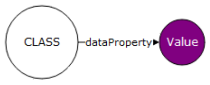
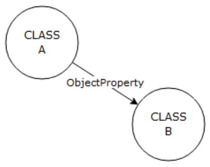
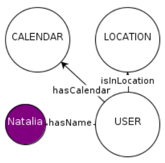
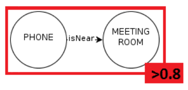
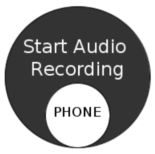
***OWL2* mapping to *Fuzzy OWL2* and Visual Model**			
OWL2	**Fuzzy OWL2**	**Visual Model**	**Look**
Class	Fuzzy Class (Fuzzy Concept)	White node	
Data type	Fuzzy Data type	Purple small node	
Data Property	Fuzzy Data Property (Fuzzy Concrete Role)	Relation arrow to a small purple node containing its value	
Object Property	Fuzzy Object Property (Fuzzy Abstract Role)	Relation arrow to white node acting as its range class	
Individual	Individual	Conceptualized by a class' white node with set of data/object property values	 CALENDAR LOCATION hasCalendar isInLocation Natalia hasName USER
Axiom (e.g. Concept Assertion)	Fuzzy Axiom	Set of nodes and arches in a user-drawn delimiter red frame which establishes degree of truth	
Application or Service	-	Grey large node with as many defined class nodes inside as parameters are required	

Table 3.2: *OWL2* to *Fuzzy OWL2* mapping and transformation from an RDF graph to a visual end-user representation

Chapter 4

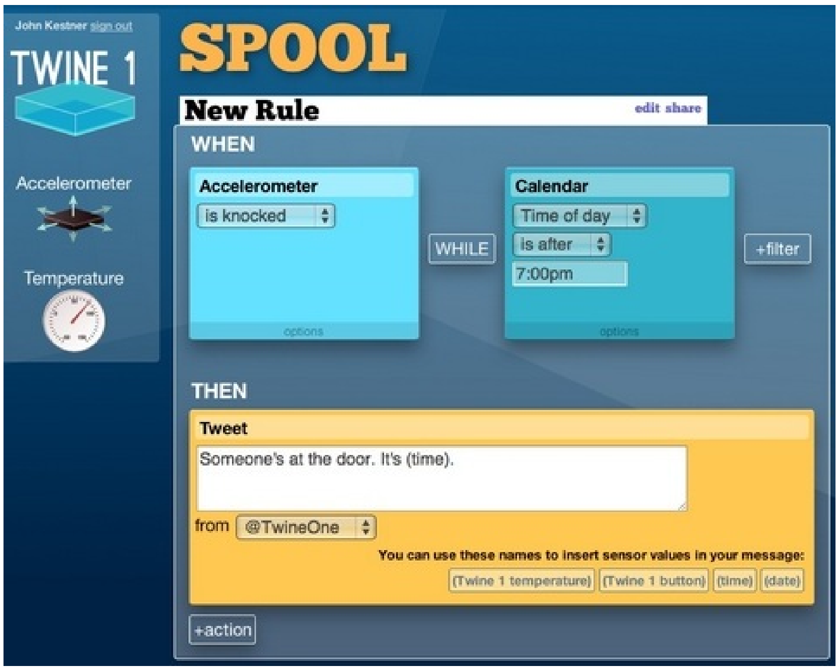
Discussion, Conclusions and Future Work

This chapter discusses what this work consisted on and the extracted conclusions after its realization. Finally, we discuss what our future work will deal with.

4.1 Discussion

A state-of-the-art on human activity detection and recognition was presented. Different techniques, mostly data-driven approaches were found in the literature. However, knowledge-driven techniques are gaining increasing interest [122]. The Semantic Web is evolving from a data modelling medium to a computational medium [123]. We make a graphical model proposal towards the ideal of extending the read-only Semantic Web with algorithmic and machine representations to achieve a Web of evolving computations. Ontology-based activity recognition has some drawbacks, e.g., it requires knowledge engineering skills to model the domain as well as expressiveness limitations are found in OWL DL, mainly related with the lack of support for temporal reasoning. With OWL, it is not possible to perform interval-based (i.e. overlapping) temporal reasoning, which is crucial for capturing complex temporal relations in between activities. Last, ontological reasoning can be computationally expensive [122]. To overcome these issues we propose regular automata to model temporal dependencies among actions that compose an activity (which in turn, can compose a human behaviour). This is an approach suitable to easier achieve interoperability, abstraction and modularity, with the added advantages by ontology-based reasoning (discussed in chapter 3 and [32]).

The closest case to ours, found in the literature, is perhaps the COSAR context aggregation and reasoning middleware. Their *Derivation of Possible Activities* algorithm, executed by an off-line ontological reasoning module, takes as input an empty ABox and the terminological part TBox to output a matrix of symbolic locations and their correspondence to inferred human activities [121]. Even if this approach uses hybrid ontological/statistical rea-



New Rule

WHEN

Accelerometer: is knocked

Calendar: Time of day, is after, 7:00pm

THEN

Tweet: Someone's at the door. It's (time).

from @TwineOne

You can use these names to insert sensor values in your message: (Twine 1 temperature) (Twine 1 button) (time) (date)

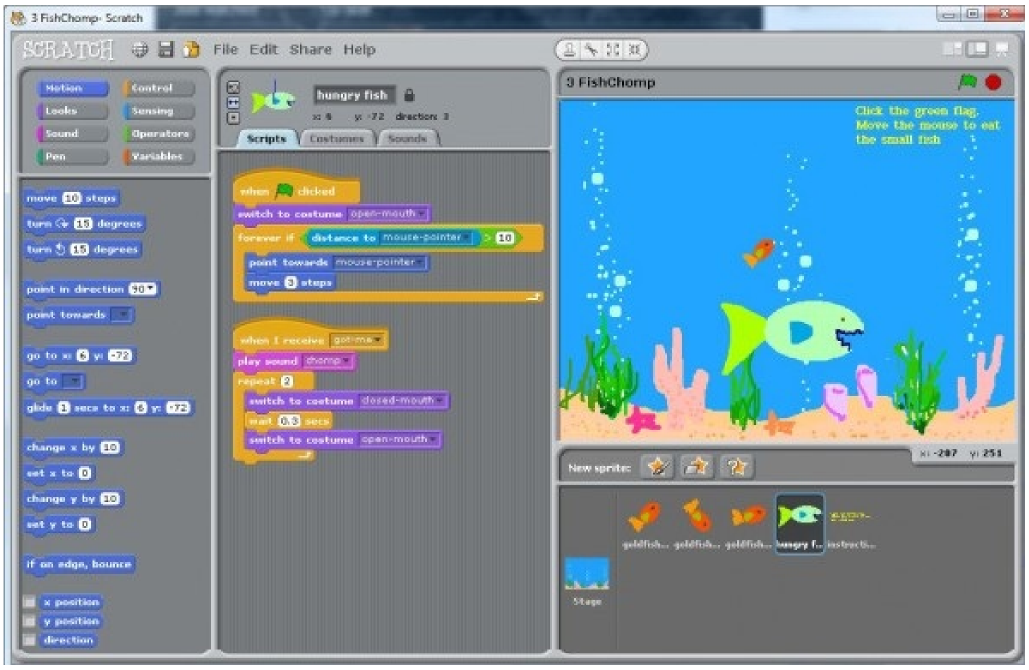
Figure 4.1: Twine Spool rule editor (http://supermechanical.com/twine/).

soners, our regular automata model decouples activities with locations, as well as with the general support for uncertainty, since a single location, in a (work/office) environment can allow to do multiple different tasks.

We also focus on the challenge of providing the end-user the possibility of rapidly prototype the behaviour of a SS, abstracting away technical details. When presenting a semantic SS to the user, the interface model, even if simplified, must adhere to the semantic formal model. Domain specific languages demonstrate support in this abstraction for integration of metamodels using ontologies (e.g. in [144]). Concerning end-user GUI and rule editors, some good examples that simplify the tasks to the user when creating their own services/applications, through simple rules, are *If This Then That* [1] (fig. 4.3) for online social services, *Twine* [2] for sensor interaction applications (fig. 4.1) or Valpas [120] intelligent environment for assisted living (fig. 4.4). However, they lack semantic capabilities. Regarding the user experience, another approach to aid the user interacting with semantic data consists of suggesting rules to the user by using his browsed RDF paths, for rewriting these as WQL queries [85]. This is one of the new paradigms to be explored with SPARQL 1.1, for which the transitive closure will allow concepts such as repetition, recursion and path queries.

The survey of programming environments for novice programmers [78] shows ways to lower the barriers to programming. Our visual model proposal, taking inspiration from the *Scratch* programming framework [119] (figure 4.2), aids customizing behaviour in SSs; but it could as well serve as

[1]http://ifttt.com
[2]http://supermechanical.com/twine/



```
when [flag] clicked
switch to costume open-mouth
forever if distance to mouse-pointer > 10
    point towards mouse-pointer
    move 3 steps

when I receive got-me
play sound chomp
repeat 2
    switch to costume closed-mouth
    wait 0.3 secs
    switch to costume open-mouth
```

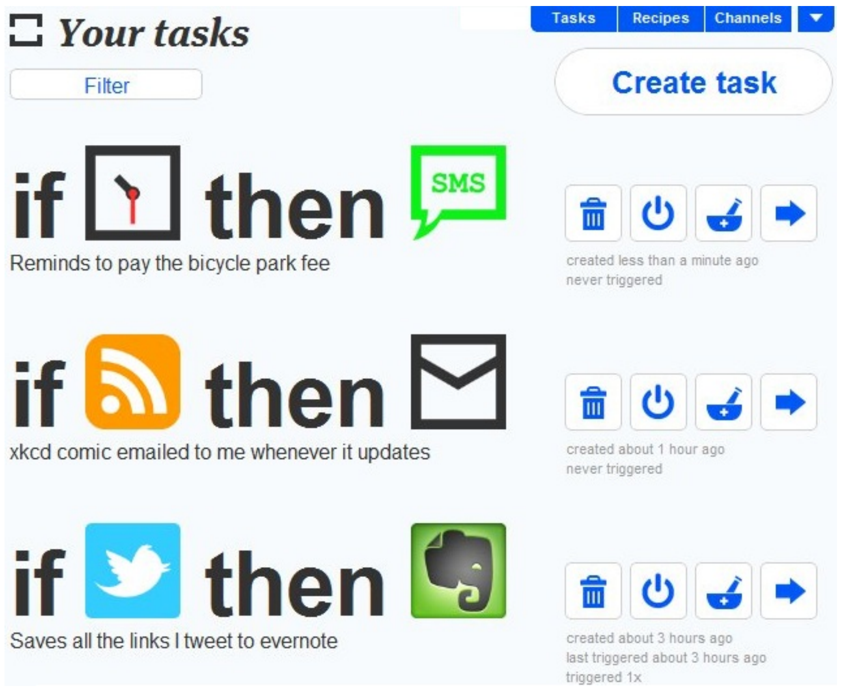
Click the green flag. Move the mouse to eat the small fish

Figure 4.2: Scratch [119] Programming Interface.

an educative interface for teaching basic SW technologies and logic programming intuitively. However, our scheme's main added value proposes a broader domain view than *Scratch*, with a general purpose semantic metamodel, supporting query federation and promoting Linked Data web services for rapid development of mash-up applications. Our end-user model pushes the devised evolution of the SW from a data modelling to a computational medium [123] by bringing the advantages of the Semantic Web closer to any non expert user. Our contribution follows visual language design guidelines [58] for being an intuitive, or *well matched*, visual language, i.e., "its representation clearly captures the key features of the represented artefact, in addition to simplify various desired reasoning tasks". The framework can serve as a general purpose tool for a broad area of domains, e.g., monitoring or automating activities in health care, assisted living, home automation, office domain, industry processes and a large etc.

4.2 Conclusions

The work presented belongs to the ongoing doctoral research, as part of the final project of the doctoral master on *Soft Computing and Intelligent Systems*. A semantic model to represent human activity context for recognition was built, with several contributions; first, a human activity context ontology to specify actions, activities and behaviours, adapted to the office and work domain. Second, we provide support for imprecision, vagueness and uncertainty, as common characteristics of changing contexts, by transforming



Reminds to pay the bicycle park fee

created less than a minute ago
never triggered

xkcd comic emailed to me whenever it updates

created about 1 hour ago
never triggered

Saves all the links I tweet to evernote

created about 3 hours ago
last triggered about 3 hours ago
triggered 1x

Figure 4.3: Example of IFTTT rule recipes (http://ifttt.com).

the ontology into a fuzzy one. Finally, two different development tools were proposed as a way to customize the behaviour, through simple rules, of a Smart Space: a) *PythonRules* middleware approach, for which learning OWL or SPARQL is not required to achieve device interoperability and interaction in the SS. b) A graphical model to represent and interact with human behaviours. For the latter, the created fuzzy ontology, *FuzzyHumanActivity.owl*, represents, detects and recognizes human actions, activities and behaviours. The current version of the ontology, validated with the reasoners HermiT 1.3.6, Pellet 2.3.0 (2.2.0 Protégé plug-in) and FuzzyDL 1.1, consists of 172 classes, 61 object properties, 47 data properties and 7 test individuals, within $\mathcal{SHIF}(\mathbf{D})$ DL expressivity. The graphical model prototype design needs further usability studies before implementation, since it will evolve at the same time as AI models for behaviour disclosure are developed and integrated with the ontological model.

4.3 Future Work

Our future plans include developing the end-user framework as a web browser application, accessible from any mobile device, to allow any kind of user to establish simple rules that model context-aware behaviour in Smart Spaces. Integration of proper and scalable subscription to changes mechanisms, as well as fulfilment of standards, is required. Rules will be able to be shared by users or customized according to the user preferences and everyday routines. First, for that purpose, behaviour models will be further developed to

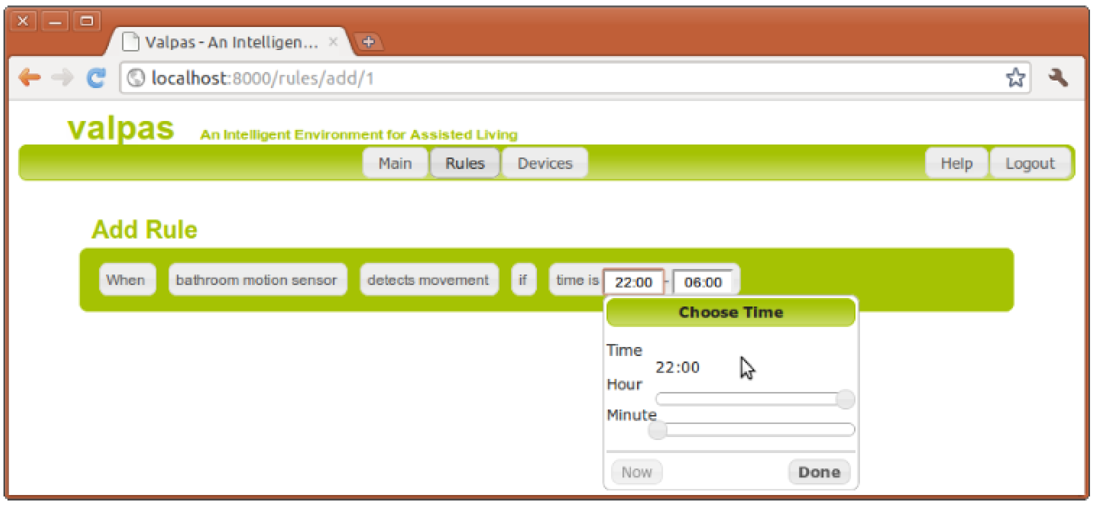


Figure 4.4: Example of ECA rule editing in *Valpas*, an intelligent environment for Assisted Living [120].

support fuzzy modelling, e.g., actions dependencies, changes in activities, or general indeterminism. Providing an ontology representation of fuzzy regular automata is part of the plan to fully support natural imprecision in behaviour modelling. Furthermore, different AI behaviour models will be studied in order to take advantage of both data-driven mechanisms and knowledge-driven techniques, such as, HMMs or Learning Automata[126, 127]. These will be studied to better mimic human behaviour adaptability, support for changes, parallelism of activities and social interaction. As a natural step after creating a fuzzy ontology, future work will also consider the support for imprecision in a platform such as Smart-M3, extending *PythonRules*, as well as adding fuzziness into our graphical model.

Bibliography

[1] Smart-M3 Ontology to Python API Generator: http://sourceforge.net/projects/smart-m3/files/smart-m3-ontology_to_python-api_generator_v0.9.1beta.tar.gz/.

[2] SPIN [online]: http://www.spinrdf.org/.

[3] SPIN<->SPARQL Converter [online]: http://sparqlpedia.org/spinrdfconverter.html.

[4] SWRL: A Semantic Web Rule Language Combining OWL and RuleML [online]: http://www.w3.org/submission/swrl/.

[5] Tagged world project. `http://taggedworld.jp/`.

[6] Eggen B. Aarts, E. Ambient intelligence in homelab., 2002.

[7] Bessam Abdualrazak, Yasir Malik, and Hen-I Yang. A taxonomy driven approach towards evaluating pervasive computing system. In *Proceedings of the Aging friendly technology for health and independence, and 8th international conference on Smart homes and health telematics*, ICOST'10, pages 32–42, Berlin, Heidelberg, 2010. Springer-Verlag.

[8] G.D. Abowd. Classroom 2000: An experiment with the instrumentation of a living educational environment. *IBM Systems Journal*, 38(4):508–530, 1999.

[9] G. Acampora and V. Loia. Fuzzy control interoperability and scalability for adaptive domotic framework. *IEEE Transactions on Industrial Informatics*, 1:97 – 111, 2005.

[10] A. Adler and R. Davis. Speech and sketching for multimodal design. In *Proceedings of the 9th International Conference on Intelligent User Interfaces*, page 214–216, 2004.

[11] Safdar Ali and Stephan Kiefer. A micro OWL DL reasoner for ambient intelligent devices. In *Proceedings of the 4th International Conference on Advances in Grid and Pervasive Computing*, GPC '09, pages 305–316, Berlin, Heidelberg, 2009. Springer-Verlag.

[12] J.F. Allen and G. Ferguson. Actions and events in interval temporal logic. *Journal of Logic and Computation*, 4:531–579, 1994.

[13] M. Amoretti, F. Wientapper, F. Furfari, S. Lenzi, and S. Chessa. Sensor data fusion for activity monitoring in ambient assisted living environments. In Stephen Hailes, Sabrina Sicari, George Roussos, Ozgur Akan, Paolo Bellavista, Jiannong Cao, Falko Dressler, Domenico Ferrari, Mario Gerla, Hisashi Kobayashi, Sergio Palazzo, Sartaj Sahni, Xuemin (Sherman) Shen, Mircea Stan, Jia Xiaohua, Albert Zomaya, and Geoffrey Coulson, editors, *Sensor Systems and Software*, volume 24 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 206–221. Springer Berlin Heidelberg, 2010.

[14] D. Anderson, M. Ros, J. Keller, M.P. Cuéllar, M. Popescu, M. Delgado, and A. Vila. Similarity measure for anomaly detection and comparing human behaviors. *International Journal of Intelligent Systems*, in press:–, 2012.

[15] Louis Atallah and Guang-Zhong Yang. The use of pervasive sensing for behaviour profiling , a survey. *Pervasive and Mobile Computing*, 5(5):447 – 464, 2009.

[16] J. Augusto. Ambient intelligence: The confluence of pervasive computing and artificial intelligence. In A. Schuster, editor, *Intelligent Computing Everywhere*, pages 213–234. Springer Berlin Heidelberg, 2007.

[17] Matthias Baldauf, Schahram Dustdar, and Florian Rosenberg. A survey on context-aware systems. *Int. J. Ad Hoc Ubiquitous Comput.*, 2(4):263–277, June 2007.

[18] Matthias Baldauf, Schahram Dustdar, and Florian Rosenberg. A Survey on Context-Aware systems. In *International Journal of Ad Hoc and Ubiquitous Computing*, volume 2, 2007.

[19] Claudio Bettini, Linda Pareschi, and Daniele Riboni. Efficient profile aggregation and policy evaluation in a middleware for adaptive mobile applications. *Pervasive and Mobile Computing*, 4(5):697–718, 2008.

[20] Fernando Bobillo. *Managing Vagueness in Ontologies.* PhD thesis, 2008.

[21] Fernando Bobillo and Umberto Straccia. fuzzyDL: An expressive fuzzy description logic reasoner. In *2008 International Conference on Fuzzy Systems (FUZZ-08)*, pages 923–930. IEEE Computer Society, 2008.

[22] Fernando Bobillo and Umberto Straccia. Fuzzy ontology representation using OWL 2. *Int. J. Approx. Reasoning*, 52(7):1073–1094, October 2011.

[23] J. Boger, J. Hoey, P. Poupart, C. Boutilier, G. Fernie, and A. Mihailidis. A planning system based on markov decision processes to guide people with dementia through activities of daily living. *IEEE Transactions on Information Technology in Biomedicine*, 10:323 –333, 2006.

[24] O. Brdiczka, P. Reignier, and J. Crowley. Detecting individual activities from video in a smart home. In *Proceedings of the International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, page 363–370, 2007.

[25] Christer Carlsson, Matteo Brunelli, and József Mezei. Fuzzy ontologies and knowledge mobilisation: Turning amateurs into wine connoisseurs. In *FUZZ-IEEE*, pages 1–7. IEEE, 2010.

[26] Christer Carlsson, Matteo Brunelli, and József Mezei. Decision making with a fuzzy ontology. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, pages 1–10, 2011. 10.1007/s00500-011-0789-x.

[27] Alexandros Andre Chaaraoui, Pau Climent-Perez, and Francisco Florez-Revuelta. A review on vision techniques applied to human behaviour analysis for ambient-assisted living. *Expert Systems with Applications*, (0):–, 2012.

[28] A. Chapko, M. Grasle, A. Emrich, D. Werth, C. Rust, N. Laum, C. Lerche, and A. Weber. User-generated mobile services for health and fitness. In *Emerging Technologies Factory Automation (ETFA), 2011 IEEE 16th Conference on*, pages 1 –8, sept. 2011.

[29] G. Chelius, C. Braillon, M. Pasquier, N. Horvais, R.P. Gibollet, B. Espiau, and C.A. Coste. A wearable sensor network for gait analysis: A six-day experiment of running through the desert. *Mechatronics, IEEE/ASME Transactions on*, 16(5):878 –883, oct. 2011.

[30] Harry Chen, Tim Finin, and Anupam Joshi. An Ontology for context-aware pervasive computing environments. In *Proceedings of the Workshop on Ontologies in Agent Systems*, 2003.

[31] Harry Chen, Tim Finin, Anupam Joshi, Lalana Kagal, Filip Perich, and Dipanjan Chakraborty. Intelligent Agents Meet the Semantic Web in Smart Spaces. *IEEE Internet Computing*, November Oktober 2004:2–12, November 2004.

[32] Luke L. Chen and Jit Biswas. Activity recognition in smart homes. int. conf on advances in mobile computing and multimedia (momm2009), 2009.

[33] Wongun Choi, C. Pantofaru, and S. Savarese. Detecting and tracking people using an rgb-d camera via multiple detector fusion. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 1076 –1083, nov. 2011.

[34] Diane J. Cook, Juan C. Augusto, and Vikramaditya R. Jakkula. Ambient intelligence: Technologies, applications, and opportunities, 2007.

[35] Diane J Cook, Aaron Crandall, Geetika Singla, and Brian Thomas. Detection of Social Interaction in Smart Spaces. *Cybernetics and systems*, 41(2):90–104, February 2010.

[36] Diane J. Cook, Michael Youngblood, Edwin O. Heierman, III, Karthik Gopalratnam, Sira Rao, Andrey Litvin, and Farhan Khawaja. Mavhome: An agent-based smart home. In *PERCOM '03: Proceedings of the First IEEE International Conference on Pervasive Computing and Communications*, page 521, Washington, DC, USA, 2003. IEEE Computer Society.

[37] Oscar Corcho, Mariano Fernández-López, and Asunción Gómez-Pérez. Methodologies, tools and languages for building ontologies: where is their meeting point? *Data Knowl. Eng.*, 46(1):41–64, July 2003.

[38] S. Coyle, D. Morris, K.-T. Lau, D. Diamond, F. Di Francesco, N. Taccini, M.G. Trivella, D. Costanzo, P. Salvo, J.-A. Porchet, and J. Luprano. Textile sensors to measure sweat ph and sweat-rate during exercise. In *Pervasive Computing Technologies for Healthcare, 2009. PervasiveHealth 2009. 3rd International Conference on*, pages 1 –6, april 2009.

[39] Aaron Crandall and Diane J. Cook. *Learning Activity Models for Multiple Agents in a Smart Space*, page 751. 2010.

[40] S. Dashtinezhad, T. Nadeem, B. Dorohonceanu, C. Borcea, P. Kang, and L. Iftode. Trafficview: A driver assistant device for traffic monitoring based on car-to-car communication. In *Proceedings of the Vehicular Technology Conference*, page 2946–2950, 2004.

[41] Miguel Delgado, Maria Ros, and Amparo Vila. Correct behavior identification system in a tagged world. *Expert Systems with Applications*, 36(6):9899 – 9906, 2009.

[42] Alfredo D'Elia, Luca Roffia, Guido Zamagni, Fabio Vergari, Paolo Bellavista, Alessandra Toninelli, and Sandra Mattarozzi. Smart applications for the maintenance of large buildings: How to achieve ontology-based interoperability at the information level. In *Computers and Communications (ISCC), 2010 IEEE Symposium on*, pages 1077 –1082, june 2010.

[43] Anind K. Dey and Gregory D. Abowd. The Context Toolkit: Aiding the development of context-aware applications. In *Workshop on Software Eng. for Wearable and Pervasive Computing*, 2000.

[44] Sajal Das Diane Cook. *Smart Environments: Technology, Protocols and Applications*. 2004.

[45] F. Doctor, H. Hagras, and V. Callaghan. A fuzzy embedded agent-based approach for realizing ambient intelligence in intelligent inhabited environments. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 35(1):55–65, 2005.

[46] Rose-Marie Droes, Maurice Mulvenna, Chris Nugent, Dewar Finlay, Mark Donnelly, Marius Mikalsen, Stale Walderhaug, Tim van Kasteren, Ben Krose, Stefano Puglia, Fabio Scanu, Marco Oreste Migliori, Erdem Ucar, Cenk Atlig, Yilmaz Kilicaslan, Ozlem Ucar, and Jennifer Hou. Healthcare systems and other applications. *IEEE Pervasive Computing*, 6(1):59–63, 2007.

[47] Saibal Dutta, Amitava Chatterjee, and Sugata Munshi. An automated hierarchical gait pattern identification tool employing cross-correlation-based feature extraction and recurrent neural network based classification. *Expert Systems*, 26(2):202–217, 2009.

[48] Dejene Ejigu, Marian Scuturici, and Lionel Brunie. An ontology-based approach to context modeling and reasoning in pervasive computing. In *Pervasive Computing and Communications Workshops, 2007. PerCom Workshops '07. Fifth Annual IEEE International Conference on*, pages 14 –19, march 2007.

[49] M. Ermes, J. Parkka, J. Mantyjarvi, and I. Korhonen. Detection of daily activities and sports with wearable sensors in controlled and uncontrolled conditions. *Information Technology in Biomedicine, IEEE Transactions on*, 12(1):20 –26, jan. 2008.

[50] Jr. Forney, G.D. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268 – 278, march 1973.

[51] A. Fox, B. Johanson, P. Hanrahan, and T. Winograd. Integrating information appliances into an interactive space. *IEEE Computer Graphics and Applications*, 20(3):54–65, 2000.

[52] D. Franklin. Cooperating with people: The intelligent classroom. In *Proceedings of the National Conference on Artificial Intelligence*, page 555–560, 1998.

[53] C. Le Gal. Smart offices. In *Smart Environments: Technology, Protocols, and Applications*, 2004.

[54] Maureen Schmitter-Edgecombe Greetika Singla, Diana J. Cook. Recognizing independent and joint activities among multiple residents in smart environments. *October*, 1(1):57–63, 2010.

[55] Tao Gu, Hung Keng Pung, and Da Qing Zhang. A middleware for building context-aware mobile services. In *Proceedings of IEEE Vehicular Technology Conference*, 2004.

[56] E. Guenterberg, H. Ghasemzadeh, and R. Jafari. A distributed hidden markov model for fine-grained annotation in body sensor networks. In *Wearable and Implantable Body Sensor Networks, 2009. BSN 2009. Sixth International Workshop on*, pages 339 –344, june 2009.

[57] E. Guenterberg, A.Y. Yang, H. Ghasemzadeh, R. Jafari, R. Bajcsy, and S.S. Sastry. A method for extracting temporal parameters based on hidden markov models in body sensor networks with inertial sensors. *Information Technology in Biomedicine, IEEE Transactions on*, 13(6):1019 –1030, nov. 2009.

[58] C. Gurr. Visualizing a logic of dependability arguments. In P. Cox, A. Fish, and J. Howse, editors, *Visual Languages and Logic Workshop (VLL 2007)*, volume 274, pages 97–109, 2007. Workshop within IEEE Symposium on Visual Languages and Human Centric Computing VL/HCC 07.

[59] Corin Gurr. Computational diagrammatics: diagrams and structure. In Denis Besnard, Cristina Gacek, and Cliff B. Jones, editors, *Structure for Dependability: Computer-Based Systems from an Interdisciplinary Perspective*. Springer London, 2006.

[60] H. Hagras, V. Callaghan, M. Colley, G. Clarke, A. Pounds-Cornish, and H. Duman. Creating an ambient-intelligence environment using embedded agents. *Intelligent Systems, IEEE*, 19(6):12 – 20, 2004.

[61] Ramón Hervás, José Bravo, and Jesús Fontecha. A context model based on ontological languages: a proposal for information visualization. *j-jucs*, 16(12):1539–1555, jun 2010.

[62] Juhani Hirvonen, Teemu Tommila, Antti Pakonen, Christer Carlsson, Mario Fedrizzi, and Robert Fullér. Fuzzy keyword ontology for annotating and searching event reports. In *KEOD*, pages 251–256, 2010.

[63] Jukka Honkola, Hannu Laine, Ronald Brown, and Olli Tyrkkö. Smart-M3 interoperability platform release. http://sourceforge.net/projects/smart-m3/, October 2009.

[64] J.E. Hopcroft, R. Motwani, and J.D. Ullman. Introduction to automata theory, languages, and computation, 2001.

[65] Matthew Horridge. A Practical Guide To Building OWL Ontologies Using Protégé 4 and CO-ODE Tools, Edition 1.3, 2011.

[66] Tâm Huynh, Mario Fritz, and Bernt Schiele. Discovery of activity patterns using topic models. In *Proceedings of the 10th international conference on Ubiquitous computing*, UbiComp '08, pages 10–19, New York, NY, USA, 2008. ACM.

[67] ISTAG. Scenarios for ambient intelligence in 2010, 2001.

[68] E. Horvitz J. Letchner, J. Krumm. Trip router with individualized preferences: Incorporating personalization into route planning. In *Eighteenth Conference on Innovative Applications of Artificial Intelligence*, page 243–260, 2006.

[69] V. Jakkula, A. Crandall, and D.J. Cook. Knowledge discovery in entity-based smart environment resident data using temporal relations-based data mining. In *Proceedings of the ICDM Workshop on Spatial and Spatio-Temporal Data Mining*, pages 531–579, 2007.

[70] V.R. Jakkula, A.S. Crandall, and D.J. Cook. Enhancing anomaly detection using temporal pattern discovery. In A. D. Kameas, V. Callagan, H. Hagras, M. Weber, and W. Minker, editors, *Advanced Intelligent Environments*, pages 175–194. Springer Berlin Heidelberg, 2009.

[71] A. Jara, M. Zamora-Izquierdo, and A. Gomez-Skarmeta. An ambient assisted living system for telemedicine with detection of symptoms. In José Mira, José Ferrández, José Álvarez, Félix de la Paz, and F. Toledo, editors, *Bioinspired Applications in Artificial and Natural Computation*, volume 5602 of *Lecture Notes in Computer Science*, pages 75–84. Springer Berlin / Heidelberg, 2009.

[72] He Jian, Dong Yumin, Zhang Yong, and Huang Zhangqin. Creating an ambient-intelligence environment using multi-agent system. In *Embedded Software and Systems Symposia, 2008. ICESS Symposia '08. International Conference on*, pages 253 –258, july 2008.

[73] M.M. Jordanova. eHealth: from space medicine to civil healthcare. In *Recent Advances in Space Technologies, 2005. RAST 2005. Proceedings of 2nd International Conference on*, pages 739 – 743, june 2005.

[74] F. Scapolo J. Leijten-JC. Burgelman K. Ducatel, M. Bogdanowicz. Scenarios for ambient intelligence in 2010., 2001.

[75] Andre Kaustell, M. Mohsin Saleemi, Thomas Rosqvist, Juuso Jokiniemi, Johan Lilius, and Ivan Porres. Framework for Smart Space Application Development. In *Proceedings of the International Workshop on Semantic Interoperability, IWSI 2011*, 2011.

[76] Justin J. Kavanagh, Steven Morrison, Daniel A. James, and Rod Barrett. Reliability of segmental accelerations measured using a new wireless gait analysis system. *Journal of Biomechanics*, 39(15):2863 – 2872, 2006.

[77] Mackintosh R Keen PGW. *The freedom economy: gaining the mcommerce edge in the era of the wireless internet.* 2001.

[78] Caitlin Kelleher. Lowering the Barriers to Programming : a survey of programming environments and languages for novice programmers. *Human Factors*, 2003.

[79] Eunju Kim, Sumi Helal, and D. Cook. Human activity recognition and pattern discovery. *Pervasive Computing, IEEE*, 9(1):48 –53, jan.-march 2010.

[80] Jung Soo Kim, Beom Oh Kim, and Kwang Suk Park. Development of HIHM (Home Integrated Health Monitor) for Ubiquitous Home Healthcare. In *Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE*, pages 363 –365, aug. 2007.

[81] B. Krose, van T. Kasteren, C. Gibson, and van den T. Dool. Care: Context awareness in residences for elderly. In *Proceedings of ISG'08: The 6th International Conference of the International Society for Gerontechnology*, pages 101–105, 2008.

[82] J. Krumm and E. Horvitz. Predestination: Inferring destinations from partial trajectories. In *UbiComp 2006: Eighth International Conference on Ubiquitous Computing*, page 243–260, 2006.

[83] Chin-Feng Lai, Yueh-Min Huang, Jong Hyuk Park, and Han-Chieh Chao. Adaptive body posture analysis for elderly-falling detection with multisensors. *Intelligent Systems, IEEE*, 25(2):20 –30, march-april 2010.

[84] E. Lank, A. Ichnowski, and S. Khatri. Zero knowledge access to a smart classroom environment. In *Proceedings of the Workshop on Ubiquitous Display Environments*, 2004.

[85] Ora Lassila. Generating rewrite rules by browsing RDF data. In *RuleML*, pages 51–57, 2006.

[86] Peter Matetelki Laszlo Kovacs and Balazs Pataki. Service-oriented context-aware framework. In *Young Researchers Workshop on Service-Oriented Computing*, 2009.

[87] Jiayang Liu, Zhen Wang, Lin Zhong, J. Wickramasuriya, and V. Vasudevan. uwave: Accelerometer-based personalized gesture recognition and its applications. In *Pervasive Computing and Communications, 2009. PerCom 2009. IEEE International Conference on*, pages 1 –9, march 2009.

[88] Zongyi Liu and Sudeep Sarkar. Improved gait recognition by gait dynamics normalization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28:2006, 2006.

[89] E. Montiel-Ponsoda-B. Villazón-Terrazas Z. Yufei G. Aguado de Cea A. García M. Fernández-López A. Gómez-Pérez M. Espinoza M. Sabou M. C. Suárez-Figueroa, K. Dellschaft. NeOn deliverable d5.4.1. NeOn methodology for building contextualized ontology networks. NeOn project. 2008.

[90] G. Marreiros, R. Santos, C. Ramos, J. Neves, P. Novais, J. Machado, and J. Bulas-Cruz. Ambient intelligence in emotion based ubiquitous decision making. In *Proceedings of the Second Workshop on Artificial Intelligence Techniques for Ambient Intelligence*, pages 86–91, 2007.

[91] U. Maurer, A. Smailagic, D.P. Siewiorek, and M. Deisher. Activity recognition and monitoring using multiple sensors on different body positions. In *Wearable and Implantable Body Sensor Networks, 2006. BSN 2006. International Workshop on*, pages 4 pp. –116, april 2006.

[92] Xiaoning Meng, Ka Keung Lee, and Yangsheng Xu. Human driving behavior recognition based on hidden markov models. In *Robotics and Biomimetics, 2006. ROBIO '06. IEEE International Conference on*, pages 274 –279, dec. 2006.

[93] J. B. Mocholí, P. Sala, C. Fernández-Llatas, and J. C. Naranjo. Ontology for modeling interaction in ambient assisted living environments. In Panagiotis D. Bamidis, Nicolas Pallikarakis, and Ratko Magjarevic, editors, *XII Mediterranean Conference on Medical and Biological Engineering and Computing 2010*, volume 29 of *IFMBE Proceedings*, pages 655–658. Springer Berlin Heidelberg, 2010. 10.1007/978-3-642-13039-7_165.

[94] D. Morris, B. Schazmann, Yangzhe Wu, S. Coyle, S. Brady, J. Hayes, C. Slater, C. Fay, King Tong Lau, G. Wallace, and D. Diamond. Wearable sensors for monitoring sports performance and training. In *Medical Devices and Biosensors, 2008. ISSS-MDBS 2008. 5th International Summer School and Symposium on*, pages 121 –124, june 2008.

[95] Deirdre Morris, Benjamin Schazmann, Yangzhe Wu, Shirley Coyle, Sarah Brady, Cormac Fay, Jer Hayes, King Tong Lau, Gordon Wallace, and Dermot Diamond. Wearable technology for bio-chemical analysis

of body fluids during exercise. In *Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE*, pages 5741 –5744, aug. 2008.

[96] Martin Murth and Eva Kühn. A heuristics framework for semantic subscription processing. In Lora Aroyo, Paolo Traverso, Fabio Ciravegna, Philipp Cimiano, Tom Heath, Eero Hyvönen, Riichiro Mizoguchi, Eyal Oren, Marta Sabou, and Elena Simperl, editors, *The Semantic Web: Research and Applications*, volume 5554 of *Lecture Notes in Computer Science*, pages 96–110. Springer Berlin / Heidelberg, 2009.

[97] U. Naeem and J. Bigham. A comparison of two hidden markov approaches to task identification in the home environment. In *Pervasive Computing and Applications. ICPCA 2007. 2nd International Conference on*, pages 383 –388, july 2007.

[98] Usman Naeem, John Bigham, and Jinfu Wang. Recognising activities of daily life using hierarchical plans. In *Proceedings of the 2nd European conference on Smart sensing and context*, EuroSSC'07, pages 175–189, Berlin, Heidelberg, 2007. Springer-Verlag.

[99] Ian Oliver and Jukka Honkola. Personal semantic web through a space based computing environment. In *Proceedings of the 2nd Interntional Conference on Semantic Computing*, 2008.

[100] R. Oppermann and M. Specht. A Context-Sensitive Nomadic Exhibition Guide. In *Second Symposiumon Handheld and Ubiquitous Computing, Springer, pp. 127-142*, 2000.

[101] A. Pentland. Perceptual environments. In D. Cook and S. Das, editors, *Smart Environments: Technologies, Protocols and Applications*, pages 175–194. Wiley, 2005.

[102] M. Philipose, K. P. Fishkin, M. Perkowitz, D. J. Patterson, D. Fox, H. Kautz, and D. Hahnel. Inferring activities from interactions with objects. *IEEE Pervasive Computing*, 3(4):50–57, 2004.

[103] An Ping, Zhiliang Wang, Xuefei Shi, Changhao Deng, Lili Bian, and Liyang Chen. Designing an emotional majormodo in smart home healthcare. *Intelligent Interaction and Affective Computing, International Asia Symposium on*, 0:45–47, 2009.

[104] Ronald Poppe. A survey on vision-based human action recognition. *Image and Vision Computing*, 28(6):976 – 990, 2010.

[105] Davy Preuveneers, Jan Van Den Bergh, Dennis Wagelaar, Andy Georges, Peter Rigole, Tim Clerckx, E Berbers, Karin Coninx, and

Koen De Bosschere. Towards an extensible context ontology for ambient intelligence. In *In: Proceedings of the Second European Symposium on Ambient Intelligence*, pages 148–159. Springer-Verlag, 2004.

[106] W. Prinz. NESSIE: An awareness environment for cooperative settings. In *Proceedings of 6th Conference on Computer-Supported Cooperative Work*, 1999.

[107] Dapeng Qiao, G.K.H. Pang, Mui Man Kit, and D.C.C. Lam. A new pcb-based low-cost accelerometer for human motion sensing. In *Automation and Logistics, 2008. ICAL 2008. IEEE International Conference on*, pages 56 –60, sept. 2008.

[108] A. Rakotonirainy and R. Tay. In-vehicle ambient intelligent transport systems (ivaits): Towards an integrated research. In *Procedings of 7th International IEEE Conference on Intelligent Transportation Systems, ITSC 2004*, pages 648–651, 2004.

[109] Carlos Ramos, Juan Carlos Augusto, and Daniel Shapiro. Ambient intelligence - the next step for artificial intelligence. *Intelligent Systems, IEEE*, 23(2):15 –18, march-april 2008.

[110] M.J. Rantz, M. Aud, G. Alexander, D. Oliver, D. Minner, M. Skubic, J. Keller, Z. He, M. Popescu, G. Demiris, , and S. Miller. Tiger place: An innovative educational and research environment. In *AAAI in Eldercare: New Solutions to Old Problems*, 2008.

[111] P. Rashidi and D. Cook. Home to home transfer learning. In *Proceedings of the AAAI Plan, Activity, and Intent Recognition Workshop*, 2010.

[112] P. Rashidi and D. Cook. Multi home transfer learning for resident activity discovery and recognition. In *Proceedings of the International workshop on Knowledge Discovery from Sensor Data*, 2010.

[113] P. Rashidi and D.J. Cook. Keeping the intelligent environment resident in the loop. In *Intelligent Environments, 2008 IET 4th International Conference on*, pages 1 –9, july 2008.

[114] P. Rashidi and D.J. Cook. Keeping the resident in the loop: Adapting the smart home to the user. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 39(5):949 –959, 2009.

[115] Parisa Rashidi, Diane J. Cook, Lawrence B. Holder, and Maureen Schmitter-edgecombe. Discovering activities to recognize and track in a smart environment. *IEEE Trans. on Knowledge and Data Engineering*, to appear:14, 2010.

[116] Stefan Reifinger, Frank Wallhoff, Markus Ablassmeier, Tony Poitschke, and Gerhard Rigoll. Static and dynamic hand-gesture recognition for augmented reality applications. *Work*, 4552:728–737, 2007.

[117] P. Remagnino and G.L. Foresti. Ambient intelligence: A new multidisciplinary paradigm. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 35(1):1 – 6, jan. 2005.

[118] P. Remagnino, H. Hagras, N. Monekosso, and S. Velastin. Ambient intelligence. In Paolo Remagnino, Gian Foresti, and Tim Ellis, editors, *Ambient Intelligence*, pages 1–14. Springer New York, 2005.

[119] Mitchel Resnick, John Maloney, Andres Monroy-Hernandez, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, and Yasmin. B. Kafai. Scratch: programming for all. *Communications of the ACM*, 52(11):60–67, 2009.

[120] Anders Rex. Design of a caregiver programmable assistive intelligent environment. Aalto University., 2011.

[121] Daniele Riboni and Claudio Bettini. COSAR: hybrid reasoning for context-aware activity recognition. *Personal and Ubiquitous Computing*, 15(3):271–289, 2011.

[122] Daniele Riboni, Linda Pareschi, Laura Radaelli, and Claudio Bettini. Is ontology-based activity recognition really effective? In *PerCom Workshops*, pages 427–431, 2011.

[123] Marko A. Rodriguez and Johan Bollen. Modeling computations in a semantic network. *CoRR*, abs/0706.0022, 2007.

[124] Natalia Díaz Rodríguez, Pasi Kankaanpää, M. Mohsin Saleemi, Johan Lilius, and Iván Porres. Programming biomedical smart space applications with BioImageXD and PythonRules. In *Proceedings of the 4th International Workshop on Semantic Web Applications and Tools for the Life Sciences*, SWAT4LS '11, pages 10–11, New York, NY, USA, 2012. ACM.

[125] M. Roman, C. Hess, R. Cerqueira, and A. Ranganathan. A middleware infrastructure for active spaces. In *IEEE Pervasive Computing*, 2002.

[126] M. Ros, M.P. Cuéllar, M. Delgado, and A. Vila. Online recognition of human activities and adaptation to habit changes by means of learning automata and fuzzy temporal windows. *Information Sciences*, in press:–, 2011.

[127] M. Ros, M.P. Cuéllar, M. Delgado, A. Vila, and H. Hagras. Detection of abnormal human activities by means of learning automata. *International Journal of Information Technology and Decision Making*, submitted:–, 2012.

[128] M. Ros, M. Pegalajar, M. Delgado, A. Vila, D.T. Anderson, J.M. Keller, and M. Popescu. Linguistic summarization of long-term trends for understanding change in human behavior. In *Fuzzy Systems (FUZZ), 2011 IEEE International Conference on*, pages 2080 –2087, june 2011.

[129] Maria Ros, Miguel Delgado, and Amparo Vila. Fuzzy method to disclose behaviour patterns in a tagged world. *Expert Systems with Applications*, 38(4):3600 – 3612, 2011.

[130] Matthew Rowe, Miriam Fernandez, Sofia Angeletou, and Harith Alani. Community analysis through semantic rules and role composition derivation. *Web Semantics: Science, Services and Agents on the World Wide Web*, 0(0), 2012.

[131] Mohsin Saleemi, Natalia Diaz, Johan Lilius, and Ivan Porres. A Framework for Context-aware Applications for Smart Spaces. In S. et a S. Balandin, editor, *ruSMART 2011 : The 4th Conference on Smart Spaces, 2011.*, page 12, St. Petersburg, 2011. LNCS.

[132] W. Shen, S. Lang, and L. Wang. iShopFloor: An internet-enabled agent-based intelligent shop floor. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 35(3):371–381, 2005.

[133] Y. Shi, W. Xie, G. Xu, R. Shi, E. Chen, Y. Mao, and F. Liu. The smart classroom: Merging technologies for seamless tele-education. *IEEE Pervasive Computing*, 2:47–55, 2003.

[134] G. Singla, D.J. Cook, and M. Schmitter-Edgecombe. Recognizing independent and joint activities among multiple residents in smart environments. *J Ambient Intelligence Humaniz. Comput.*, 1:57–63, 2010.

[135] V. Stanford. Infrastructure for distributed and embedded systems and interfaces. In *Proceedings of the Embedded Systems, Ambient Intelligence and Smart Surroundings Conference*, 2005.

[136] M. Stoettinger. Context-awareness in industrial environments, www.mobile-safety.com, 2004.

[137] Lin Sun, Daqing Zhang, Bin Li, Bin Guo, and Shijian Li. Activity recognition on an accelerometer embedded mobile phone with varying positions and orientations. In Zhiwen Yu, Ramiro Liscano, Guanling Chen, Daqing Zhang, and Xingshe Zhou, editors, *Ubiquitous Intelligence and Computing*, volume 6406 of *Lecture Notes in Computer Science*, pages 548–562. Springer Berlin / Heidelberg, 2010.

[138] E. Tapia, S. Intille, and K. Larson. Activity recognition in the home using simple and ubiquitous sensors. *Pervasive Computing*, 3001:158–175, 2004.

[139] Douglas L. Vail, Manuela M. Veloso, and John D. Lafferty. Conditional random fields for activity recognition. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, AAMAS '07, pages 235:1–235:8, New York, NY, USA, 2007. ACM.

[140] J. Vales-Alonso, P. López-Matencio, J. Alcaraz, J. Sieiro-Lomba, E. Costa-Montenegro, and F. González-Castaño. A dynamic programming approach for ambient intelligence platforms in running sports based on markov decision processes. In Zdzislaw Hippe, Juliusz Kulikowski, and Teresa Mroczek, editors, *Human – Computer Systems Interaction: Backgrounds and Applications 2*, volume 98 of *Advances in Intelligent and Soft Computing*, pages 165–181. Springer Berlin / Heidelberg, 2012.

[141] T. van Kasteren and B. Krose. Bayesian activity recognition in residence for elders. *IET Conference Publications*, 2007(CP531):209–212, 2007.

[142] S.A. Velastin, B.A. Boghossian, J. Sun B.P.L. Lo, and M.A.Vicencio-Silva. Prismatica: Toward ambient intelligence in public transport environments. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, 35(1):164–182, 2005.

[143] Maria Poveda Villalon, Mari Carmen Suárez-Figueroa, R. García-Castro, and A. Gómez-Pérez. A context ontology for mobile environments. In *Proceedings of Workshop on Context, Information and Ontologies - CIAO 2010 Co-located with EKAW 2010*, volume 626, Germany, October 2010. CEUR-WS.

[144] Tobias Walter and Jürgen Ebert. Combining DSLs and ontologies using metamodel integration. In *Proceedings of the IFIP TC 2 Working Conference on Domain-Specific Languages*, DSL '09, pages 148–169, Berlin, Heidelberg, 2009. Springer-Verlag.

[145] M. Weiser. The computer for the twenty-first century. *Scientific American*, 165:94–104, 1991.

[146] D.H. Wilson, D. Wyaat, and M. Philipose. Using context history for data collection in the home. In *Pervasive*, volume 3468, 2005.

[147] T. Gu X.H.Wang, D.Q. Zhang and H.K. Pung. Ontology based context modeling and reasoning using OWL. In *Workshop Proceedings of the 2nd IEEE Conference on Pervasive Computing and Communications*, 2004.

[148] Lu Xia, Chia-chih Chen, and J K Aggarwal. Human detection using depth information by kinect. *Pattern Recognition*, pages 15–22, 2011.

[149] Hiroyuki Yamahara, Hideyuki Takada, and Hiromitsu Shimakawa. Behavior detection based on touched objects with dynamic threshold determination model. In Gerd Kortuem, Joe Finney, Rodger Lea, and Vasughi Sundramoorthy, editors, *Smart Sensing and Context*, volume 4793 of *Lecture Notes in Computer Science*, pages 142–158. Springer Berlin / Heidelberg, 2007.

[150] S.S. Yau, S. Gupta, F. Karim, S. Ahamed, Y. Wang, and B. Wang. A smart classroom for enhancing collaborative learning using pervasive computing technology. In *Proceedings of the WFEO World Congress on Engineering Education*, 2003.

[151] Jie Yin, Qiang Yang, Dou Shen, and Ze-Nian Li. Activity recognition via user-trace segmentation. *ACM Trans. Sen. Netw.*, 4(4):19:1–19:34, September 2008.

[152] YoungTaek Jin Yun Her, Su-Kyoung Kim. A context-aware framework using ontology for smart phone platform. In *International Journal of Digital Content Technology and its Applications, Volume 4, Number 5*, 2010.

[153] Daqing Zhang, Manli Zhu, Hengseng Cheng, Yenkai Koh, and Mounir Mokhtari. Handling heterogeneous device interaction in smart spaces. In *Proceedings of the Third international conference on Ubiquitous Intelligence and Computing*, UIC'06, pages 250–259, Berlin, Heidelberg, 2006. Springer-Verlag.

[154] Huiyu Zhou and Huosheng Hu. Human motion tracking for rehabilitation, a survey. *Biomedical Signal Processing and Control*, 3(1):1 – 18, 2008.

[155] Fengqing Zhu, M. Bosch, Insoo Woo, SungYe Kim, C.J. Boushey, D.S. Ebert, and E.J. Delp. The use of mobile devices in aiding dietary assessment and evaluation. *Selected Topics in Signal Processing, IEEE Journal of*, 4(4):756 –766, aug. 2010.