UNIVERSIDAD DE DEUSTO

TOWARDS MORE RELIABLE AND EFFICIENT INTELLIGENT ENVIRONMENTS: UNCERTAINTY, VAGUENESS AND REASONING DISTRIBUTION

Tesis doctoral presentada por Aitor Almeida

Dirigida por Dr. Diego López-de-Ipiña

Bilbao, Abril de 2013

UNIVERSIDAD DE DEUSTO

TOWARDS MORE RELIABLE AND EFFICIENT INTELLIGENT ENVIRONMENTS: UNCERTAINTY, VAGUENESS AND REASONING DISTRIBUTION

Tesis doctoral presentada por Aitor Almeida

dentro del Programa de Doctorado en Sistemas de Información

Dirigida por Dr. Diego López-de-Ipiña

El doctorando El director

Bilbao, Abril de 2013

Towards More Reliable and Efficient Intelligent Environments: Uncertainty, Vagueness and Reasoning Distribution

Author: Aitor Almeida

Advisor: Diego López-de-Ipiña

The following web-page address contains up to date information about this dissertation and related topics:

`http://paginaspersonales.deusto.es/aitor.almeida/`

Text printed in Bilbao

First edition, April 2013

Para Ama y Amama, gracias.

Abstract

Intelligent Environments and context-aware systems aim to improve users' everyday life by taking care of daily activities while being completely invisible. In order to achieve this, three objectives must be accomplished: a) the reactions to the context changes have to be accurate, b) those reactions have to be timely and c) computers have to disappear into the environment. These objectives can be difficult to achieve at times. In order to achieve sensible and accurate reactions to changes in the environment, the model of the context must be as close to reality as possible. This model must be as expressive as possible to achieve a more precise inference over the context and thus to better cater to the needs of the users. On the other hand, achieving both timely reactions and environment integrated computational capabilities can be conflicting at times. Integrating the computational capabilities into the environment often results in using more limited devices, which in turn results in higher inference times. This dissertation aims to tackle these problems. In order to create a better context model, a new context reasoning mechanism that combines semantic inference with uncertainty and vagueness will be presented. This mechanism will allow to better model the environment, achieving more expressivity than traditional context models. The proposed mechanism also includes a data fusion algorithm that integrates the measures taken from different devices to create a unified vision of the environment. To provide timely reactions and help integrating the computational capabilities in the environment, a distributed inference architecture will be described. This architecture will allow splitting the global inference task into smaller inference sub-units, resulting in a faster, lighter, more efficient and more robust inference.

Resumen

El objetivo de los entornos inteligentes y los sistemas sensibles al contexto es mejorar la vida diaria de los usuarios haciéndose cargo de las actividades que se puedan automatizar de manera transparente, siendo completamente invisibles para el usuario. Para conseguir esto se deben alcanzar tres objetivos: a) las reacciones a los cambios en el contexto deben de ser lo más exactas posibles, b) las reacciones deben llevarse a cabo en un tiempo prudencial y c) los dispositivos computacionales deben de desaparecer en el entorno, convirtiéndose en invisibles.

Estos objetivos pueden ser difíciles de alcanzar en ciertos casos. Para que las adaptaciones se lleven a cabo de manera exacta y eficiente, el modelo de contexto debe de ser lo más cercano posible a la realidad. Este modelo debe de proveer la expresividad necesaria para conseguir una inferencia más precisa y para adaptarse mejor a las necesidades de los usuarios. Por otro lado, conseguir reacciones rápidas y que los dispositivos estén integrados en el entorno son objetivos enfrentados. Integrar las capacidades computacionales en el entorno suele resultar en dispositivos más limitados, lo que a su vez acarrea mayores tiempos de inferencia.

Esta tesis hace frente a estos problemas. Para crear un mejor modelo de contexto, se propone un nuevo mecanismo de razonamiento que integre la inferencia semántica con incertidumbre y vaguedad. Esto permitirá conseguir una mayor expresividad a la hora de modelar el contexto e incluirá además un mecanismo de fusión de datos que integrará las diferentes medidas en una visión global del entorno. Además se creará una estructura distribuida de razonamiento que permitirá alcanzar conclusiones en un menor tiempo y de manera más eficiente.

Contents

List of Figures

List of Tables

AAL: Ambient Assisted Living
API: Application Programming Interface
AmI: Ambient Intelligence
ANTLR: ANother Tool for Language Recognition
BDI: Belief-desire-intention
CLIPS: C Language Integrated Production System
CNP: Contract Net Protocol
FCL: Fuzzy Control Language
FOAF: Friend of a friend
GPS: Global Positioning System
IP: Internet Protocol
Java ME: Java Platform, Micro Edition
LAN: Local Area Network
MEBNs: Multi Entity Bayesian Networks
OSGi: Open Service Gateway Initiative
OWL: Web Ontology Language
PC: Personal Computer
PDA: Personal Digital Assistant
RDF: Resource Description Framework
RDQL: RDF Data Query Language
SGML: Standard Generalized Markup Language
SPARQL: SPARQL Protocol and RDF Query Language
SLA: Service Level Agreement
SWRL: Semantic Web Rule Language
WSN: Wireless Sensor Network

"Begin at the beginning" the King said gravely, "and go till you come to the end: then stop."

Lewis Carroll in Alice's Adventures in Wonderland

CHAPTER

1

Introduction

Intelligent Environments, as defined by Weiser in his seminal article(Weiser, 1991) *"The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it."* and by Steventon and Wright(Steventon and Wright, 2006) as *"Intelligent environments are spaces in which computation is seamlessly used to enhance ordinary activity. One of the driving forces behind the emerging interest in highly interactive environments is to make computers not only genuine user-friendly but also essentially invisible to the user"*, are nearer to their existence than ever. Nowadays we are surrounded by smart phones, smart TVs, smart homes, smart appliances, smart cars and smart cities that help us on our daily activities. Low-cost embedded platforms like Arduino[1], Teensy[2], Raspberry Pi[3] and STM32 Discovery[4] are widely available and are used by a growing community of hardware enthusiasts to create a plethora of computationally augmented devices.

With all this computational intelligence present in the environment, the information available to applications to adapt their behavior is remarkable. This context

[1] http://www.arduino.cc/

[2] http://www.pjrc.com/teensy/

[3] http://www.raspberrypi.org/

[4] http://mouser.com/stm32discovery/

(as defined by Dey(Dey, 2001), *"Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves"*) must be captured, modeled, merged, interpreted, understood and served to be useful for other applications. Context managers ease this process for the client applications. They provide a unified vision of the context, providing a common context model for all the applications. Context managers mediate with sensors and actuators, providing an abstraction layer over the specific device implementation and APIs. They also provide context aggregation, merging the context information provided by heterogeneous sources and reasoning over context data, creating higher level context, inferring interpretations for the context and understanding the user situation for decision support.

Using this context information, context-aware applications can adapt themselves to better cater to user needs. Applications can react to environmental measure changes (e.g. turning on the lights when the user enters a room or turning up the heater when the temperature drops), to the user's location (e.g. providing contextualized recommendations) or to his current activity (e.g. muting the TV if the user receives a phone call). In order to achieve these personalized behaviors, two elements are essential:

- Applications must be provided with a sound context model. This model must be expressive enough to represent all the information available in the environment.

- Inference over context information must be able to process all the expressiveness of the context model. This inference must be done in a timely manner because a) context changes dynamically, taking too much time to achieve a conclusion can result in working with outdated context and b) users' expect the applications to react in a reasonable amount of time.

Two remarkable and distinguished features of this dissertation are a) the combination of uncertainty and vagueness with semantic modeling to provide a more expressive inference over context as described in Chapter 3 and b) the distribution of the global inference task among a ecosystem of reasoning engines as described in Chapter 4.

1.1 Motivation

Situations in real environments do not always fall in clearly defined categories and facts are not always a hundred percent right. Reality, and so the context model used to represent it, can be ambiguous at times. Faulty sensors provide wrong measures, detection systems are not perfect and the captured information can be uncertain or the user's perception can be vague. The context model should be able to represent this uncertainty and vagueness, as it is an intrinsic part of reality.

Lukasiewicz and Straccia(Lukasiewicz and Straccia, 2008) give a definition for uncertainty: *"We recall that under uncertainty theory fall all those approaches in which statements rather than being either true or false, are true or false to some probability or possibility (for example, "it will rain tomorrow"). That is, a statement is true or false in any world, but we are "uncertain" about which world to consider as the right one, and thus we speak about e.g. a probability distribution or a possibility distribution over the worlds."*. And for vagueness: *"...vagueness/fuzziness theory fall all those approaches in which statements (for example, "the tomato is ripe") are true to some degree, which is taken from a truth space. That is, an interpretation maps a statement to a truth degree, since we are unable to establish whether a statement is completely true or false due to the involvement of vague concepts, such as "ripe", which only have an imprecise definition."*. Using these concepts in the context model, we aim to provide a more realistic and expressive representation of the environment. As it will be addressed in Chapter 3 and Chapter 5, this will lead to more informed reactions to context changes and to a more robust data model.

On the other hand, reasoning over context can be cumbersome, especially when dealing with a semantic context model. When referring to context reasoning, we are using Chen's(Chen, 2004) definition: *"Context reasoning can play different roles in sensing and knowledge sharing. In sensing, context reasoning is a process that reasons over the sensing data to make interpretations about the context. In knowledge sharing, context reasoning is a process that reasons over different contextual knowledge to detect and resolve inconsistent information"*. In Intelligent Environments, the maximum reasoning time is given by the amount of time users are ready to wait until a change happen. For example, lights must be turned on

when the user enters a room, not 15 seconds later. To ensure that changes happen in a timely fashion, the reasoning architecture must be able to deal with the vast amount of context data that modern Intelligent Environments provide. With the spread of computationally augmented devices and appliances and the growth of the number of available sensors, the context model complexity increases dangerously. To tackle this problem and to improve the inference process we propose to split the inference task of a given domain among a ecosystem of reasoning engines. As will be described in Chapter 4 and Chapter 5, this will improve both the timeliness and the reliability of the inference process.

1.2 Hypothesis, Goals and Limitations

Based on the current state of Intelligent Environments, the hypothesis of this dissertation is:

> It is possible to deploy effective Intelligent Environments into real domains provided that two features are given:
>
> 1. The uncertainty and vagueness that naturally occurs in real scenarios are taken into account, improving the expressivity of the context model.
> 2. The inference task of the domain is distributed among an ecosystem of reasoning engines to improve both the timeliness and the reliability of the inference.

To be able to validate this hypothesis the general goal of this dissertation is:

> To design and implement a reasoning mechanism that takes into account both uncertainty and vagueness and to design and implement a reasoning architecture able to split the inference task into a set of reasoning engines.

This general objective can be achieved by undertaking the following more specific steps:

1. To study the current state of the art on uncertainty and vagueness modeling and reasoning and on distributed reasoning.

2. To design an ontology that models the uncertainty and vagueness of the context information.

3. To design and implement a reasoning mechanism that combines both uncertainty and vagueness with the semantic inference.

4. To design and implement a reasoning architecture that allows splitting and distributing the global inference task into an ecosystem of reasoning engines.

5. To validate the obtained results both qualitatively and quantitatively.

The resulting reasoning architecture should also satisfy the following requirements:

1. The reasoning mechanism should be able to reason over the spatial and semantic information of the context information; to process the uncertainty and vagueness of the data to provide a vision of the context information closer to the ambiguous nature of the reality and to apply a data fusion process to provide a holistic picture of the context information of each location

2. The reasoning architecture should be independent of the semantic reasoning engine used on each inference sub-unit as long as the reasoning engine is able to process OWL ontologies.

3. The reasoning architecture should have the following characteristics:

    - Temporal decoupling of the different inference units.
    - Spatial decoupling of the inference process.
    - Minimize the number of triples and rules that each reasoning engine has to manage.
    - Sensibly distribute the information according to the different interests of the reasoning engines.
    - Allow the dynamic modification of the created organization.

Several features of the reasoning architecture will be considered beyond the scope of this research:

- The system will not provide any automatic method to automatically ascertain the certainty factor of the used sensors.
- This research work will not deal with semantic reasoning in mobile devices.

1.3 Research Methodology

To achieve the stated goals a research strategy was designed based on the following activities:

1. To carry out an exhaustive review of the state of the art in the areas of context modeling, context reasoning, semantic reasoning and ontologies. This analysis has been reinforced by attending specialized scientific congresses.
2. Perform a critical evaluation of the existing solutions, analyzing their scope and limitations and identifying the areas where it was possible to make a contribution to the state of the art.
3. Design and develop the different modules of the ambiguity-aware distributed context management infrastructure, extending their scope and capabilities gradually.
4. Perform experiments and evaluate the performance of the developed modules.
5. Attend congresses to present the achieved contributions and receive feedback from the scientific community.
6. Network with experts at conferences, meetings and email.
7. Update the contributions and redesign the system with the feedback attained from the previous actions.
8. Develop and deploy the final ambiguity-aware distributed context management infrastructure, which takes into account the uncertainty and vagueness of the context to provide a more realistic vision of the environment.
9. Disseminate the results obtained to the scientific community.

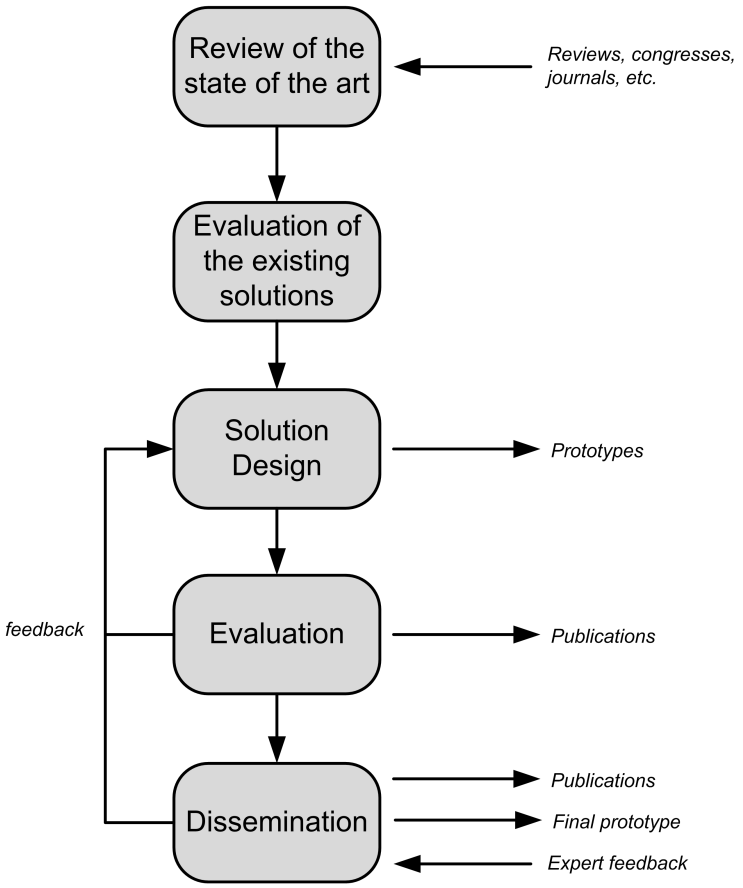

Figure 1.1: Followed research methodology

1.4 Thesis Outline

The thesis is structured in six chapters.

Chapter 1 outlines the motivation, hypothesis, goals and limitations of the performed research, as well as the followed methodology.

Chapter 2 analyses the state of the art in three different areas: context management, modeling of the uncertainty and vagueness in context information and distributed reasoning. This chapter compares the existing solutions and analyses their scope and limitations.

Chapter 3 describes the developed ontology, reasoning engine and data-fusion algorithm to model and process the uncertainty and vagueness of the context information.

Chapter 4 presents the multi-agent architecture developed to distribute the global inference task among a ecosystem of reasoning engines to improve the inference times. We discuss how distributing the reasoning process does not only improve the inference times but also results in a more robust system.

Chapter 5 provides a description of the evaluation performed to validate the system and the obtained results. This validation is comprised of two parts: a qualitative evaluation of the inference over the uncertainty and vagueness of the system and a quantitative evaluation of the multi-agent reasoning ecosystem.

Finally, Chapter 6 draws the conclusions of this research work. We discuss the achieved goals and analyze the advantages and drawbacks of the developed system. Some future lines of research are proposed and the chapter ends with some final remarks.

Even if you are on the right track, you'll get run over if you just sit there.

Will Rogers

CHAPTER

2

State of the Art

In this chapter the state of the art in the different subjects related to this thesis will be studied. This related work review will be centered in three main topics: 1) context management, 2) semantic modeling of the context's uncertainty and vagueness and 3) distributed reasoning. A chronological review of the evolution of the context managers will be presented. This review will highlight the two areas where the contributions of the thesis will be done:

- The first of those areas is the creation of a semantic context model that takes into account the uncertainty and the vagueness of the context information.

- The second of those areas is the development of a distributed reasoning mechanism that allows splitting the inference task among a group of inference engines.

The work done in those two areas will be analyzed, comparing it to the approach proposed in this dissertation.

2.1 A Chronological Review of the Evolution of Context Management

In this section a chronological review of the most relevant work in context management (see Figure 2.1) will be provided. At the end of this section a comparison of the different systems will be presented, analyzing their limitations

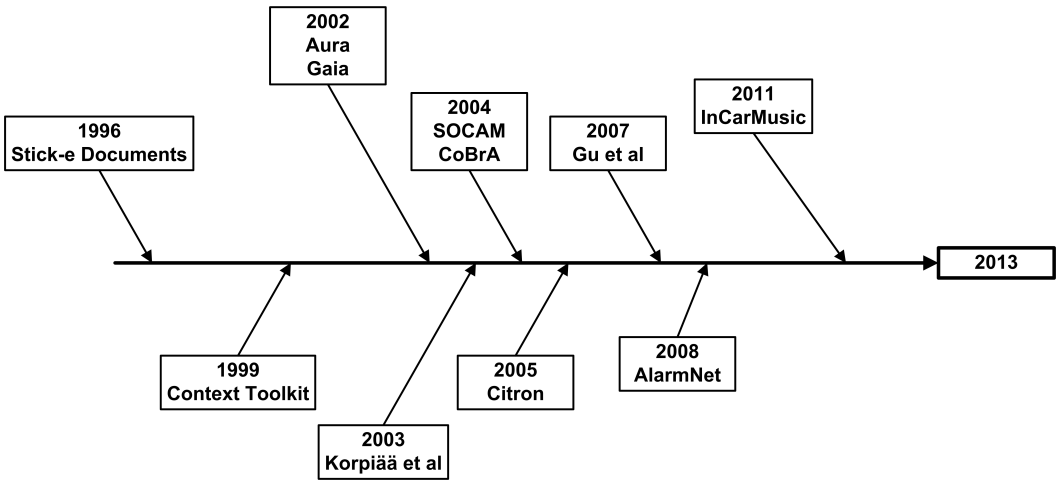

Figure 2.1: Timeline of the analyzed systems

2.1.1 Stick-e Documents (1996)

One of the first frameworks that managed the context information to improve the applications behavior was the Stick-e Documents(Brown, 1996). This framework is used to create context aware applications in the form of post-it notes that display a message when certain conditions are met. Each Stick-e Document is defined by its message and the related context. The framework takes into account the following context to trigger messages:

- *Location* of the note.
- *Adjacency* to other objects or people. The system uses the Active Badge(Want et al., 1992) system to recover the location of users and objects.

```
<note some attributes>
    <required>
    <with> J.D. Bovey <or> X. Chen
    <at> coordinates of location
    <content>
    This is the content.
</note>
```

Figure 2.2: Stick-e Document extracted from (Brown, 1996)

- *Critical states* provided by sensors. E.g. when the temperature drops below 22ºC.
- *Computer states*. E.g. when a certain folder is accessed.
- *Imaginary companions*, that is, persons that cannot be detected because they are not wearing an Active Badge. It is the user who has to provide this info to the system.
- Messages can also be displayed at a certain *time*.

All this context values can be combined to create complex trigger conditions for the Stick-e Documents. The content and its conditions are represented using SGML as can be seen in Figure 2.2.

2.1.2 Context Toolkit (1999)

The Context Toolkit(Salber et al., 1999) is a framework designed to ease the development of context-aware applications. Its main objective is to help recovering the context from a collection of heterogeneous sensors. To do this, the framework uses the concept of context widgets, that serve as an interface between the applications and the intelligent environment. Authors identify the following difficulties when using context data in applications:

- The context data is acquired from unconventional sensors. The authors cite for example recovering the location information of a mobile device using

outdoor GPS receivers (this was a common problem when the article was published).

- The recovered information must be abstracted and normalized to be used. For example the location information can be acquired from various system that do not use the same point of reference (E.g., GPS and an indoor location system).
- Context may be acquired from distributed and heterogeneous sensors. In some instances, to create high-level context information, the data recovered from various sensors must be processed and merged.
- Context information is dynamic and must be detected in real time.

Widget Class	IdentityPresence
Attributes	
Location	*Location the widget is monitoring*
Identity	*ID of the last user sensed*
Timestamp	*Time of the last arrival*
Callbacks	
PersonArrives (location, identity, timestamp)	*Triggered when a user arrives*
PersonLeaves (location, identity, timestamp)	*Triggered when a user leaves*

Figure 2.3: Example of definition of the Presence widget, extracted from (Salber et al., 1999)

In order to overcome these problems the authors propose the use of context widgets, which offer the following advantages:

- The context widgets hide the complexity of the underlying sensors, providing an abstract interface for each context type (E.g., users will not know if the presence is detected using an indoor location system, floor sensors or video processing).
- The context widgets abstract context information, adapting it to the necessities of the target application.

- Finally context widgets provide reusable and customizable building blocks. This makes ease to reuse widgets in different applications.

2.1.3 Aura (2002)

Aura(Garlan et al., 2002) is a context-aware framework designed bearing in mind ubiquitous computing environments that encompass wireless communications, handheld or wearable computers and smart spaces. Aura aims to reduce the distractions in pervasive computing environments, retaining human attention as much as possible. Aura uses *cyber foraging*(Satyanarayanan, 2001) to complete and augment the capabilities of the mobile devices, improving the user experience. Authors call this the *Personal Aura*, which always surrounds the user and knows the capabilities available in the environment to perform the required tasks.

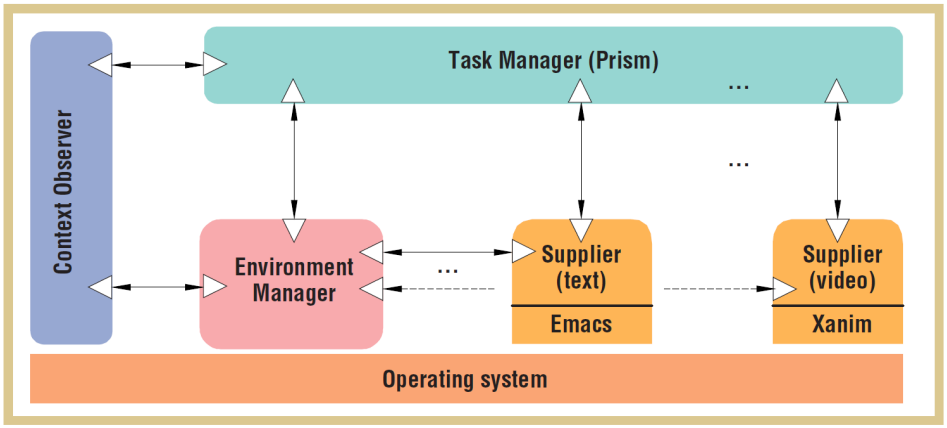

Figure 2.4: Architecture of the Aura framework, extracted from (Garlan et al., 2002)

The architecture of the framework is composed by:

- Prism, the task manager that represents the concept of Personal Aura.
- The Context Observer that gives information on the physical context and monitors relevant events, reporting them to Prism and the Environment Manager.
- The Environment Manager that serves as a gateway to the environment.

```
<auraTask id="demo">
    <service type="editText">
        <duration unit="minutes" bad="10" good="30"/>
        <settings pane_height="360" pane_width="200">
            <spelling enabled="yes"
            ignoreAllCaps="yes"/>
            <editing overstrike="no"
            replaceSelection="yes"/>
        </settings>
    </service>
    <material origin="myTextFile" format="txt">
        <state cursor="104" scroll="28" zoom="100"/>
    </material>
</auraTask>
```

Figure 2.5: Example of a description of a Aura service, extracted from (Sousa and Garlan, 2002)

- The Suppliers, that are the building blocks that services are composed of.

2.1.4 Gaia (2002)

Gaia(Roman et al., 2002) is a meta-operating system for active spaces. Authors define active spaces as physical spaces that contain real objects, an heterogeneous, interconnected collection of devices and users performing activities. Gaia provides a framework to develop user-centric, resource-aware, multi-device, context-sensitive and mobile applications in these spaces. The main elements of the Gaia architecture are:

- The Event Manager, which manages the events in the active space and uses a decoupled communication model based on suppliers, consumers and channels.
- The Context Service, which allows applications to query and register for specific context data.

- The Presence Service, which maintains information about software components, devices and people.
- The Space Repository, which stores information about what software and hardware elements exist in the Active Space.
- The Context File System, a context aware file system that using context data to simplify many of the activities usually performed manually.

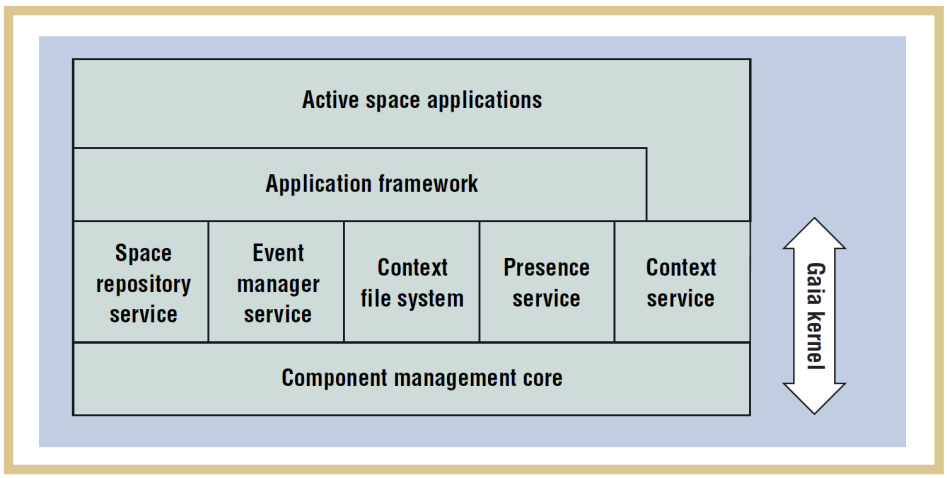

Figure 2.6: Gaia architecture, extracted from (Roman et al., 2002)

2.1.5 **Korpiää et al. (2003)**

The framework(Korpipaa et al., 2003) objectives are to recognize semantic context in real time in the presence of uncertain noisy and rapidly changing information and delivering that context data to the final applications using an event-based architecture. The authors use the blackboard paradigm(Engelmore and Morgan, 1988) as the communication mechanism between the framework elements. The basic elements of the framework architecture are:

- The Context Recognition Service receives low level context and processes it to create high level, more abstract context information. The inference process of the framework takes places in this element.

- The Resource Servers recover the information from the data sources. These servers also preprocess this data to eliminate inconsistencies or wrong values. The data is also semantically annotated.

- The Context Manager, which is the central element that stores the context information so the final applications can use it. The clients can access these data using a pull (queries) or push (events) approach.

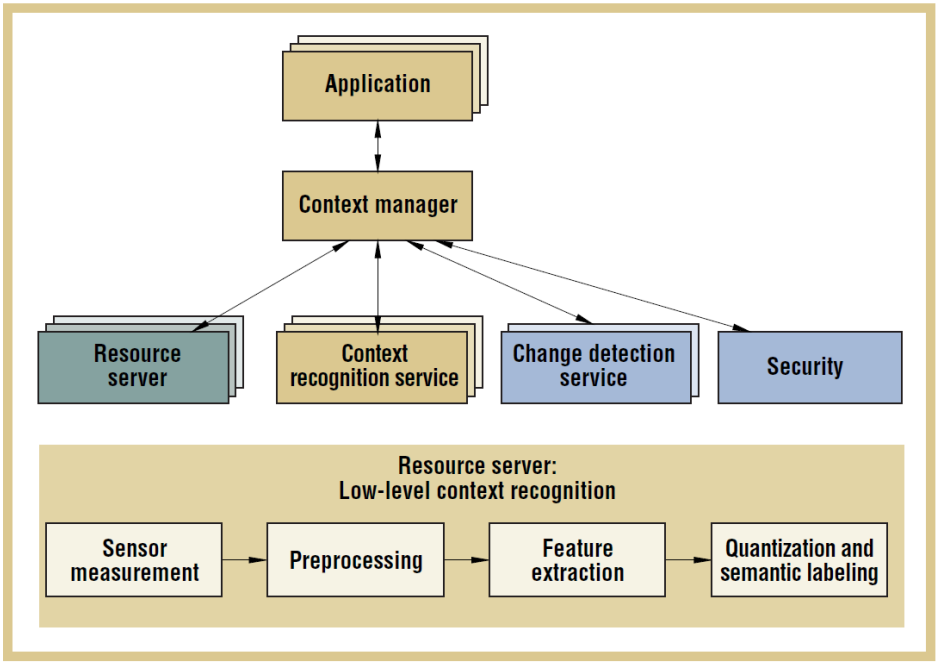

Figure 2.7: Architecture of the Context Management Framework, extracted from (Korpipaa et al., 2003)

Although this framework models both uncertainty (using a confidence level) and vagueness (using fuzzy sets) it does not combine them during the inference process. It only uses these metrics as additional conditions for the reasoning. As will be discussed in the next chapter, the inference mechanism proposed in this dissertation combines both metrics to annotate the inferred facts, providing a more expressive inference.

```
ContextRequest( Environment:Light )
->{Bright, Natural, NotAvailable}
ContextSetRequest( {Environment:Humidity,
    Device:Activity:Position}
->{Dry, AntennaUp}
ContextSetRecognitionRequest(Environment:Sound:Type )
->Car
```

Figure 2.8: Example of context requests, extracted from (Korpipaa et al., 2003)

2.1.6 SOCAM (2004)

SOCAM(Gu et al., 2004) is a context-aware infrastructure for developing and prototyping applications in an Intelligent Environment. SOCAM is developed on top of the Open Service Gateway Initiative (OSGi)[1], which is a service-oriented open standard. The OSGi framework includes management functionalities like installing, activating, deactivating, updating and removing services. SOCAM also follows an ontology based approach to model the context, using the OWL 1.1 version.

The SOCAM architecture is composed by these elements:

- The Context Providers that abstract the context information from heterogeneous sources. They also semantically annotate this information according to the defined ontology.
- The Context Interpreters that provide inference over the context information. The reasoning in this framework takes places in this element.
- The Context Database, where all the semantized context information is stored.
- The Context-Aware Applications that use the context information provided by the framework, adapting their behavior to the current context.
- The Service-Locating Service, which provides a search mechanism for the Context Providers and Context Interpreters.

[1] http://www.osgi.org

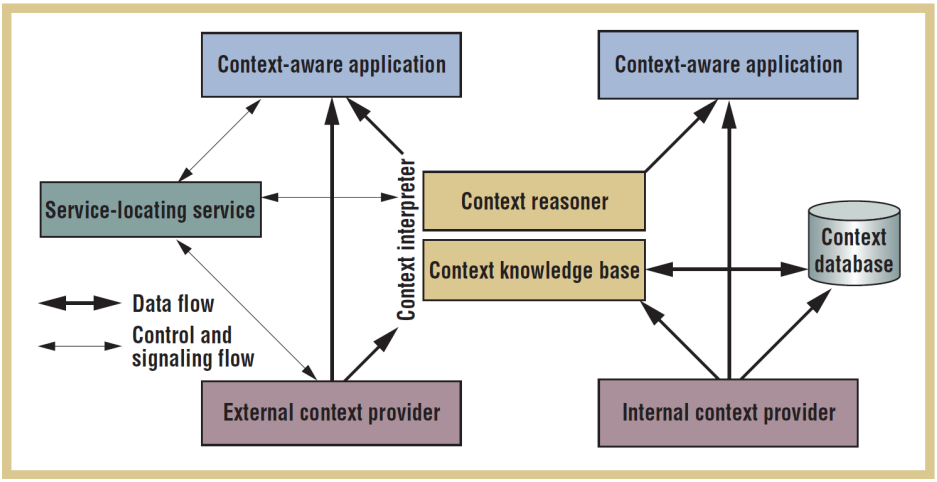

Figure 2.9: SOCAM architecture, extracted from (Gu et al., 2004)

2.1.7 CoBrA (2004)

The Context Broker Architecture (CoBrA)(Chen, 2004) is a framework for the creation of context aware applications in Intelligent Environments. CoBrA uses OWL ontologies to model the context information and defines rules over the context to improve the context data and interpret it. It also provides a policy language to control the access to private context.

The COBRA architecture is composed by four main components:

- The Context Knowledge Base stores the context information, including the ontology schema, the instances and metadata to describe the structure of the stored information.

- The Context-Reasoning Engine, which provides logical inference for reasoning over the context information. This inference includes creating new aggregated context information from the acquired data, interpreting the context information and detecting and resolving inconsistent information.

- The Context-Acquisition Module is a collection of libraries for capturing the context information from sensors, agents and the web. It abstracts the process of capturing the context providing high-level functionality.

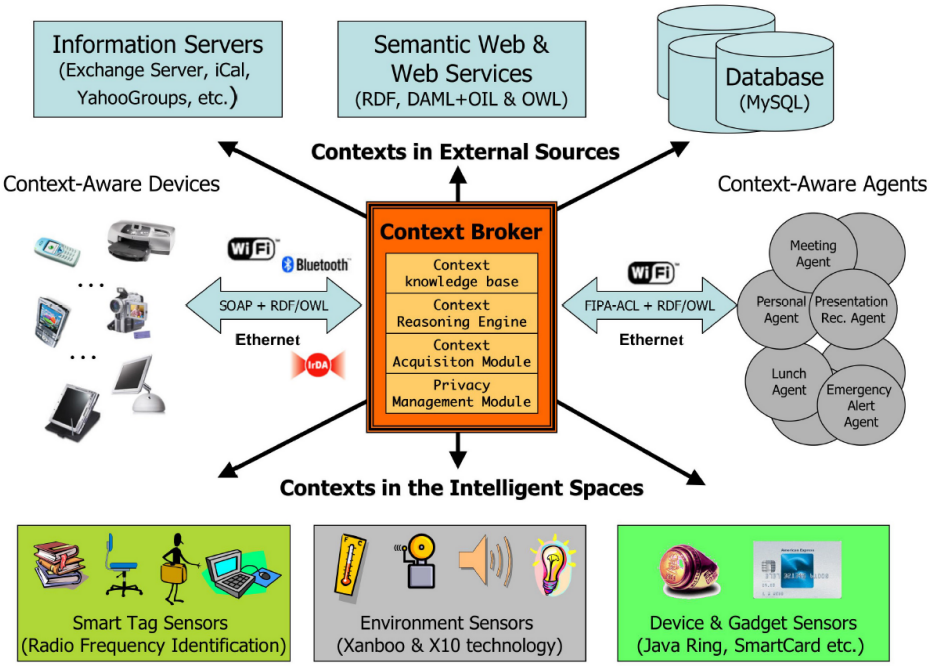

Figure 2.10: COBRA architecture, extracted from (Chen, 2004)

- The Privacy-Management Module, which manages the users privacy policies and controls the access to the context information.

2.1.8 Citron (2005)

Citron(Yamabe et al., 2005) is a framework for context acquisition and sharing aimed to personal devices. Citron acquires information related to the user and the environment and this context information is used to adapt the applications running on the personal device. For the communications Citron follows the blackboard paradigm, using Tuple Spaces(Carriero and Gelernter, 1989) to share the context information.

The Citron architecture consists of two software components:

- The Citron Worker is the module that provides the sensor data analysis. The Citron Workers receive the raw data from the sensors and process it creating higher level context information. E.g., a worker could receive data from a skin resistance sensor and infer if the user is holding an object or not.

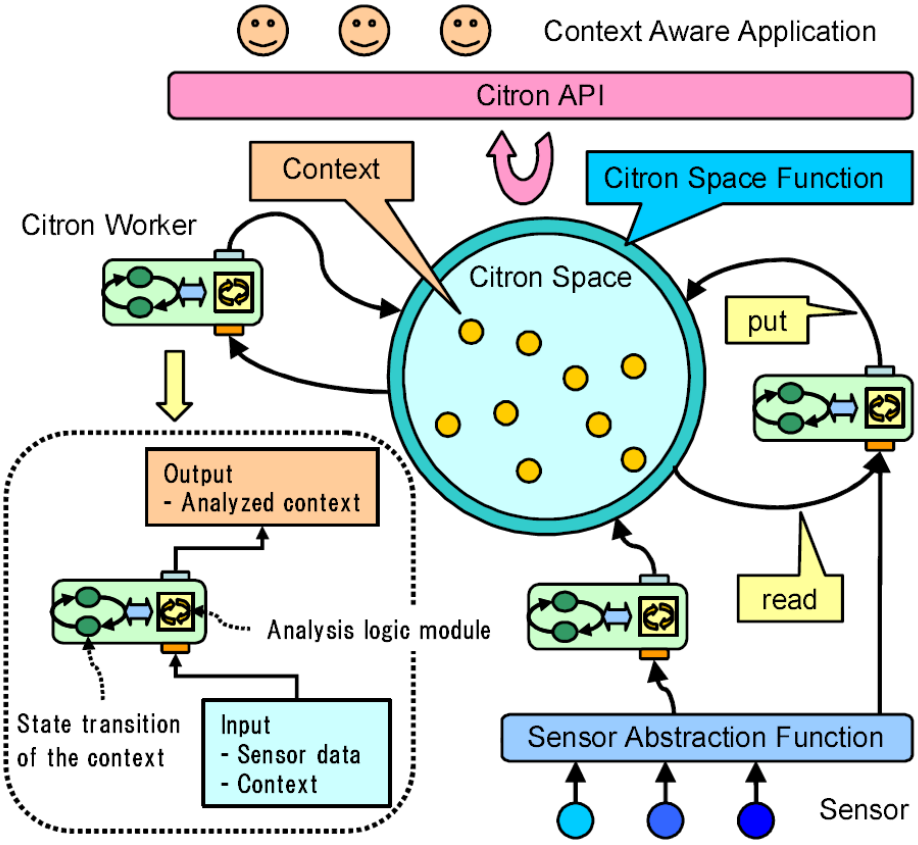

Figure 2.11: Citron architecture, extracted from (Yamabe et al., 2005)

- The Citron Space is the tuple space that stores the context information provided by the Citron Workers. The Citron Space handles the data queries performed by the Citron Workers and the applications running over Citron.

2.1.9 **Gu et al. (2007)**

This framework(Gu et al., 2007) focuses on providing ontology processing and reasoning for mobile context-aware application running on Java based handheld devices. The framework allows mobile devices to capture context information from sensors in the environment, to process that data and to infer facts from the context information. The framework architecture is composed by the following elements:

- The Sensor Wrappers, which capture data from the sensors and transform it to semantic information.
- The RDF/OWL Parser, which analyses and merges the context data, checks for inconsistencies and converts the textual syntax into internal tree-based representations.
- The Model Extraction, which converts the tree-based representation to a more easy to process format for the reasoning engine.
- sRDQL Query Engine, which supports a reduced version of the RDQL query language.
- The Reasoning Engine, which use a forward-chaining rule engine to extract new data from the context information.
- The Context Repository, a database with the context information.

2.1.10 **AlarmNet (2008)**

AlarmNet(Wood et al., 2008) is a AAL system for residential monitoring that uses a two-way data flow between the front and back-ends to prove context-aware protocols customized to the residents behaviours. AlarmNet uses environmental, physiological and activity sensors to capture the context information.

The system architecture can be divided in the following elements:

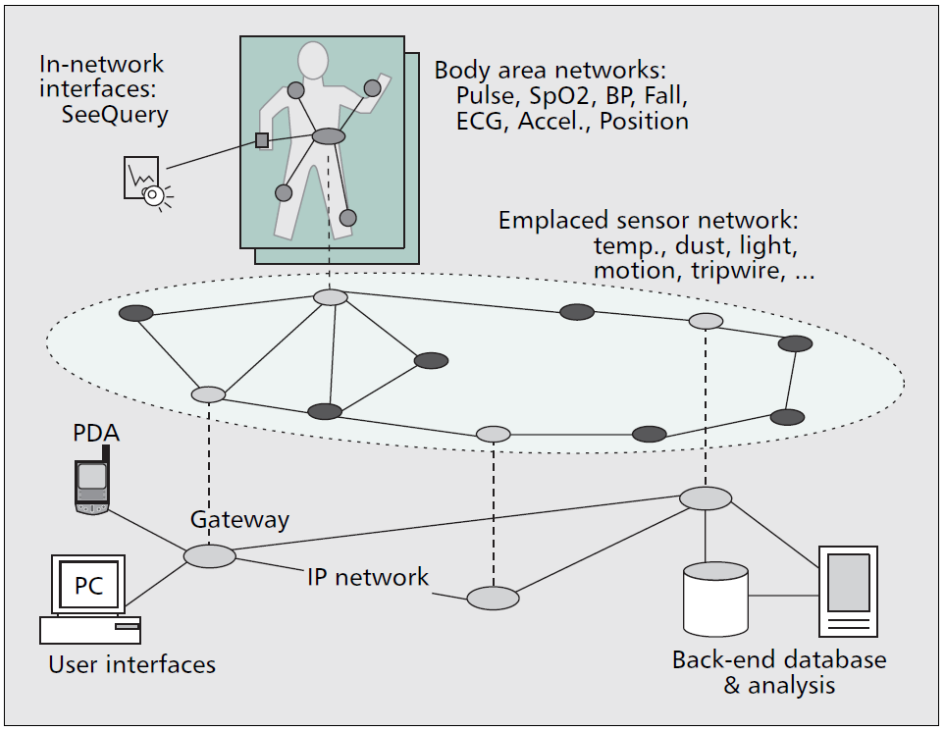

Figure 2.12: AlarmNet architecture, extracted from (Wood et al., 2008)

- The Mobile Body Networks, which are wireless sensor devices worn by the users. This sensors capture physiological and activity data. Each network is customized to the specific needs of the user. The networks contain a gateway that mediate with the WSN, providing integration with the system.
- The Emplaced Sensors that are deployed in the environment. These sensors capture context information like temperature, luminosity, motion or dust.
- The AlarmGate applications serve as an application level gateway between the WSN and the IP networks. These applications run on embedded platforms.
- The Back-End Programs that perform the inference over the context information, creating behavior profiles that enable context-aware power management and privacy, detection of deviations in personal behavior and decline in the users health.
- The User Interfaces allow caretakers, family and other users to query the sensor data.

2.1.11 InCarMusic (2011)

InCarMusic(Baltrunas et al., 2011) is a context-aware recommender system that adapts to specific situations where the recommended resources will be used. In this case the system recommends songs while the users are using the car. The system takes into account context information like the current traffic conditions, the driver's mood, the weather, the road type or the sleepiness of the driver. Using these data the system creates recommendations for music types related to certain context configurations.

The recommender has been developed to run on Android systems and it is designed to reside in the user's smartphone.

2.1.12 Comparison

In Table 2.1 the results of the state of the art review for the evolution in context management are summarized. We focus our analysis in two characteristics: the

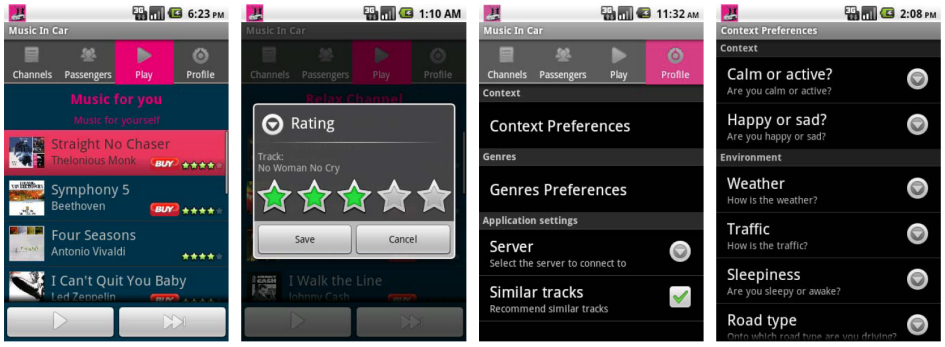

Figure 2.13: InCarMusic user interface, extracted from (Baltrunas et al., 2011)

inclusion of uncertainty and vagueness in the context model and the nature (centralized or distributed) of the inference process that takes place in the Context Manager. Other authors have performed an extensive analysis on the desired characteristics for context management. The most relevant of those works are those done by Dey and Abowd(Dey et al., 2001), Guan et al.(Guan et al., 2007), Baldauf and Dustdar(Baldauf et al., 2007) and Hong et al.(Hong et al., 2009). The last two surveys provide an extensive review of the literature in the area of context-aware applications.

As can be seen, there are some other frameworks that provide either distributed architectures or some partial modeling of the uncertainty and vagueness, but with certain limitations:

- Several frameworks provide a distributed architecture, which each element residing in a different machine, but the inference is not performed in a *coordinated*, *distributed* way. The inference mechanism is either centralized on a single element or takes places in different elements that do not have any type of coordination between them.

- Korpiää et al. take into account the context information uncertainty and vagueness, but these metrics do not interact with each other during the reasoning process. As will be shown on Chapter 3, the method proposed in this thesis combines both metrics into the inference process, and the uncertainty values of the inferred facts are directly affected by the degrees of truth (the membership function of the fuzzy facts).

	Context Uncertainty	**Context Vagueness**	**Reasoning**	**Notes**
Stick-e Documents	No	No	-	-
Context Toolkit	No	No	-	-
Aura	No	No	-	-
Gaia	No	No	Distributed architecture, Centralized Reasoning	The architecture is distributed, but the reasoning is performed in a single element.
Korpiää et al.	Confidence	Fuzzy Sets	Centralized	Uncertainty and Vagueness do not interact with each other during the reasoning process.

Table 2.1: Comparison of the analysed systems.

	Context Uncertainty	Context Vagueness	Reasoning	Notes
SOCAM	No	No	Distributed architecture, Centralized Reasoning	Multiple context interpreters, but each one performs an independent centralized reasoning, not communicating with the others. They do not solve a common inference problem.
CoBrA	No	No	Centralized	-
CITRON	No	No	Distributed architecture, Centralized Reasoning	-
Gu et al.	No	No	Centralized	-
AlarmNet	No	No	Centralized	-
InCarMusic	No	No	Centralized	-

Table 2.2: Cont. of Table 2.1

2.2 Semantic Context Modeling

As discussed in (Strang and Linnhoff-Popien, 2004), modeling the context with ontologies offers the following advantages:

- Ontologies are the most expressive approach to model context.
- Composition and management of the model can be done in a distributed manner.
- It is possible to partially validate the contextual knowledge.
- One of the main strengths of ontologies is the simplicity to enact the normalization and formality of the model.

Multiple ontology based context models have been developed in the past. In this section the most relevant models will be discussed.

2.2.1 SOUPA

SOUPA(Chen et al., 2004) is a set of ontologies oriented to ubiquitous and pervasive applications used by the COBRA project(Chen, 2004). It is composed by two sub-sets: SOUPA Core and SOUPA Extensions (see Figure 2.14). The SOUPA Core defines the elements that are present in any ubiquitous application, while SOUPA Extensions support more specific applications.

Elements of the SOUPA ontology are extracted from other ontologies and are mapped using the *owl:equivalentClass* and *owl:equivalentOntology* properties. Using these properties instead of importing the whole ontology the overheat is reduced.

2.2.1.1 SOUPA Core

The core of the SOUPA ontology defines the following elements: People, agents, BDI (belief-desire-intention), actions, policies, time, space and events.

Person

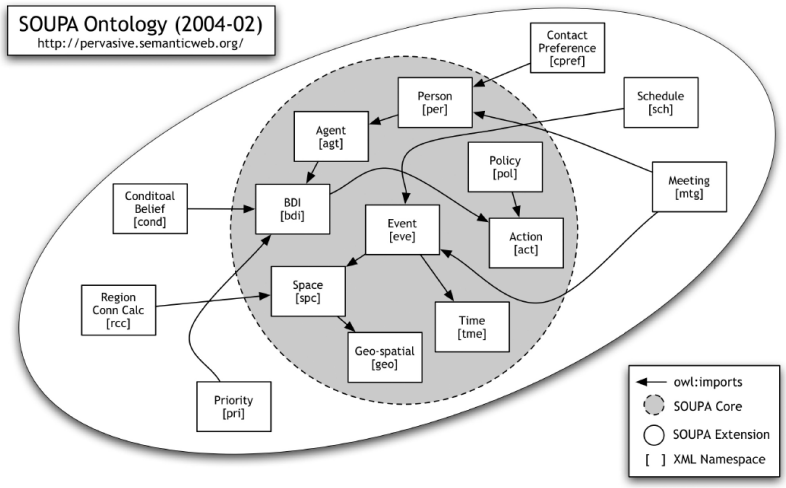

Figure 2.14: SOUPA Ontology, extracted from (Chen et al., 2004)

This element defines the necessary vocabulary to express the profile and contact information of a person. It is equivalent to the *foaf:Person* class from the FOAF[1](Friend of a friend) ontology.

Action

This class is used to define actions in the system: who performs the action, who receives it, the related object, the location and time of the action, the device used in the action.

Policy

This class models the security and privacy policies of the system. It also provides a descriptive logic mechanism to reason over them. It is based on the Rei(Kagal et al., 2003) policy language. These policies are applied to the actions defined with the previous class, stating whether an action can take place or not.

BDI

This class is used to define the state of the software agents and the users of the

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

system, using the belief-desire-intention approach of MoGATU BDI(Perich, 2004). It can describe statements and plans using a subclass of *act:action* which has the extra properties *bdi:precondition* and *bdi:effect*. Agent's desires can be unachievable or be in conflict with other agent's desires.

Agent

This class is used to model the software agents of the system.

Time

This element of the ontology allows to model time instants, intervals and the relations between them. It can be used to describe the events that take place in the system. This ontology uses the vocabulary of the DAML-Time[1] ontology.

Space

This class models the locations and spatial relations of the elements of the system, expressed both as symbolic values and as geo-spatial coordinates. It is based on the OpenCyc[2] ontology and OpenGIS[3] language.

Event

This class is used to model activities that have spatial and temporal components. E.g. the discovery of a Bluetooth device at a certain time and place.

2.2.1.2 SOUPA Extensions

This set of ontologies has been designed with two objectives in mind: define a vocabulary to model applications related with meetings in smart spaces and to prove how the SOUPA Core can be extended to cater to specific scenarios.

The defined ontologies are:

- Meeting & Schedule.
- Document & Digital Document.

[1] http://www.cs.rochester.edu/ ferguson/daml/
[2] http://www.opencyc.org/
[3] http://www.opengeospatial.org/standards/gml

- Image Capture.
- Region Connection Calculus.
- Location.

2.2.2 CONON

CONON(Wang et al., 2004) is used by the SOCAM(Gu et al., 2004) project to model the context of pervasive computing applications. It is also divided into two sets, one with the general information shared between all the applications and the other one domain specific (see Figure 2.15).

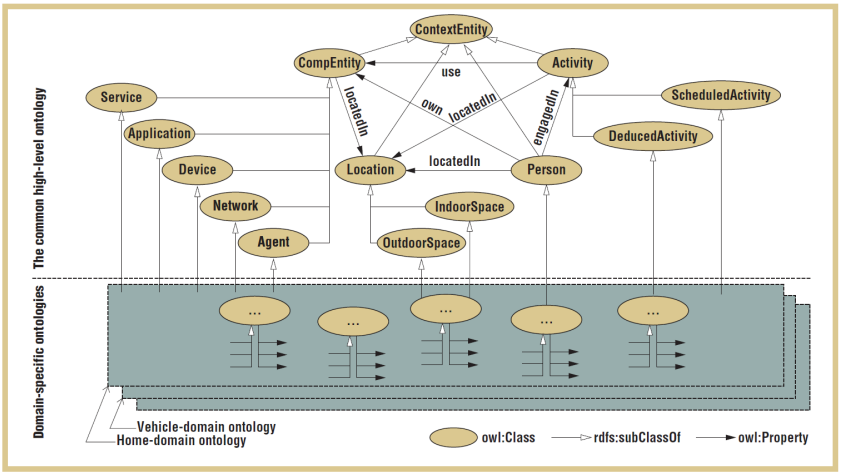

Figure 2.15: CONON Ontology, extracted from (Gu et al., 2004)

For the core ontology, five main entities are defined:

1. Location
2. User
3. Activity
4. Computational Entity
5. Context Entity

2.2.3 CODONT

CODONT(Preuveneers et al., 2004) is used by the CODAMOS project and its main aim is to create an adaptable and flexible infrastructure for AmI applications. To achieve this it is focused on four entities: User, Service, Platform and Environment (see Figure 2.16).

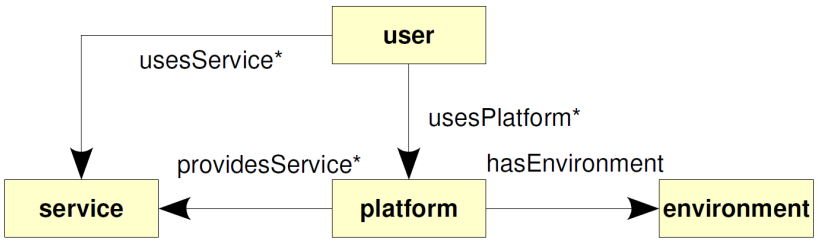

Figure 2.16: CODONT Ontology, extracted from (Preuveneers et al., 2004)

User

In the CODONT ontology, context information is only important if it affects the users. The user class has several properties(see Figure 2.17):

- Preferences, which change depending on the situation.
- Profile, which is static.
- Task, which can be divided into activities.
- Role, which describes the user's role in the system.
- Mood, which affects the preferences.

Environment

CODAMOS models three elements of the environment(see Figure 2.18): location, time and environmental conditions (meteorology, temperature...). The environment is not associated with a specific user, instead, it is related to the devices

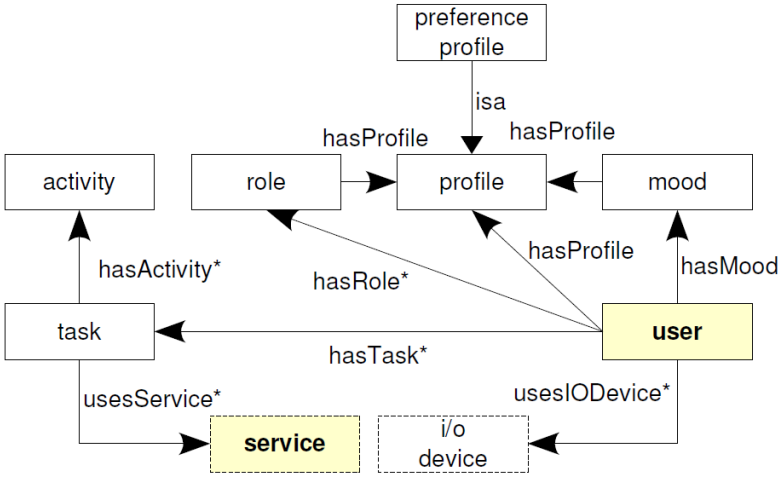

Figure 2.17: User concepts, extracted from (Preuveneers et al., 2004)

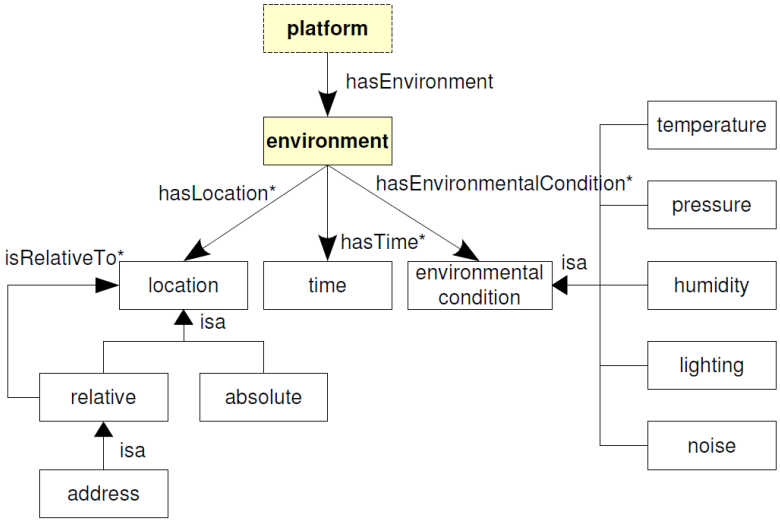

Figure 2.18: Environment concepts, extracted from (Preuveneers et al., 2004)

that capture it.

Platform

This element describes both the hardware and the software that are available for the user(see Figure 2.19). It is necessary to known which software is available in a device to infer if a service can be executed in it and to build device specific services automatically. It is also important to know the hardware of a device to be able to infer its capabilities.

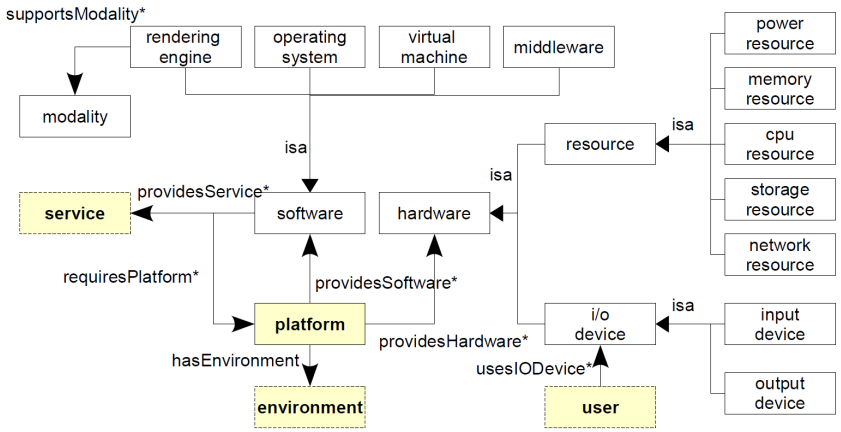

Figure 2.19: Platform concepts, extracted from (Preuveneers et al., 2004)

Services

This element models the computational entities that offer an specific functionality(see Figure 2.20). It uses three aspects to describe a service:

1. Service profile: a human readable description of the service functionality

2. Service model: the service workflow

3. Service grounding: details about the implementation of the service.

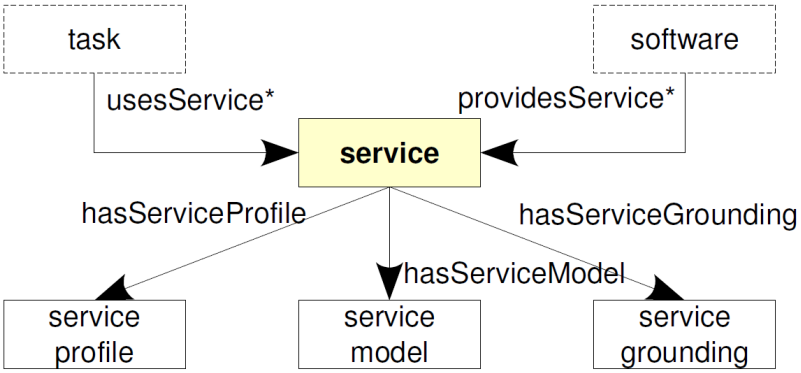

Figure 2.20: Services concepts, extracted from (Preuveneers et al., 2004)

	SOUPA	**CONON**	**CoDAMoS**
Main Concepts	Person, Agent, BDI, Policy, Event, Action, Time, Space	Location, Person, Activity, Computing Entity, Context Entity	User, Service, Platform, Location, Time, Enviromental condition

Table 2.3: Main concepts of the ontologies

2.2.4 Comparison

These three ontologies have several common elements and some unique concepts. The main concepts of the ontologies can be seen in Table 2.3

The common concepts in the tree ontologies are:

- Space/Location: Location is one of the most important elements of the contextual information. It is important to not only model the locations of the environment, but also the spatial relationships between the different elements.
- Time: Both SOUPA and CODAMOS model the time instants and intervals. SOUPA also defines temporal relationships allowing to model the workflow of the actions and the events.
- Person/User: Users are the central element of every Intelligent Environment. The three ontologies allow to model the users. With SOUPA and CODAMOS it is also possible to model the mental state of the user.
- Action/Activity: SOUPA and CONON define concepts that allow to describe the user's actions.
- Computing Entity/Platform: CONON and CODAMOS model the devices that are present in the Intelligent Environment.

The main problem with these ontologies is not *what* do they model, but *how* they model it. These ontologies do not take into account the uncertainty and vagueness of the context information, losing expressivity. In the next section some approaches to modeling uncertainty and ontologies will be described.

2.3 Modeling Context Uncertainty and Vagueness Using Ontologies

Several authors have worked on combining uncertainty or vagueness with ontologies. An extensive survey can be found in (Lukasiewicz and Straccia, 2008). In the case of the uncertainty, in (Costa et al., 2008) the authors present a probabilistic generalization of OWL called PR-OWL, based in Multi Entity Bayesian Networks

(MEBNs) which allows the combination of first order logic with Bayesian logic. This ontology represents the knowledge as parameterized fragments of Bayesian networks. In (Ding et al., 2006) the authors propose another probabilistic generalization of OWL called BayesOWL which also uses Bayesian networks. The authors suggest a mechanism which can translate an OWL ontology to a Bayesian network, adding probabilistic restrictions when building the network. The created Bayesian network maintains the semantic information of the origin ontology and allows ontological reasoning modeled as Bayesian inference. The authors in (Yang and Calmet, 2005) describe another integration of OWL with Bayesian networks, a system named OntoBayes. It uses an OWL extension annotated with probabilities and dependencies to represent the uncertainty of Bayesian networks. These probabilistic extensions are not confined to OWL only; in (Nottelmann and Fuhr, 2006) an extension for OWL Lite is discussed and in (Fukushige, 2004) and (Udrea et al., 2006) extensions for RDF are presented. Several authors have also addressed the combination of the vagueness (represented as the usage of fuzzy sets) with ontologies. In (Stoilos et al., 2006) authors analyze how SHOIN could be extended adding the possibility of using fuzzy sets (f-SHOIN). They also propose a fuzzy extension for OWL. In (Bobillo and Straccia, 2009) authors describe a fuzzy extension for SROIQ(D) and present an Fuzzy OWL2 Ontology. In (Parry, 2004) a fuzzy ontology for the management of medical documents is discussed. This ontology can store different membership values. Additionally the author has created a mechanism based in the occurrence of keywords in the title, abstract or body of the document to calculate the membership value of the different categories. In (Lee et al., 2005), its authors describe a fuzzy ontology used to automatically create summaries of news articles. These authors have also created a mechanism for the automatic creation of the fuzzy ontology based on the analysis of the news. Finally, in (Tho et al., 2006) the authors propose a mechanism to create automatically fuzzy ontologies. The created ontologies include the membership values of the different terms.

2.3.1 Comparison

In Table 2.4 a summary of the state of the art review is shown. As can be seen authors decide to use either uncertainty or vagueness in their models, without combining them. The proposed solution uses both approaches, enriching the context model with both uncertainty and vagueness. It also takes into account the interaction between them (as explained in Chapter 3), assessing how the vagueness modifies the resulting certainty factor. By assessing both aspects of ambiguity, the resulting system is able to model a broader range of situations and interactions in smart environments.

	Uncertainty	**Vagueness**
da Costa et al.	Yes	No
Ding et al.	Yes	No
Yang et al.	Yes	No
Nottelman et al.	Yes	No
Fukushige	Yes	No
Udrea et al.	Yes	No
Stoilos et al.	No	Yes
Bobillo et al.	No	Yes
Parry	No	Yes
Lee et al.	No	Yes
Tho et al.	No	Yes

Table 2.4: Comparison of the different approaches to modeling uncertainty and vagueness

2.4 Distributed Reasoning in Intelligent Environments

Several authors have tackled the problem of the inference distribution in scenarios where the knowledge is not centralized. This decentralization can occur for several reasons:

- The nature of the selected domain. For example the peer-to-peer architecture of the sensor networks or the distribution of knowledge in the Semantic Web.

- The need to distribute the reasoning to be able to process large sets of data

In (Serafini and Tamilin, 2005) authors present an algorithm that addresses the problem of reasoning with multiple related ontologies. Their system interconnects multiple OWL ontologies, using mappings between concepts and applies a distributed tableau reasoning technique. Their approach is to create a distributed algorithm while we share the knowledge between a network of inference engines based on their declared interests.

In (Oren et al., 2009) authors present a parallel and distributed platform for processing large amounts of distributed RDF data. They discuss a divide-conquer-swap strategy that converges towards completeness. The differences with the approach presented in this dissertation are two:

1. The propose approach splits the problem in loosely coupled subparts while they see it as a complete problem

2. They do the fact sharing process automatically while in our case this process is dictated by the previously expressed interests of each context consumer

In (Adjiman et al., 2006) authors present a consequence finding algorithm for peer-to-peer settings. The algorithm computes consequences gradually from the solicited peer to peers that are increasely distant. Authors argue that centralizing the reasoning in a single server in peer-to-peer systems is not feasible for two reasons: first it would be costly to gather the data available through the system and second it would be useless because of the dynamicity of these types of networks, where peers join and leave the system continuously. In (Chu et al., 2003), authors introduce an architecture for designing distributed inference algorithms in ad-hoc sensor networks. The architecture is centered in solutions for sensor networks and takes into account the movement of the nodes. In (Cabitza et al., 2005) authors present a Java package that provides programmers with a middleware that allows to ensemble different inference systems implemented in Jess transparently. The architecture presented in this dissertation, the authors propose a middleware that offers temporal and spatial decoupling of the inference, but their middleware is linked to one specific reasoner, Jess. In our case, due to the usage of OWL ontologies to model the context, different semantic reasoning engines can be used in

each context consumer seamlessly. In (Paskin et al., 2005) authors introduce an architecture for distributed inference in sensor networks. This architecture is robust to unreliable communication and node failures. To do this the nodes of the network assemble in a stable spanning tree, and later transform the spanning tree into a junction tree for the inference problem. Finally, in (Guestrin et al., 2007), authors present a distributed algorithm for approximate probabilistic inference in dynamical systems. This distributed approach to approximate dynamic filtering is based on a distributed representation of the assumed density in the network.

The approach proposed in this dissertation differs from the previously discussed systems in the following aspects:

1. It is not based on an ad-hoc distributed inference algorithm. We split the reasoning problem in different loosely coupled subparts according to the interests expressed by the context consumers. This allows us to use standard modeling techniques (OWL ontologies) and reasoning engines (in our case semantic reasoning engines like Pellet(pel) or Jena 2 Inference Support(apa)).

2. We take into account the certainty factor of the data. This allows us to filter low quality data, reducing the inference time lost in useless reasoning.

3. We also take into account the location of the context data in order to filter the information received by the Context Consumers.

2.4.1 Contract Net Protocol

The Contract Net Protocol (CNP) is a widely used protocol in multi-agent systems. It has proved to be a flexible and low communication interaction protocol for task assignment(Knabe et al., 2002). For this reason CNP is the selected protocol to implement the distributed reasoning architecture described in Chapter 4.

CNP has been used since the early 80s (Smith, 1980)(Smith and Davis, 1981) and it has received several improvements since then(Xu and Weigand, 2001), like the ones described in (Sandholm, 1993)(Weigand et al., 1998)(Sachs et al., 2000). Multi-agent systems based on the Contract Net Protocol have been used to tackle several problems. In (Ouelhadj et al., 2005), authors discuss a system for efficient

job scheduling on the Grid using multi-agent systems and a Service Level Agreement (SLA) negotiation protocol based on the Contract Net Protocol. The proposed system is composed by three types of agents: user agents, local scheduler agents, and super scheduler agents. In (Hsieh and Chiang, 2009), authors propose a multi-agent system with Petri nets to design and implement Holonic Manufacturing Systems (HMS) to fulfill the requirements of orders. Authors describe an architecture and a two-layer contract net protocol for planning order holons, product holons and resource holons in HMS. In (Huang et al., 2011) authors use a multi-agent system based on the contract net protocol for reservoir flood control optimization. Finally in (Ouelhadj et al., 1998) and (Ouelhadj et al., 2000) authors describe a multi-agent architecture for dynamic scheduling in flexible manufacturing systems which involves only resource agents using contract net protocol.

2.5 Summary

The contributions of this dissertation will be focused in the two areas (see Figure 2.21) highlighted in the state of the art analysis: a semantic reasoning model that takes into account uncertainty and vagueness and a distributed reasoning architecture that allows distributing the global inference task among a ecosystem of reasoning engines.

Taking into account the uncertainty and vagueness the context management process will be able to better model the environment and to achieve more informed decisions, as will be explained in Chapter 3 and Chapter 5.

Distributing the global inference task will achieve several objectives: Attain the temporal and spatial decoupling of the different inference engines, reduce the number of triples that each reasoning engine has to process, compartmentalize the information according to the different interests of the reasoning engines and allow the dynamic, on-the-fly reorganization of the reasoning architecture to adapt to the domain needs. This will be explained in Chapter 4 and Chapter 5.

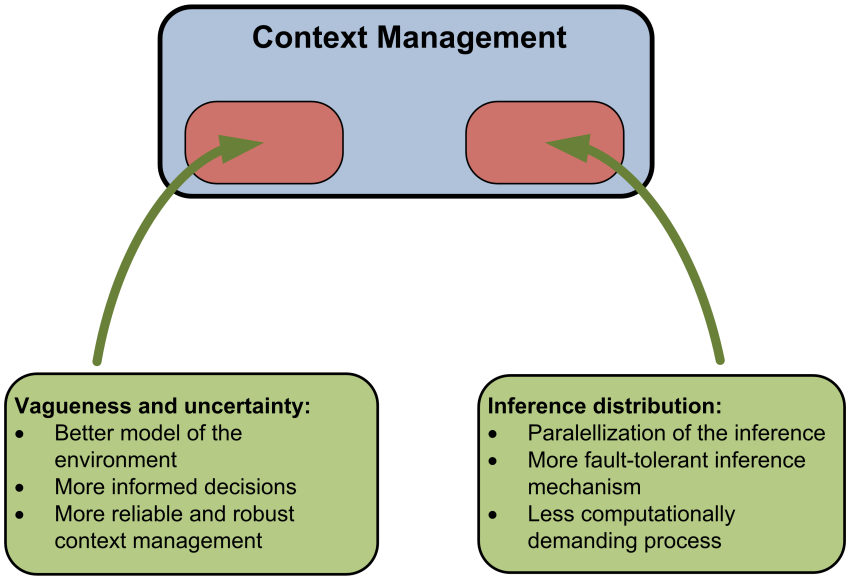

Figure 2.21: Contributions to the state of the art

There is no such uncertainty as a sure thing.

Robert Burns

CHAPTER

3

Ambiguity in Context Data

Intelligent environments host a diverse and dynamic ecosystem of devices, sensors, actuators and users. When modeling real environments certainty cannot be taken for granted. Reality, and hence the user context (Dey, 2001), is ambiguous. Sensors and devices are not perfect and their measurements carry a degree of uncertainty; for example, several thermometers in the same room can provide conflicting temperature values and there always exists the human factor. Not every user can provide the exact temperature they want for their bath—most of them will only say that they want it "warm". For this reason, when developing smart spaces and ambient intelligence applications, it is important to address ambiguity in order to model the context more realistically. To provide our systems with this feature, we have centered our work on two aspects of ambiguity: uncertainty and vagueness. We use uncertainty to model the truthfulness of the different context data by assigning to them a certainty factor (CF). This way we can know the reliability of each piece of information and act accordingly. This knowledge also allows us to create a more robust data fusion process to resolve the problem of the existence of multiple providers for the same piece of information at the same location. On the other hand, vagueness helps us to model those situations where the boundaries between categories are not clearly defined. This usually occurs when users are involved. Different users will have different perceptions about what constitutes a

cold room or a noisy environment. We have addressed this problem using fuzzy sets to model the vagueness. By taking into account ambiguity in contextual information, the aim is to improve the reliability of context management systems. As Black argues, vagueness should not be equated with subjectivity (Black, 1937). From our point of view, as will be discussed, modeling uncertainty and vagueness improves the precision of the system. With this information the system is able to better assess the actual state of the context, being able to react to a broader range of situations.

3.1 AMBI2ONT: An Ontology for the Ambiguous Context

Modeling the context in ubiquitous computing systems encompasses several challenges. Strang and Linnhoff-Popien (Strang and Linnhoff-Popien, 2004) identify six requirements that this type of systems present for modeling the context:

1. *Distributed composition:* The distributed nature of ubiquitous computing systems precludes the existence of a central element that creates, deploys and maintains the context data. The different distributed elements of the system contribute to create the context model. The multi-agent architecture of our context management platform, where the context data is created and processed in parallel in several elements, demands that the context model must support the distributed composition.

2. *Partial validation:* The context model must be able to partially validate both the instances and structure of the context data in the different elements of the system. In this dissertation, the existence of multiple context consumers that work with different parts of the context makes this requirement even more important to assure the consistency of the context data.

3. *Richness and quality of information:* The quality of the data provided by sensors can vary over time. For this reason the context model must inherently support quality and richness information.

4. *Incompleteness and ambiguity:* A real fact is that the context data available in ubiquitous computing systems is generally incomplete or ambiguous. The context model must, therefore, be able to deal with these situations.

5. *Level of formality:* The selected model must be able to precisely describe the context, providing a shared understanding between the elements of the architecture. This level of formality and the adoption of knowledge modeling standards may also foster the reuse and integration of the context model set forward by this dissertation.

6. *Applicability to existing environments:* The context model must be appropriate for the already existing ubiquitous infrastructures.

Following these requirements (see Table 1 in (Strang and Linnhoff-Popien, 2004)) Strang and Linnhoff-Popien identify ontology based models (Uschold and Gruninger, 1996) as the most suitable ones for context modeling in ubiquitous computing systems. Studer et al(Studer et al., 1998) define ontologies as *"formal, explicit specification of a shared conceptualization"*.Ontologies are a formal representation of shared concepts, conceptual frameworks that model the elements of a domain and their relations, using a common vocabulary. In information science, ontologies can be used to model a domain and to reason over its elements. In our system, we have used the OWL 2 Web Ontology Language to model the context. The W3C defines OWL 2(owl, a) as *"an ontology language for the Semantic Web with formally defined meaning. OWL 2 ontologies provide classes, properties, individuals, and data values and are stored as Semantic Web documents. OWL 2 ontologies can be used along with information written in RDF, and OWL 2 ontologies themselves are primarily exchanged as RDF documents"*. Although the primary exchange syntax for OWL 2 is RDF/XM (rdf, b) and it must be supported by all OWL 2 tools, it also supports several other syntaxes: OWL/XML(owl, e), Functional Syntax(owl, d), Manchester Syntax(owl, b) and Turtle(tur).

OWL 2 is divided in three sublanguages that offer different advantages. These sublanguages are called profiles: OWL 2 EL, OWL 2 QL, and OWL 2 RL. According to the OWL 2 Overview(owl, a): *"OWL 2 EL enables polynomial time*

```
<owl:Class rdf:ID="Temperature">
    <rdfs:subClassOf rdf:resource="#ContextData"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="locatedIn">
    <rdfs:range rdf:resource="#Location"/>
    <rdfs:domain rdf:resource="#Temperature"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="providedBy">
    <rdfs:range rdf:resource="#Thermometer"/>
    <rdfs:domain rdf:resource="#Temperature"/>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="measure">
    <rdfs:range
     rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
    <rdfs:domain rdf:resource="#Temperature"/>
</owl:DatatypeProperty>
```

Figure 3.1: The concept of temperature serialized in RDF/XML format.

algorithms for all the standard reasoning tasks; it is particularly suitable for applications where very large ontologies are needed, and where expressive power can be traded for performance guarantees. OWL 2 QL enables conjunctive queries to be answered in LogSpace (more precisely, AC0) using standard relational database technology; it is particularly suitable for applications where relatively lightweight ontologies are used to organize large numbers of individuals and where it is useful or necessary to access the data directly via relational queries (e.g., SQL). OWL 2 RL enables the implementation of polynomial time reasoning algorithms using rule-extended database technologies operating directly on RDF triples; it is particularly suitable for applications where relatively lightweight ontologies are used to organize large numbers of individuals and where it is useful or necessary to operate directly on data in the form of RDF triples". The profile used in our system has been OWL 2 RL. Context ontologies deal with a large number of instances and the RL profile is the most suitable one in those cases. One of the problems encountered by us whilst modeling context data previous ubiquitous environments is the uncertainty and vagueness of the gathered information. In preliminary work of this dissertation(Almeida et al., 2009), none of this information was used, which led to a loss of important data like the certainty of the measures taken by the sensors. During the work carried out for the Imhotep[1] framework(Almeida et al., 2011), as part of the initial phase of this dissertation, fuzzy terms were used to describe a small part of the context (the capabilities of mobile devices and users) in a human-friendly manner. One of the objectives of this dissertation is to develop a framework capable of managing the ambiguity and incertitude that often characterizes the reality. To do this, an ontology modeling these concepts have been created.

As shown in Figure 3.2 the main elements of the ontology are *Location*, *LocableThing* and *LinguisticTerm*.

Location

The subclasses of this class model the spatial information of the context(see Figure 3.3). Thanks to the spatial relations, the system can identify those entities that are collocated in the same room, calculate distances, find nearest objects... The elements in this hierarchy are divided between points and two-dimensional spaces.

[1] http://www.morelab.deusto.es/imhotep/

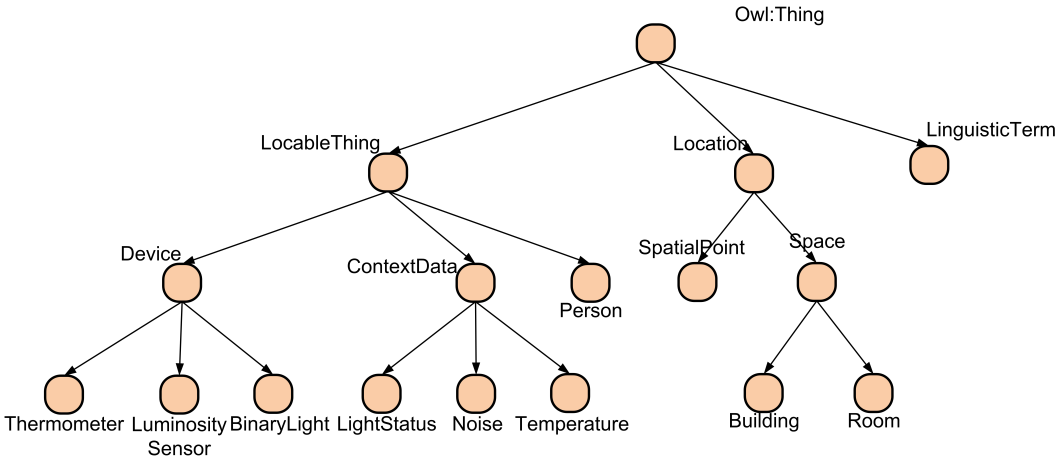

Figure 3.2: Subset of the main ontology concepts.

Points are represented by the *SpatialPoint* class. This element has the following properties:

- *x* and *y* values of the point.
- *limits:* expresses if the point is part of the limits of a two-dimensional space.
- *contains:* expresses if a element is located in that point.
- *subsumed_by:* expresses if that point is located inside a two-dimensional space.

Two-dimensional spaces are divided into *Buildings* and *Rooms*. They have the following properties:

- *name:* the name of the space
- *is_adjacent:* expresses which other elements are located near the space.
- *is_limited_by:* expresses which points define the space.
- *contains:* expresses which elements are contained by the space.
- *subsumes:* expresses which other spaces are subsumed by the space.

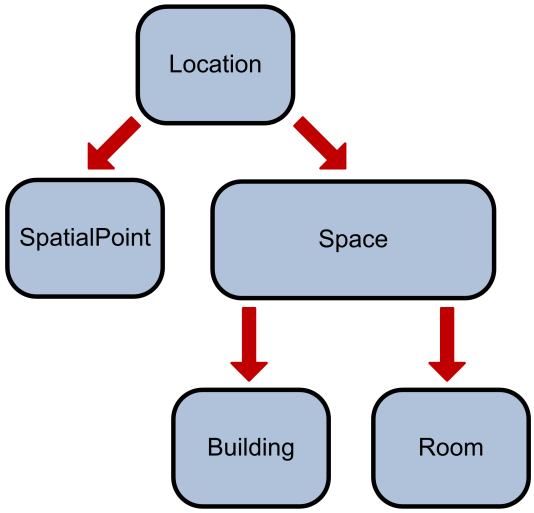

Figure 3.3: Location hierarchy.

- *subsumed_by:* expresses if it is located inside another space.

LocableThing

The subclasses of this class represent the elements of the system that have a physical location(see Figure 3.4). It contains three subclasses: the *Person* subclass represents the users, the *Device* subclass models the different devices of the environment and the *ContextData* subclass models the measurements taken by the devices or inferred by the data fusion process. As it will be explained, there are two types of measurements in the system, those taken by the devices and the global measures for each room calculated by our data fusion mechanism. All the *LocableThing*-s share some common properties:

- *name:* the name of the element.
- *placed_in:* the *Location* of the thing.
- *collocated_with:* expresses which other elements are located in the same location of the *LocableThing*.
- *is_contained_by:* expresses if a *LocableThing* is contained by a *Space*
- *is_adjacent:* expresses which other elements are in adjacent locations.

Device-s have a *provides* property that expresses which *ContextData* is provided by that device. *ContextData* have a *provided_by* property that expressed which *Device* has captured that data. The different measures that are children of *ContextData* also have a category that sorts them into different groups. This enables the multi-agent system described in the next chapter to perform searches using broader concepts. Every *ContextData* also has a *crisp_value* property indicating the crisp value of the measure, a *certainty_factor* and several *linguistic_terms*. In the next subsection, more details will be given about how uncertainty and vagueness are modeled.

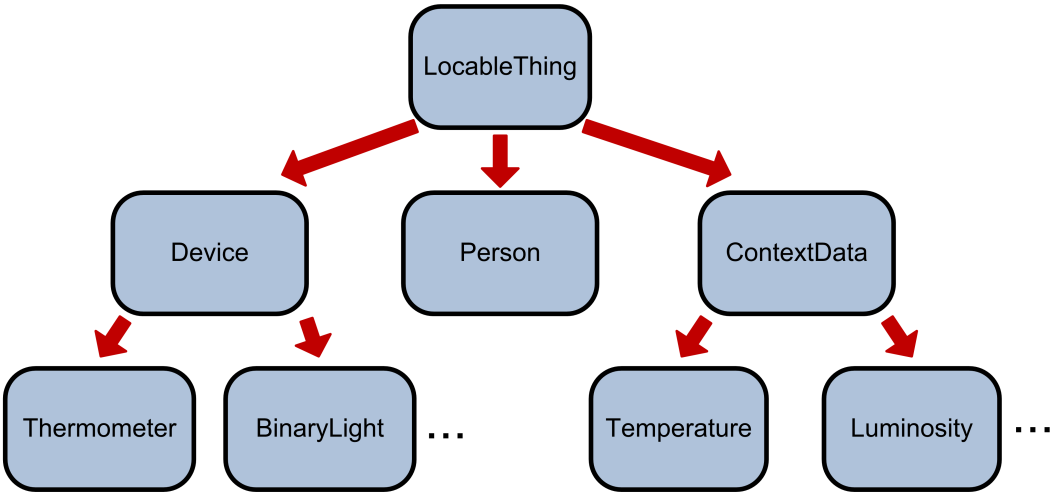

Figure 3.4: LocableThing hierarchy.

LinguisticTerm

This class models the fuzzy linguistic terms of the values of the context data. The ontology only stores the linguistic term and membership value of each individual of context data. Currently, the ontology does not model the membership functions and rules used by the inference engine.

3.1.1 Modeling Uncertainty and Vagueness

Our ontology models two aspects of the ambiguity of the context data: the *uncertainty* (represented by a certainty factor, CF) and the *vagueness* (represented by

fuzzy sets). Uncertainty models the likeliness of a fact, for example *"the temperature of the room is 27 °C with a certainty factor of 0.2 and 18 °C with a certainty factor of 0.8"* means that the value of the temperature is more probably 18 °C (but it cannot be both of them). In the case of vagueness, it represents the degree of membership to a fuzzy set. For example *"the temperature of the room is cold with a membership of 0.7"* means that the room is mostly cold. In Figure 3.5, it can be seen how those values are stored in the ontology. Each *ContextData* individual has the following properties:

- *crisp_value:* the measure taken by the associated sensor. In our system, a sensor is defined as anything that provides context information.
- *certainty_factor:* the degree of credibility of the measure. This metric is given by the sensor that takes the measure and takes values between 0 and 1.
- *linguistic_term:* each measure has its fuzzy representation, represented as the linguistic term name and the membership degree for that term.

This can be seen in the example shown on Figure 3.5. The temperature measurement has a crisp value of 32 °C with a certainty factor of 0.7. After processing that crisp value with the associated membership functions, it can be inferred (using the domain specific fuzzy membership functions) that the membership degree for cold is 0, for mild is 0.1 and for hot is 0.9; so the room is mainly hot. With the inclusion of uncertainty the proposed model fulfills the *"richness and quality of information"* requirement from the Strang and Linnhoff-Popien (Strang and Linnhoff-Popien, 2004) list. AMBI2ONT is able to model the quality of the context data using a certainty factor. Also, with the combination of the uncertainty and the vagueness, it fulfills the *"incompleteness and ambiguity"* requirement, being able to model the ambiguity that naturally occurs in real environments. It is important to note that the proposed ontology and reasoning engine supports two types of uncertainty:

- *Uncertain data.* It is the uncertainty originated from the capture of the context data by the devices. This uncertainty is created by the imperfect nature

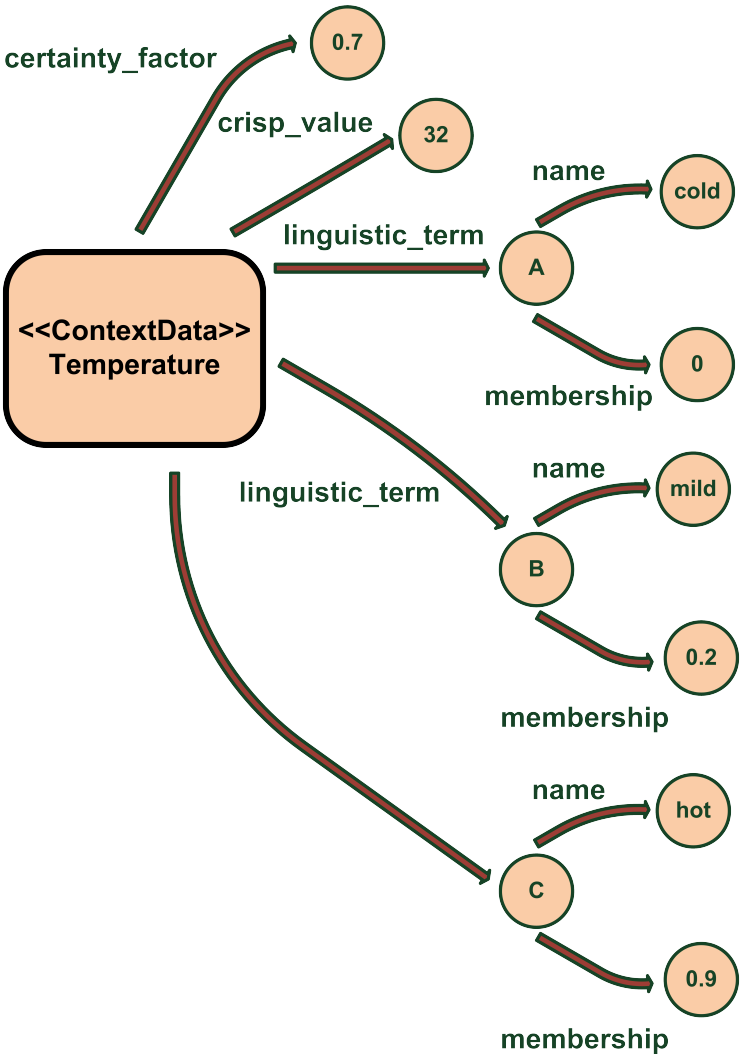

Figure 3.5: Example of the ambiguity data for a temperature measure stored in the ontology.

```
RULEBLOCK r1
    AND : MIN;
    ACT : MIN;
    ACCY : MAX;

    RULE 1 : CF 0.6
             IF PRESSURE IS HIGH AND
             TEMPERATURE IS LOW
             THEN PRECIPITATION IS RAIN
END_RULEBLOCK
```

Figure 3.6: An example of an uncertain fuzzy rule.

of the devices (see Figure 3.5 for an example). This uncertainty in the ontology using the *certainty_factor* property. An example of this uncertainty would be a measure from a thermometer stating that the temperature is 20ºC with a CF of 0.8.

- *Uncertain rules.* It is the uncertainty that occurs in the execution of the rules. Although the inferred context will be inserted in the ontology with the resulting certainty factor, the rule itself is not part of the ontology (an example of these rules can be seen in Figure 3.6). This uncertainty implies that the outcome of a rule is not fixed. An example of this uncertainty would be a rule that given the facts temperature low and barometric pressure high infers that precipitation rain with a CF of 0.7. This means that not always rains when the temperature is low and the pressure high, but there is a good chance of it happening.

3.2 Semantic Context Management for Ambiguous Data

The semantic context management is done in four steps (see Figure 3.7):

1. Add the measures to the ontology.

2. Process the semantic and positional information.
3. Apply the data fusion mechanism.
4. Process the ambiguity contained in the data.

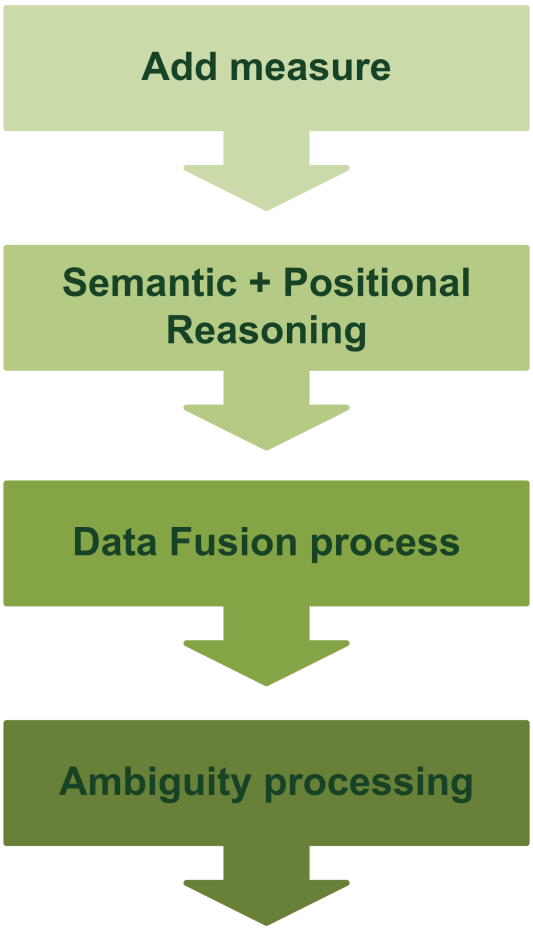

Figure 3.7: Context management process.

3.2.1 Adding the Measure

To add a measurement to the ontology, the sensor must provide the measurement type, its value, location and a certainty factor. It is assumed that each sensor, as part of its datasheet, knows or specifies its certainty factor based on its type and

manufacturer. It is also assumed that the certainty factor of the sensor can change over time depending on the environment (e.g., a thermometer can be pretty accurate for temperatures between -10 °C and 50 °C, but the measurement quality can degrade outside that range). For that reason, the sensor certainty factor is not stored in the ontology when the sensor registers itself.

3.2.2 Processing the Semantic and Spatial Data

Once the measurements have been added, a semantic inference process is applied to achieve two goals:

1. Make explicit the hidden implicit knowledge in the ontology.

2. Infer the positional information of each measurement.

To do this, two different sets of rules are used: the semantic rules and the spatial heuristic rules. To make the semantic reasoning less cumbersome, we implement a subset of the RDF Model Theory(rdf, a) and the OWL Model Theory(owl, g). An example of the used rules can be seen in Figure 3.8.

The spatial heuristic rules are used to infer higher level spatial information from the coordinates provided by the sensors. This information comprises data like the room in which the sensor is located; the devices, people and sensors surrounding it and the relative location to other *LocableThing*-s. To do this inference, the system uses six spatial inference rules. The first rule is used to infer if a *SpatialPoint* is located inside a *Space* as can be seen in Figure 3.9 The rule used to infer in which *Space* is located a *LocableThing* given its coordinates is show in Figure 3.10. Three rules are used to infer which *LocableThing*-s are placed in the same location and collocated with each other: Figure 3.12, Figure 3.13 and Figure 3.11. Finally, the last rule is used to infer which *LocableThing*-s are near each other but not in the same location (e.g, A mobile phone is located in Room A and a user in Room B, been both adjacent rooms) as can be seen in Figure 3.14.

```
[rdfs9:
    (?x rdfs:subClassOf ?y),
    (?a rdf:type ?x)
    ->
    (?a rdf:type ?y)
]

[OWL-InverseOf-A:
    (?x ?prop1 ?y),
    (?prop1 owl:inverseOf ?prop2),
    ->
    (?y ?prop2 ?x)
]
```

Figure 3.8: An example of the implemented semantic rules. The first rule is extracted from the RDF Model Theory and models the transitivity of the subClassOf relationship. The second rule is extracted from the OWL Model Theory and models the behavior of the inverse.

```
[SPATIAL1-pointInSpace:
   (?loc1 rdf:type <ambi2ont:SpatialPoint>),
   (?loc1 <ambi2ont:x> ?x),
   (?loc1 <ambi2ont:y> ?y),
   (?area1 rdf:type <ambi2ont:Space>),
   (?area1 <ambi2ont:is_limited_by> ?p1),
   (?area1 <ambi2ont:is_limited_by> ?p2),
   (?area1 <ambi2ont:is_limited_by> ?p3),
   (?area1 <ambi2ont:is_limited_by> ?p4),
   (?p1 <ambi2ont:x> ?p1x),
   (?p1 <ambi2ont:y> ?p1y),
   (?p2 <ambi2ont:x> ?p2x),
   (?p2 <ambi2ont:y> ?p2y),
   (?p3 <ambi2ont:x> ?p3x),
   (?p3 <ambi2ont:y> ?p3y),
   (?p4 <ambi2ont:x> ?p4x),
   (?p4 <ambi2ont:y> ?p4y),
   greaterThan(?x, ?p1x),
   greaterThan(?y, ?p1y),
   lessThan(?x,?p3x),
   lessThan(?y,?p3y),
   greaterThan(?x, ?p2x),
   lessThan(?y,?p2y),
   greaterThan(?y, ?p4y),
   lessThan(?x,?p4x),
   ->
   (?loc1 <ambi2ont:is_contained_by> ?area1)
]
```

Figure 3.9: Rule for positioning a point into a two dimensional space

```
[SPATIAL2-Locableplaced_in:
    (?thg1 rdf:type <ambi2ont:LocableThing>),
    (?thg1 <ambi2ont:placed_in> ?loc1),
    (?loc1 rdf:type <ambi2ont:SpatialPoint>),
    (?loc1 <ambi2ont:is_contained_by> ?area1)
    ->
    (?thg1 <ambi2ont:placed_in> ?area1)
]
```

Figure 3.10: Rule for infering the area that an element is located in

```
[SPATIAL3-Locablecollocated_withLocableInRoom:
    (?thg1 rdf:type <ambi2ont:LocableThing>),
    (?thg1 <ambi2ont:placed_in> ?loc1),
    (?loc1 rdf:type <ambi2ont:Room>),
    (?loc1 <ambi2ont:name> ?name1),
    (?thg2 rdf:type <ambi2ont:LocableThing>),
    (?thg2 <ambi2ont:placed_in> ?loc2),
    (?loc2 rdf:type <ambi2ont:Room>),
    (?loc2 <ambi2ont:name> ?name2),
    equal(?name1,?name2)
    ->
    (?thg1 <ambi2ont:collocated_with> ?thg2)
]
```

Figure 3.11: Rules for identifiying colocated elements in the same room

```
[SPATIAL4-Locablecollocated_withLocableInArea:
    (?thg1 rdf:type <ambi2ont:LocableThing>),
    (?thg1 <ambi2ont:placed_in> ?loc1),
    (?loc1 rdf:type <ambi2ont:Space>),
    (?loc1 <ambi2ont:name> ?name1),
    (?thg2 rdf:type <ambi2ont:LocableThing>),
    (?thg2 <ambi2ont:placed_in> ?loc2),
    (?loc2 rdf:type <ambi2ont:Space>),
    (?loc2 <ambi2ont:name> ?name2),
    equal(?name1,?name2)
    ->
    (?thg1 <ambi2ont:collocated_with> ?thg2)
]
```

Figure 3.12: Rules for identifiying collocated elements in the same area

```
[SPATIAL5-Locablecollocated_withLocableInPoint:
    (?thg1 rdf:type <ambi2ont:LocableThing>),
    (?thg1 <ambi2ont:placed_in> ?loc1),
    (?loc1 <ambi2ont:x> ?x1),
    (?loc1 <ambi2ont:y> ?y1),
    (?thg2 rdf:type <ambi2ont:LocableThing>),
    (?thg2 <ambi2ont:placed_in> ?loc2),
    (?loc2 <ambi2ont:x> ?x2),
    (?loc2 <ambi2ont:y> ?y2),
    equal(?x1,?x2)
    equal(?y1,?y2)
    ->
    (?thg1 <ambi2ont:collocated_with> ?thg2)
]
```

Figure 3.13: Rules for identifiying collocated elements in the same point

```
[SPATIAL6-Locableis_adjacentLocable:
(?thg1 rdf:type <ambi2ont:LocableThing>),
(?thg1 <ambi2ont:placed_in> ?loc1),
(?thg2 rdf:type <ambi2ont:LocableThing>),
(?thg2 <ambi2ont:placed_in> ?loc2),
(?loc1 ambi2ont:is_adjacent> ?loc2),
->
(?thg1 <ambi2ont:is_adjacent> ?thg2)
]
```

Figure 3.14: Rules for identifiying adjacent elements

3.2.3 The Data Fusion Process

Once the location and semantic information of the measurements has been inferred and processed, the data fusion process is applied. From the previous step we can infer that each room can have multiple sensors that provide the same context data (e.g., various thermometers located in the same room). Usually the values and certainty factors of those measurements do not coincide. To be able to take the proper actions we need to process those differing measurements to assess the real status of the room. To tackle this problem, we have created a data fusion mechanism that refines those individual measurements into a single global one for each room. We have implemented two types of strategies for this process: tourney and combination. Using the *tourney strategy*, the measure with the best CF is selected as the global measure of the room. On the other hand, the *combination strategy* has three different behaviors, as stated in (Bloch, 1996):

- *Severe:* The worst certainty factor from all the input measurements is assigned to the combined measurement.
- *Indulgent:* The best certainty factor from all the input measurements is assigned to the combined measurement.
- *Cautious:* An average certainty factor is calculated using the certainty factor from the input measurements.

To determine the combined measurement value we weight the individual values using their certainty factors, as seen in equation 3.1.

$$m_g = \frac{\sum_{i=0}^{n}(m_i \times cf_i)}{\sum_{i=0}^{n}(cf_i)} \tag{3.1}$$

where:

- m_i: the measurement's values
- cf_i: the measurement's certainty factor

It is also possible to set a minimum CF level for the measurements. Measurements with a CF value below this threshold will not be taken into account in the calculations. In Table 3.1, we show an example of how the data fusion process works when the combination strategy is used.

	Value	**CF**	**Global Value**	**CF Severe**	**CF Indulgent**	**CF Cautious**
M1	18	0.7	19.9	0.6	0.8	0.7
M2	22	0.7				
M3	16	0.6				
M4	20	0.8				

Table 3.1: Example of the data fusion.

3.2.4 Processing the Ambiguity

As previously mentioned, two aspects of the ambiguity are modeled: the *uncertainty* and the *vagueness*. To be able to reason over this information, the JFuzzyLogic(jfu) Open Source fuzzy reasoner has been adapted to also accept uncertainty information. JFuzzyLogic follows the Fuzzy Control Lenguage (FCL)(fuz, 2007) standard for its rule language. The modified reasoner supports two types of uncertainty: uncertain data and uncertain rules. The first type occurs when the input data is not completely reliable (as seen in the example shown in Table 3.1). To support this type of uncertain data, the API of the reasoner (see Figure 3.15) has been modified.

```
reasoner.addVariable("Temperature1", 18, 0.7);
reasoner.addVariable("Temperature1", 22, 0.7);
reasoner.addVariable("Temperature1", 16, 0.6);
reasoner.addVariable("Temperature1", 20, 0.8);
```

Figure 3.15: Use of the modified reasoner to process the data of the previous example.

```
//Define the CF token
CF : ('c'|'C')('f'|'F');
//Define a clause that uses a token followed
//by a real number
cf_clause: CF^ REAL;
//Add the clause to the rule body
rule_block : RULEBLOCK^ ID (rule_item)* END_RULEBLOCK!;
rule_item : operator_definition | activation_method
| accumulation_method | rule;
rule : RULE^ rule_name COLON! cf_clause if_clause
then_clause (with)? SEMICOLON! ;
```

Figure 3.16: Modification of the FCL grammar to add the uncertainty.

The second type of uncertainty takes place when the outcome of a rule is not fixed, for example "if the barometric pressure is high and the temperature is low there is a 60% chance of rain". To model this aspect of the uncertainty, the grammar of the FCL language has been modified. JFuzzyLogic uses ANother Tool for Language Recognition (ANTLR)(ant) to generate the FCL parsers. ANTLR is *"a language tool that provides a framework for constructing recognizers, interpreters, compilers, and translators from grammatical descriptions containing actions in a variety of target languages"*.

The grammar file of the FCL language included with JFuzzyLogic has been modified to include information about the certainty of the rules (see Figure 3.16). An example of the modified rules can be seen in Figure 3.6.

Uncertainty and fuzziness can appear in the same rule and influence each other. To tackle this problem, the inference model described in (Orchard, 1998) has been

implemented. This model contemplates three different situations depending on the nature of the antecedent and consequent of the rule and the matching fact: a) *CRISP Simple Rule* where both antecedent and matching facts are crisp values, b) *FUZZY_CRISP Simple Rul*e where both the antecedent and matching facts are fuzzy and the consequent is crisp and finally c) the *FUZZY_FUZZY Simple Rule* where all three are fuzzy. In the case of the *CRISP Simple Rule*, the certainty factor of the consequent is calculated using the equation 3.2.

$$CF_c = CF_r \times CF_f \tag{3.2}$$

where:

- CF_c: the certainty factor of the consequent.
- CF_r: the certainty factor of the rule
- CF_f: the certainty factor of the fact

In the case of the *FUZZY_CRISP Simple Rule*, the certainty factor of the consequent is calculated using equation 3.3, equation 3.4.

$$\begin{aligned} CF_c &= CF_r \times CF_f \times S \\ S &= P(F_\alpha|F'_\alpha) \textit{ if } N(F_\alpha|F'_\alpha) > 0.5 \\ S &= (N(F_\alpha|F'_\alpha) + 0.5) \times P(F_\alpha|F'_\alpha) \textit{ otherwise} \end{aligned} \tag{3.3}$$

$$P(F_\alpha|F'_\alpha) = \text{max}(\text{min}(\mu_{F_\alpha}(u), \mu_{F'_\alpha}(u))), \forall u \in U \tag{3.4}$$

$$N(F_\alpha|F'_\alpha) = 1 - P(\overline{F}_\alpha|F'_\alpha) \tag{3.5}$$

Finally in the case of the *FUZZY_FUZZY Simple Rule*, the certainty factor of the consequent is calculated using the same formula as the CRISP Simple RULE. Currently, this type of combined reasoning for complex rules that involve multiple clauses in their antecedent is not supported.

3.3 Conclusion

In this chapter, an ontology that models the ambiguity of smart environments and a flexible semantic context management solution have been described. The proposed context management solution is able to process the uncertainty and vagueness of the contextual information. A data fusion mechanism applied in the case that multiple data sources for the same measurement exist in one room has been described. The inclusion of these three elements (uncertainty, vagueness and data fusion) in context management systems provides several advantages:

1. Taking into account uncertainty and vagueness of the contextual information provides a more detailed picture of the current state of the user environment. This leads to more informed decisions of the Smart Environments.

2. Uncertainty allows to ascertain the quality of the data, discarding those measurements below a certain threshold. The result of the more robust data model is the improvement of the system reliability.

3. Vagueness enables to model user perceptions. This allows to react better to their requirements.

4. The necessity of a data fusion process in context managing system can be clearly seem in those scenarios where the environment contains a large number of sensors. To tackle this problem, a unified picture of the environment to avoid conflicting behaviors is provided.

These advantages are described in more detail in Chapter 5. The solution devised has also some constraints. The most important one is the extra knowledge required from the Intelligent Environment deployers to model both aspects of the ambiguity (uncertainty and vagueness) when designing the domain specific applications. A broader knowledge of the domain is necessary in order to correctly create the rules according to the different certainty factors. Intelligent Environment deployers are also required to model the fuzzy membership functions for the different measurements, but this same problem arises when a traditional fuzzy inference engine is used. Finally, it is necessary to ascertain the certainty factor of each sensor

to be able to provide the certainty of the taken measures. If data about the sensor capabilities is not provided by the vendor's datasheet, Intelligent Environment deployers would have to perform a testing/calibration phase prior to the system deployment to determine these capabilities. Despite of these drawbacks, the proposed context management solution will provide Intelligent Environment deployers with a more truthful view of the situation of the modeled smart environment. The extra work taken when modeling the environment will lead to more informed decisions. This will help to avoid the unwanted behavior.

As discussed in the related work chapter, the solution proposed differs from previous ambiguity modeling approaches in that it combines both aspects of the ambiguity in one solution which also provides a semantic model of the environment. As a result, this solution is able to model a broader range of situations and interactions, providing richer context information. It also differs from mediation-based approaches like (Dey and Mankoff, 2005), since the solution set forward does not require any feedback from the user to identify and ascertain the uncertainty. Non-expert users will find difficult to evaluate the certainty of the raw data of the sensors. Instead of using a feedback process to appraise the certainty factor of more abstract context information, a data fusion process that calculates the combined uncertainty of the measurements of a specific location has been implemented together with a reasoning process that infers the certainty factor of the data created from the combination of raw measurements. In any case, it would be interesting to add some kind of feedback from Intelligent Environment deployers or an automatic method to assess the validity of the calculated certainty factor in order to allow the system to learn and infer the correct context data.

I resolved to stop accumulating and begin the infinitely more serious and difficult task of wise distribution.

Andrew Carnegie

CHAPTER

4

Distributed reasoning for context management

Smart environments need to manage context information(Dey, 2001) in order to react to the changes in it. Managing correctly context data allows applications to adapt themselves better to it, providing an enhanced experience to the users. Usually ontologies are regarded as one of the best approaches(Strang and Linnhoff-Popien, 2004) to model context information. OWL(owl, a) ontologies offer a rich expression framework to represent context data and their relations. In the preliminary work of this dissertation developing a context management middleware(Lopez-de Ipina et al., 2008) significant problems when using semantic reasoning engines in real smart environments were identified. As the number of triples in the ontology increases, the inference time for environment actions becomes unsustainable. In order to be able to deploy these systems in real environments they must be able to react to context changes is a reasonable time for the reaction to be considered appropriate. On the other hand, modern smart environments host a diverse ecosystem of computationally enabled devices. With the wide adoption of low-cost platforms like Arduino, Teensy, Raspberry Pi and STM32 Discover and the ubiquity of the smartphones, Smart Environments have more pervasive computing capabilities at their disposal nowadays than ever before. The existence of a large number

of devices available in the environment offers several advantages to help context management systems overcome the problem of high inference time in semantic reasoning.

The approach followed by this work has been to split the inference task among different reasoning engines or context consumers according to the interests stated by each of them. The inference is no longer performed by a central reasoning engine, but divided into a peer-to-peer network of context producers and context consumers. Dividing the inference problem into smaller sub-units makes it more computationally affordable. Context consumers only receive the information they are interested in and do not have to process non-relevant data. Another benefit is that the different parts of the reasoning process can take place in parallel, reducing the overall global time taken to resolve the inference problem. This parallelization allows to achieve a more feasible response time in the reactions to environmental changes. In order to create this distributed architecture, a multi-agent solution has been devised where each agent represents a context provider (sensor data, inferred facts from the reasoning engines...) or a context consumer (reasoning engines, actuators that react to environmental changes...). Each of these agents has a set of interests, represented by the context data type, a location and the certainty factor of the data. Taking into account this information, agents organize themselves using the Contract Network Protocol into an architecture that allows them to satisfy their interests.

4.1 Inference Sharing Process

The solution proposed relies on splitting the reasoning process (see Figure 4.1) into smaller inference units in order to achieve the following results:

1. Attain the *temporal decoupling* of the different inference units. This will allow the inference to be done concurrently in various reasoning engines. The parallelization of the inference process reduces the overall time required to reach certain conclusions.

2. Attain the *spatial decoupling* of the inference process. This will increase the general robustness of the system making it more fault-tolerant. Problems in

one reasoning engine will not stop all the inference, as opposed to a centralized approach

3. *Reduce the number of triples* and rules that each reasoning engine has to manage. This allows the use of more computationally constrained devices to carry out the inference process.

4. *Compartmentalize the information* according to the different interests of the reasoning engines. This will reduce the amount of data that is shared and exchanged over the network.

5. Allow the *dynamic modification of the created organization*. Devices in modern smart environments can change their location frequently (e.g., mobile phones, motes...), disappear (e.g., they get broken) or appear. The created hierarchy must change to adapt itself to these modifications.

This organization also allows the creation of a reasoning engine hierarchy that provides different abstraction levels in the inferred facts. As can be seen in Figure 4.1, the reasoning engines at the bottom use pretty simple concepts. But as the reasoning ascends in the hierarchy, the concepts become more complex and abstract as a result of the aggregation of the different inferred facts.

In order to decide how the reasoning should be divided among the reasoning engines, three factors have been taken into account:

1. *The ontological concepts* that will be used by the reasoner. These concepts are organized in a taxonomy (see Figure 4.2) depicting the class relations of the ontology. The AMBI2ONT(Almeida and López-de Ipiña, 2012) ontology discussed in the previous chapter is used. When trying to find an appropriate context provider, the context consumer can search for specific concepts (in the example *"Luminosity"*) or broader ones (as *"ElectricalConsumption"* that encompass *"LightStatus"*, *"AirConditioning"* and *"ApplianceStatus"*).

2. *The location* where the measures originate from. As with the ontological concepts, a taxonomy of the locations extracted from the ontology is used. This taxonomy models the *"contains"* relations between the different rooms,

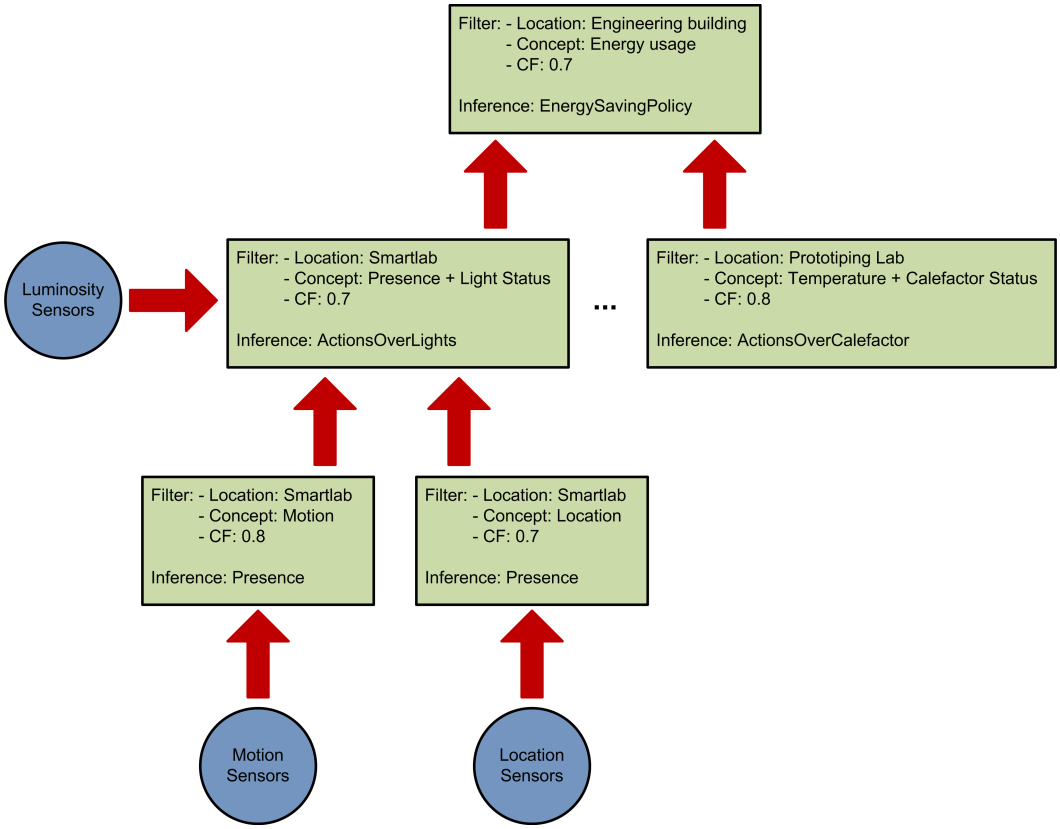

Figure 4.1: Reasoning hierarchy depicting the sensors (circles) and reasoning engines (squares) that take part in the inference process

floors and buildings (see Figure 4.3). The context consumer can search for an specific location (the Smartlab laboratory in the example) or for a set of related locations (for example all the rooms in the first floor of the engineering building)

3. *The certainty factor (CF)* associated to each context information piece. As was discussed in the previous chapter and in (Almeida and López-de Ipiña, 2012), modeling real environments taking certainty for granted is usually a luxury that a context management framework cannot afford. Reality, and so the context, is ambiguous. Sensors and devices are not perfect and their measures carry a degree of uncertainty, e.g., several thermometers in the same room can provide conflicting measures of the temperature. For this reason, the context consumer can specify a minimum CF in its searches. This allows context consumers to set a minimum threshold of quality for the context information to receive.

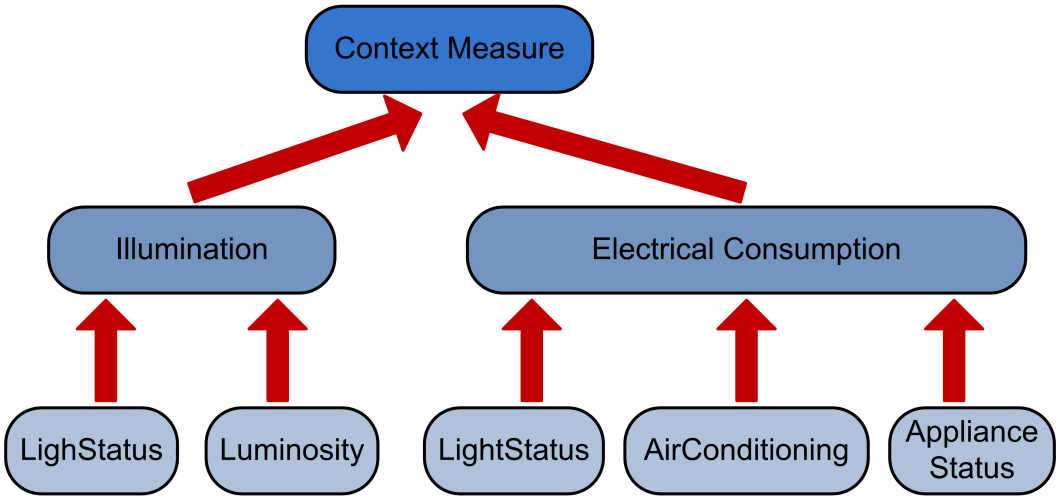

Figure 4.2: Part of the concept taxonomy extracted from the AMBI2ONT ontology.

An example of the usage of these three factors to divide the reasoning process can be seen in Figure 4.1. The different context consumers state their requirements (squares in the figure) and a reasoning hierarchy is created according to them.

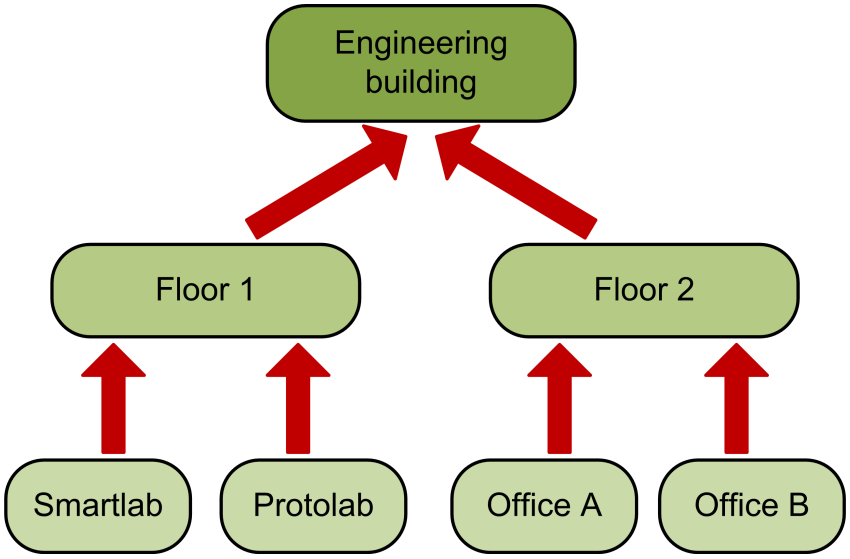

Figure 4.3: Part of the location taxonomy used on the system . The taxonomy depicts the *contains* relations of the used ontology.

4.2 System Architecture

To create this reasoning hierarchy, an agent-based architecture (see Figure 4.4) with two types of agents has been used:

1. *Context Providers:* These agents represent those elements in the architecture that can provide any context data. Each context provider has one context type (luminosity, temperature...), one location and a certainty factor. If a device can provide more than one context type, then it will be represented by one independent agent for each context type. Context Providers can be sensors, devices or inference engines that provide the inferred facts to other Context Consumers.

2. *Context Consumers:* These agents represent those elements that need to consume context data. Each context consumer has a set of interests defined by the context types, locations and minimum certainty factors. Context Consumers can be inference engines, devices or actuators that react to the changes in the environment.

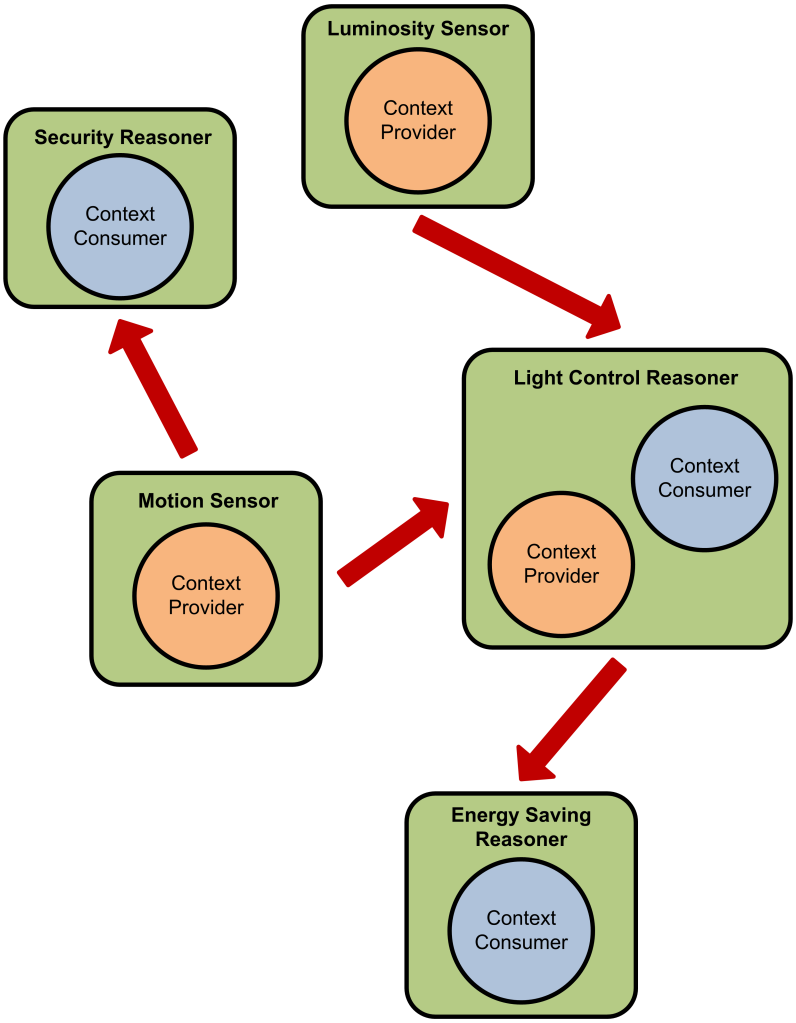

Figure 4.4: Example of the dataflow of the agents in the architecture.

The differentiation between providers and consumers is simply logical. The same device can act as a provider and a consumer at the same time, receiving data from sensors and supplying the inferred facts to another device. While the resulting inference process follows somewhat a hierarchy, the agent structure is purely a peer-to-peer architecture where there is no central element. Agents negotiate among them to find which other agents who can meet their needs.

The architecture has been implemented using the JADE framework(jad). JADE (Java Agent DEvelopment Framework) is a software framework implemented in Java. It simplifies the implementation of multi-agent systems through a middleware that complies with the FIPA[1] (Foundation for Intelligent Physical Agents) specifications. JADE provides several tools that facilitate the development of multi-agent systems:

- A runtime environment where agents reside.
- A library that allow developers to create their agents.
- Tools to administer and control the deployed agents.

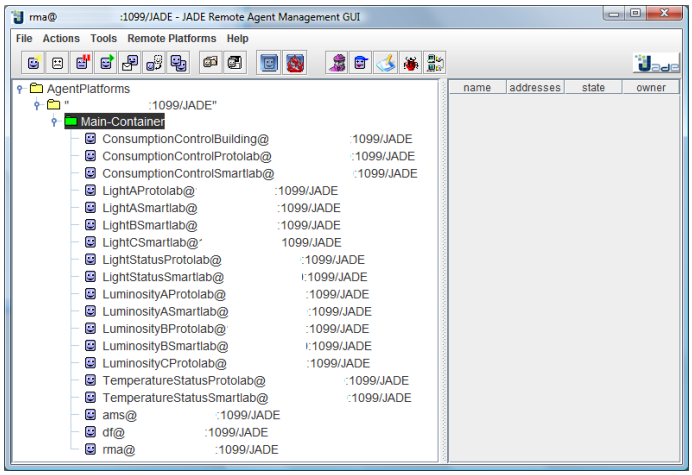

Figure 4.5: JADE Agent Management GUI with several Context Consumer Agent and Context Provider Agents already launched. Being the Main Container the AMS (Agent Management System) and DF (Directory Facilitator) agents are also present.

[1] http://www.fipa.org/

A JADE application is composed of multiple Agents (see Figure 4.5), each one having its own unique identifier. Each agent executes different tasks and communicates with other agents exchanging messages. Agents exist on top of a Platform, which provides to the agents a series of services (like the message delivering one). A platform is composed by Containers. Each running instance of the JADE runtime environment is called a container as it can contain several agents. Containers can be run in different machines, achieving so the distributed platform. There is always one Main Container in each platform and all other containers register in it. The main container differs from other containers in that it must be the first container to start in the platform and that it includes two unique agents:

- *AMS (Agent Management System):* it provides the naming service and is the only agent able to manage the platform (start, kill, shutdown agents, create remote containers...). Other agents must send requests to the AMS to perform these actions.

- *DF (Directory Facilitator):* it provides a Yellow Pages service where agents can announce the services they provide or find the services provided by other agents.

A JADE agent extends the *jade.core.Agent* class (see Figure 4.7) and implements a set of behaviors. Every JADE agent has the following characteristics:

- It has a unique name in the execution environment.

- It runs on a single thread (*single-threaded*).

- It has two methods that need to be overwritten:

  - It has a *setup()* method that is called at the beginning of the agent execution. This method is used to initialize the agent, stating which ontology is using and setting its behaviors.

  - It has a *takedown()* method that is called within a *doDelete()* call from the AMS. It is used to free the resources used by the agent.

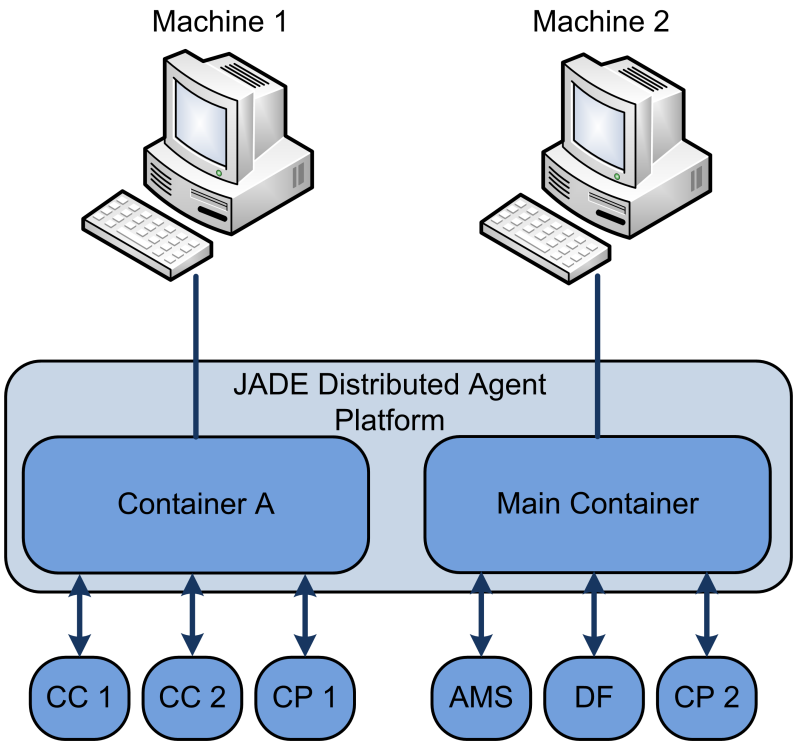

Figure 4.6: An example of JADE deployment. The Main Container has the AMS (Agent Management System) agent and DF (Directory Facilitator) agent. In both containers there are several CC (Context Consumer) and CP (Context Provider) agents. Both containers belong to the same platform instance.

```
import jade.core.Agent;

public class DummyAgent extends Agent {
    @Override
    protected void setup(){
        System.out.println("It's alive, it's moving!");
    }

    @Override
    protected void takeDown() {
        System.out.println("Bye, bye");
    }
}
```

Figure 4.7: Example of a simple agent.

- In the agent implementation several behaviors related to the agent are defined. These behaviors describe the functionality of the agent (how the agents send and receives messages, its actions...). There are different behavior classes depending on the nature of each activity (one time behaviors, conditional behaviors, periodic behaviors and so on)

One JADE agent can have several states:

- *INITIATED:* The agent has been created but it has not yet been registered in the AMS, assigned a unique name or an address. The agent cannot communicate with other agents.
- *ACTIVE:* The agent has already been registered in the AMS and has been assigned a unique name and an address. It can communicate with other agents.
- *SUSPENDED:* The thread of the agent has been stopped. The agent cannot execute any behavior.
- *WAITING:* The agent is blocked waiting for something. Its thread is suspended and will wake up when a certain condition is met.

- *TRANSIT:* A mobile agent is in this state when is moving to another location. The system will continue saving the messages for the agent until it is again active.

- *DELETED:* The agent has been eliminated. Its execution thread has ended and its entry in the AMS has been deleted.

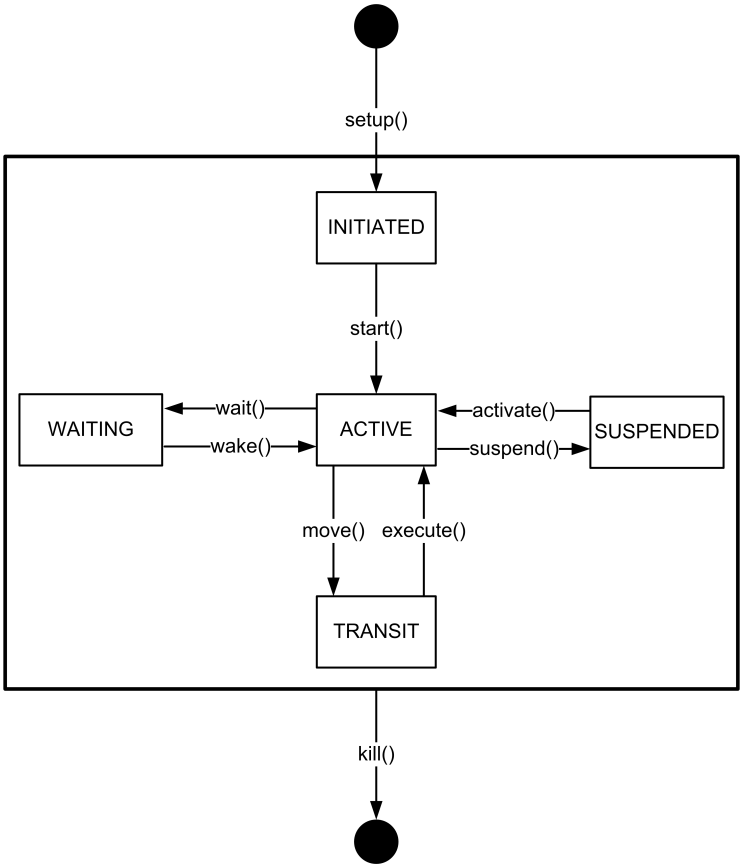

Figure 4.8: Transitions between the agent's states.

Communication in JADE is based in the asynchronous message passing paradigm. Agents can communicate between themselves no matter the container or platform they reside in. Each agent has a message queue where the JADE runtime posts the messages sent by other agents. The main components of a message are:

- The sender of the message.

- The receivers of the message.
- The communicative intention. It is called the *performative* and indicates what the intention of the sender is. Examples of performatives are: REQUEST, INFORM, ACCEPT_PROPOSAL, QUERY_IF...
- The contents with the information of the message.

4.3 The Negotiation Process

The negotiation meets the FIPA ACL specification(fip, 2002) for agent communication, following the Contract Net Protocol(Smith, 1980). The Contract Net Protocol is used for distributed problem solving, finding the cooperative solution to problems using a decentralized and loosely coupled approach. This approach meets the objectives of the proposed system. Usually, this protocol is used when each individual entity of the architecture does not have all the necessary information to solve the problem. This is not the case in the majority of the traditional context management systems (see Chapter 2). The context information is centralized in a single point to process it. In this work, the need to process separately different parts of the problem is not a limitation, it is indeed the sought behavior. The negotiation process using Contract Net Protocol has four main characteristics(Smith, 1980):

1. It is done locally and does not involve any centralized element.
2. It is a two-directional exchange of information.
3. Each participant evaluates that information from its own perspective.
4. The final agreement is reached by mutual selection.

To be able to take part in the negotiation process, all participants have to have information about the three factors previously described. Without these data, the negotiation process cannot happen. The negotiation follows these steps:

1. The Context Consumer Agent (CCA) sends a Call For Proposals (CFP) stating the context type and location that it is interested in and the minimum certainty factor (CF) expected from the context.

2. The Context Provider Agents (CPA) replies to the CFP with individual proposals stating what they can offer to the CCA.

3. The CCA checks the received proposals. Some of the context consumers have a maximum number of context sources they can subscribe to concurrently. This is dictated by the computational capabilities of the device running the CCA. If the number of received proposals is above this limit the CCA selects those that it considers the best (comparing their CF).

4. The CCA sends an Accept to the selected CCPs and subscribes to the context updates.

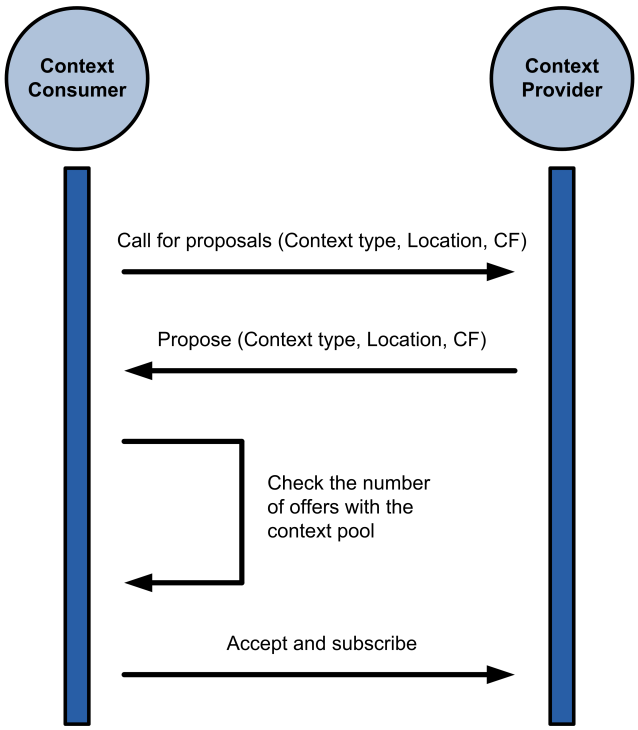

Figure 4.9: Negotiation between a context provider and a context consumer.

4.4 Conclusions

In this chapter, the problems of semantic inference in smart environments has been discussed. On one hand, ontologies are one of the best approaches to context mod-

eling. On the other hand, semantic reasoning can be slow and cumbersome. In order to tackle this problem in smart environments consisting of a rich ecosystem of computationally enabled objects and devices, to divide the inference process into smaller units and distribute it among several semantic inference engines has been proposed.

Distributing the inference process brings about several advantages. Firstly, *temporal and spatial decoupling* of the inference sub-units is obtained. This is done splitting the reasoning problem into different agents of the network. Thanks to the *spatial decoupling*, if one node loses network connectivity or suffers a hardware problem the others can try and partially solve the reasoning problem without the output of that inference sub-unit. Depending on the problem domain, the reasoning engine network will be able to attain a partial solution instead of no solution at all. The *temporal decoupling* allows parallelizing the inference, reducing the total time needed to achieve a solution (see the next chapter for the results of the validation of the multi-agent system in several different scenarios). Secondly, the *number of triples* that each reasoning engine has to process is reduced by sharing then among the different agents in the network. As will be shown in the next chapter, the semantic inference time increases exponentially as the data to be processed does. Splitting the data between different agents helps, along with the temporal decoupling, to reduce the total inference time. Thirdly, the *information is compartmentalized* in different reasoning engines depending on the interests of each one. Currently, the proposed system does not have any security mechanism that takes advantage of this characteristic to enforce privacy settings on context data, but is an interesting feature to implement in the future. Finally, due to the negotiation mechanism, the *reasoning engine architecture can reorganize itself* if there is any change in the reasoning engines interests or the nature of the context providers. This allows the proposed system to adapt to the changes in the nature of the environment.

Fear cannot be banished, but it can be calm and without panic; it can be mitigated by reason and evaluation.

Vannevar Bush

CHAPTER

5

Evaluation

In the previous chapters, the two main contributions of this dissertation have been presented:

- A reasoning engine for context management that combines semantic reasoning, vagueness (as fuzzy sets) and uncertainty (as a certainty factor) in Chapter 3. This reasoning engine also includes data fusion techniques to offer a complete picture of the context information.

- An architecture to split the inference problem among several reasoning engines in Chapter 4. This architecture allows for more efficient reasoning over context information.

In order to evaluate the first contribution, a qualitative comparison of context management with and without vagueness, uncertainty and data fusion has been performed. With this qualitative evaluation, it will be proven how the inclusion of these three elements will allow to better model the context in Intelligent Environments. How this additional information leads to better and more informed decisions will also be demonstrated.

On the other hand, a quantitative evaluation of the second contribution, proving how splitting the inference process leads to a more efficient and fast reasoning over

the context information, will also be performed. To do this, the time that takes to resolve the global inference problem in a centralized approach and in several distributed scenarios will be compared. As a result of the evaluation, the factors that influence the results will be identified, and it will be discussed how to adapt the architecture to achieve the maximum efficiency in each scenario.

5.1 Inference Over the Uncertainty and Vagueness of Context Information

In order to illustrate the importance of modeling ambiguity and having a data fusion process in context management systems, several smart environment scenarios will be described. Each scenario will show different aspects closely related to our context manager: modeling the certainty of the captured measures and the rules, modeling the subjectivity in the perceptions of the users and the importance of creating a global picture for each space.

5.1.1 Scenario 1: Temperature Control in a Laboratory. Data uncertainty and data fusion

In this first scenario, the importance of evaluating the certainty factor of the context data and the importance of the data fusion process will be explained. In this scenario, the temperature of a research laboratory wants to be maintained at 23 degrees. Four thermometers (see Table 5.1) located inside the lab will be used to measure the temperature. The laboratory has a radiator to regulate the temperature. To simplify the example, the only operation allowed is to switch it on and off. The low certainty factor in *"Thermometer 4"* is due to the fact that this thermometer is broken. For this example, each thermometer has a self-diagnostic mechanism that constantly evaluates its status. The self-diagnostic details fall outside the scope of this example.

The scenario will have three versions (see Table 5.2). The first one will use a semantic model without uncertainty and vagueness information and without any data fusion process. The second one will include a rather simple average value calculation as the data fusion mechanism. Finally, the third one will be the process

Device	Type	CF	Value
Thermometer 1	Sensor	0.8	25
Thermometer 2	Sensor	0.95	23
Thermometer 3	Sensor	0.9	22
Thermometer 4	Sensor	0.4	11
Radiator	Actuator	NA	NA

Table 5.1: Devices present in scenario 1.

	Uncertainty	Vagueness	Data fusion
Version 1	No	No	No
Version 2	No	No	Yes
Version 3	Yes	Yes	Yes

Table 5.2: Configuration of each version of the scenarios.

previously described. In each scenario, the sensors and actuators that take part, their values and CF and the rules that model the behavior of the system will be described. To make the examples easier to understand, the rules will be written in pseudocode using simple triple-like expressions.

In the first version, with no ambiguity modeling or data fusion (see Table 5.2), the rules would those shown in Figure 5.1 and Figure 5.2.

There are several problems with this approach. First, the system is not able to take into account the low reliability of the thermometer T4, using its temperature measure as input for the reasoning engine. This will result in incorrect behavior when that measure is processed, prompting the system to switch on the radiator to increase the room temperature. Secondly, each temperature measurement is processed individually, resulting in unreliable results that depend on the order of this processing. In this version, the context manager has no means of knowing the global measure for the laboratory and must process each measure individually, resulting in contradictory actions in this case.

In the second version (see Table 5.2) a simple data fusion mechanism will be

```
thermoX type thermometer
thermoX location LaboratoryA
thermoX hasValue temperatureX
temperatureX < 21
->
radiatorX type radiator
radiatorX location LaboratoryA
radiatorX status ON
```

Figure 5.1: Switching on the radiator.

```
thermoX type thermometer
thermoX location LaboratoryA
thermoX hasValue temperatureX
temperatureX < 21
->
radiatorX type radiator
radiatorX location LaboratoryA
radiatorX status ON
```

Figure 5.2: Switching the radiator off.

```
LaboratoryA hasGlobalTemperature temperatureX
temperatureX < 21
->
radiatorX type radiator
radiatorX location LaboratoryA
radiatorX status ON
```

Figure 5.3: Switching on the radiator with a global measure.

```
LaboratoryA hasGlobalTemperature temperatureX
temperatureX > 25
->
radiatorX type radiator
radiatorX location LaboratoryA
radiatorX status OFF
```

Figure 5.4: Switching the radiator off with a global measure.

used. This example will help us illustrate the need of a sound and adaptable data fusion mechanism. The rules that control the behavior will be those shown in Figure 5.3 and Figure 5.4.

In this case, the system has a unified vision of the temperature in the room and is able to take a unique action to adjust the radiator, but still does not take into account the differences in the certainty factors of the sensors. As a result the global temperature in the laboratory is 20.25 °C and the radiator will be turned on.

Finally, in the third version of the scenario the context manager described in Chapter 3 is used, i.e. processing the ambiguity of the context data and applying a richer data fusion process. In this case, the combination strategy with a cautious behavior for the data fusion process (see Chapter 3) is used. The rules are almost identical to the second version, but include an important change—now they state a minimum CF for body of the rule to be launched. The rules that control the behavior will be those shown in Figure 5.5 and Figure 5.6.

According to the selected configuration the global temperature value will be 22.3 (see Equation 5.1) and the global certainty factor will be 0.775 (see Equa-

```
LaboratoryA hasGlobalTemperature temperatureX
temperatureX < 21 WITH CF 0.8
->
radiatorX type radiator
radiatorX location LaboratoryA
radiatorX status ON
```

Figure 5.5: Switching on the radiator with uncertainty.

```
LaboratoryA hasGlobalTemperature temperatureX
temperatureX > 25
->
radiatorX type radiator
radiatorX location LaboratoryA
radiatorX status OFF
```

Figure 5.6: Switching the radiator off with uncertainty.

tion5.2).

$$m_g = \frac{25 \times .85 + 23 \times 0.95 + 24 \times 0.9 + 11 \times 0.4}{0.85 + 0.95 + 0.9 + 0.4} = 22.3 \tag{5.1}$$

$$CF_g = \frac{0.85 + 0.95 + 0.9 + 0.4}{4} = 0.775 \tag{5.2}$$

The certainty factor of the global measure will be too low to launch the rule (we set a minimum CF of 0.8 for the temperature in the rules), but we can tune even more the configuration and establish a CF threshold for the measures in the data fusion process. As was explained in Chapter 3, using the threshold the measurements with a CF below it are automatically discarded. Setting the threshold for the data fusion problem at 0.8 the measurements given by the broken thermometer are discarded, obtaining the results shown in Equation 5.3 and Equation 5.4.

$$m_g = \frac{25 \times 0.85 + 23 \times 0.95 + 24 \times 0.9}{0.85 + 0.95 + 0.95} = 23.96 \tag{5.3}$$

	Version 1	**Version 2**	**Version 3**
Result	The radiator controlling rules are fired two times (only two of the four values meet the conditions). The final state of the radiator will depend on the order in which the rules are evaluated	The computed global measure will be 20.25 °C. The radiator will be turned on.	The computed global measure will be 23.96 °C. No action will be taken.

Table 5.3: Results of each version of the first scenario.

$$CF_g = \frac{0.85 + 0.95 + 0.9}{3} = 0.9 \tag{5.4}$$

Taking into account the certainty factor of the context data and applying a data fusion process the context manager can ascertain much more reliably the real state of the laboratory, providing a better picture of its current situation. As a result the context manager knows that the temperature value is between the acceptable limits and that no action must be taken. As can be seen on Table 5.3 the results between different versions vary significantly. The results in the first version are completely unreliable, as the final state depends on the order that the rules are evaluated (this is problematic because several semantic reasoning engines do not guarantee the order in which matching rules will fire(apa)). The second version uses the measurement from the broken thermometer to compute the global measure, inducing an error in the calculation that results in the radiator been switched on. Finally, the last version takes into account the certainty factor, resulting in a more reliable system.

5.1.2 Scenario 2: Occupation Data Inferred from the Status of the Lights. Rule uncertainty.

This scenario will illustrate the necessity of modeling another aspect of the uncertainty. In the previous scenario, the importance of assessing the uncertainty in the taken measurements was explained; this one will be centered in the uncertainty of

```
lightX type light,
lightX location LaboratoryA
lightX hasValue On
->
LaboratoryA hasStatus peopleInside
```

Figure 5.7: Assessing the presence of people on a room.

```
RULE CF 0.8
lightX type light,
lightX location LaboratoryA
lightX hasValue On
->
LaboratoryA hasStatus peopleInside
```

Figure 5.8: Assessing the presence of people on a room using uncertainty.

the rule. In this scenario, the room occupation data wants to be inferred based on the status of the lights. We will have two versions of this scenario with and without uncertainty.

The first version is quite straightforward, without taking into account the outcome uncertainty. The rules that model the behavior are show in Figure 5.7.

In this case, if the lights of the laboratory are ON, it is assumed that there is someone inside. But with this behavior, situations where the users get out leaving the lights on are not taken into account. Using the rule uncertainty these situations could be modeled. Assuming that after studying the habits of the users, it can be inferred that 80% of the time, when a light is on there is someone in the laboratory, then the resulting rule is the one shown in Figure 5.8.

Using the rule uncertainty we can model better the contextual information (in this case the information obtained after processing the raw data from the measurements). The result is a more reliable model of the smart environment that ascertains better its real status. With the rule uncertainty, the system can model the fact that the lights been on does not automatically imply that someone is inside the room. With this knowledge the system can show a more complex behavior, taking the

```
RULE CF 1
LaboratoryA hasGlobalTemperature temperatureX
temperatureX HOT
->
airCoditioningX type AirConditioning
airCoditioningX location LaboratoryA
airCoditioningX temperature 22
```

Figure 5.9: Controling the room temperature.

certainty factor into account when deciding the appropriate response.

5.1.3 Scenario 3: User Preferences for Temperature. Vagueness and uncertainty.

This final scenario will explain the importance of taking into account the human factor when modeling the context data. In this case, vagueness is used to model the user perceptions and it is represented using fuzzy sets. This scenario will be similar to the first one, but the temperature limits will be expressed using fuzzy facts. To show how the uncertainty and the vagueness interact, a FUZZY-CRISP Simple Rule with fuzzy uncertain input facts and a crisp output will be used:

- *Input data:* Fuzzified room temperature with an associated certainty factor (0.95 in this example). We assume that the fuzzification process is done accordingly to the previously captured user perceptions.
- *Output data:* Crisp temperature value for the air conditioning system with a certainty factor determined using the Equation 3.3 presented in Chapter 3.
- *Rule:* FUZZY-CRISP simple rule with an associated certainty factor. The used rule is shown in Figure 5.9

The first step will be to calculate the value of $P(\overline{F}_\alpha|F'_\alpha)$ (see Figure 5.10 for a graphical explanation) and with it the measure of necessity (N, see Equation 3.5 in Chapter 3) as shown in Equation 5.5.

$$N(F_\alpha|F'_\alpha) = 1 - 0.6 = 0.4 \tag{5.5}$$

N is lower than 0.5, so the similarity (S) measure will be:

$$S = (N(F_\alpha|F'_\alpha) + 0.5) \times P(F_\alpha|F'_\alpha) = (0.4 + 0.5) \times 0.9 = 0.81) \tag{5.6}$$

With the similarity we can calculate the certainty factor of the outcome following formula:

$$CF_c = CF_r \times CF_f \times S = 1 \times 0.95 \times 0.81 = 0.76 \tag{5.7}$$

Using fuzzy sets, the vagueness present in the context data can be modeled. This allows to model user perceptions and preferences (e.g., *"I like the water hot"*, *"Turn on the air conditioning system when the room is hot"*) without the loss of expressiveness obtained when using only crisp values.

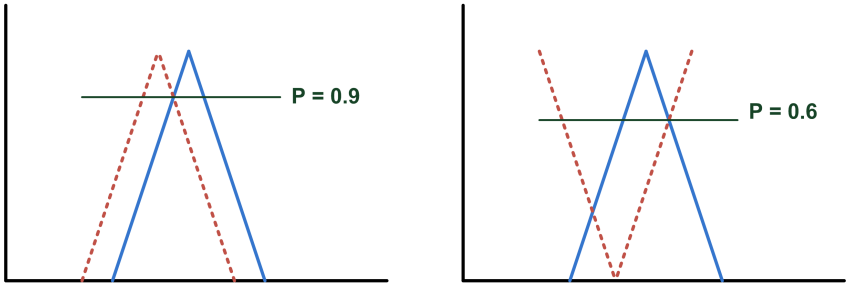

Figure 5.10: Possibility calculation. The straight line represents the expected fuzzy fact and the dotted line the provided fuzzy fact.

5.1.4 Discussion

As this scenarios show, the inclusion of uncertainty, vagueness and a data fusion process in context management systems provides several advantages:

- Taking into account uncertainty and vagueness of the contextual information provides a more detailed picture of the current state of the user environment. This leads to more informed decisions of the Smart Environments.

- Uncertainty allows us to ascertain the quality of the data, discarding those measurements below a certain threshold. The result of the more robust data model is the improvement of the system reliability.
- Vagueness allows us to model user perceptions. This allows us to react better to their requirements.
- The necessity of a data fusion process in context managing system can be clearly seem in those scenarios where the environment contains a large number of sensors. To tackle this problem, a unified picture of the environment to avoid conflicting behaviors is provided.

The created system has also some constraints. The most important one is the extra knowledge required from the system-deployers to model both aspects of the ambiguity (uncertainty and vagueness) when designing the domain specific applications. A broader knowledge of the domain is necessary in order to correctly create the rules according to the different certainty factors. System-deployers are also required to model the fuzzy membership functions for the different measurements, but this same problem arises when a traditional fuzzy inference engine is used. Finally is necessary to ascertain the certainty factor of each sensor to be able to provide the certainty of the taken measures. If data about the sensor capabilities is not provided by the vendor, system-deployers would have to perform a calibration phase prior to the system deployment to determine these capabilities. Despite of these drawbacks the proposed context managing system will provide system-deployers with a more truthful view of the situation of the modeled smart environment. The extra work taken when modeling the environment will lead to more informed decisions. This will help to avoid the unwanted behavior discussed in the previous section, like performing actions using unreliable context information given by broken sensors or processing multiple measures of the same location. The result will be a more reliable context management process.

5.2 Multi-Agent Inference Engine Ecosystem

To evaluate the proposed solution in the area of distributed reasoning four scenarios have been created. In the first one a centralized inference engine processes all the

data from the context providers (see Figure 5.11).

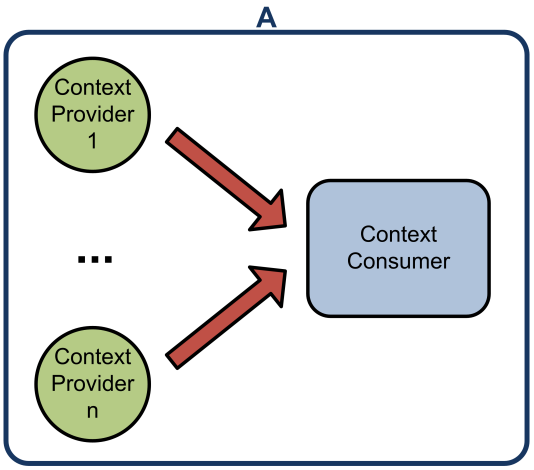

Figure 5.11: Scenario A, a centralized approach where a single inference engine processes all the data from the context providers

In the second one a distributed architecture of inference engines is used (see Figure 5.12). This scenario has three inference engines (Context Consumer A, B and C). Some of the inference has been parallelized (Context Consumer A and B), while part of it can only be performed after some of the data has been processed (Context Consumer C). The context data has been distributed between the three Context Consumers. Context Consumer A will process the 40% of the Context Providers, Context Consumer B will process the 50% and Context Consumer C will take the final 10% of the Context Providers and the inferred facts from Context Consumer A and B.

In the third scenario (see Figure 5.13) the inference process is done in three serialized Context Consumers. The context data has been distributed between the three of them. Context Consumer A takes the 40% of the data, Context Consumer B takes the inferred facts from A and another 10% of the context data and finally Context Consumer C will process B's inferred facts and the other 50% of the context data.

In the fourth and final scenario (see Figure 5.14) Context Consumers process the context data at the same time in a completely parallel architecture. The data is divided in the same way it was divided in the third scenario.

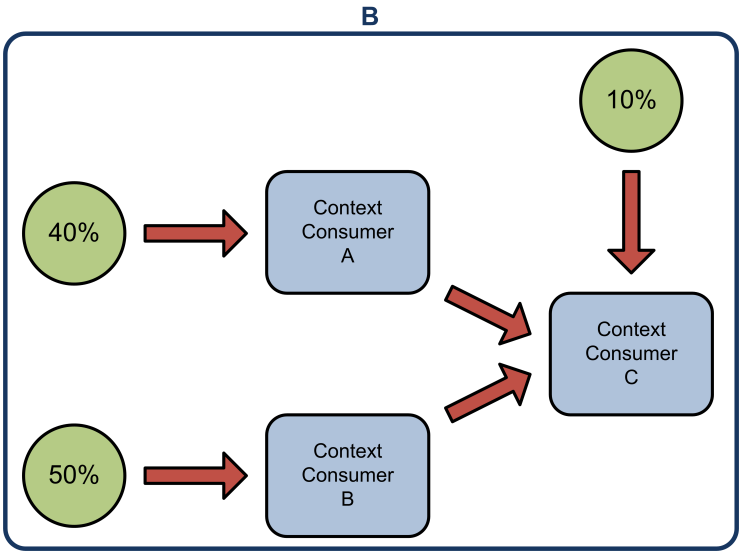

Figure 5.12: Scenario B, a distributed approach where three inference engines process the data from the context providers

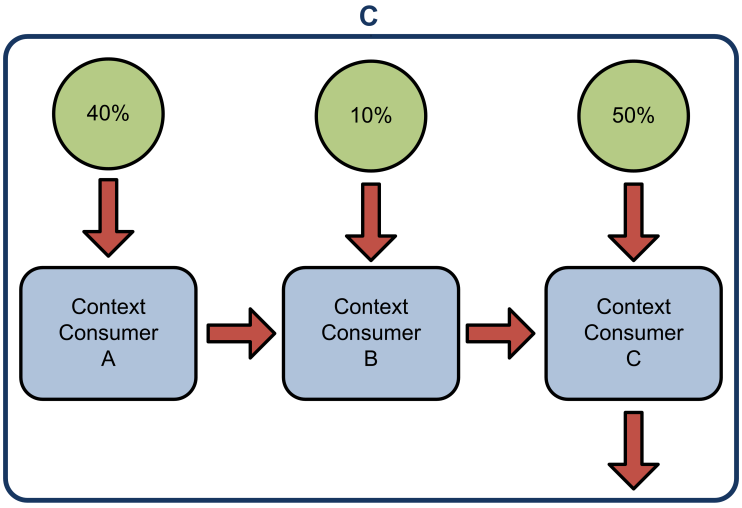

Figure 5.13: Scenario C, a completely serialized approach where the outcome of the previous reasoner is the input of the next one

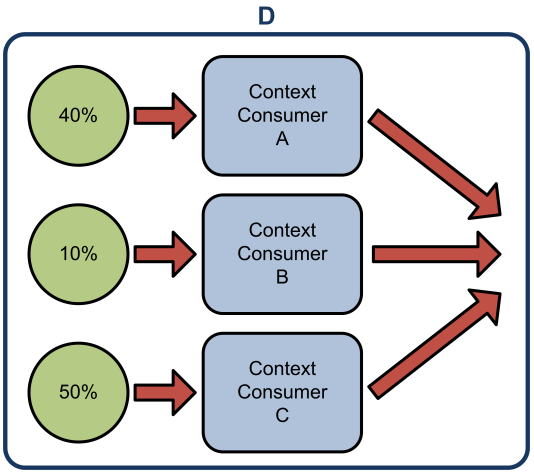

Figure 5.14: Scenario D, a completely parallelized inference process

A summary of the experiments can be found in Table 5.4:

- For each scenario we have performed a series of experiments using from 40 to 300 Context Providers.
- Each experiment has been repeated 100 times.
- In all scenarios the inference engines were run in an Intel Core Duo P8700 with 4 GB of memory. As was explained in Section 6, currently it is not possible to perform these tests in embedded devices.
- A standard well-known reasoning engine has been used to evaluate the multi-agent reasoning engine ecosystem instead of the reasoning engine described in Chapter 3. The reason is to avoid the "noise" introduced by an unknown reasoning engine and to isolate the multi-agent system from other factors. This will allow to better evaluate the performance of the system. The used inference engine was the one provided with the Jena Framework(apa).
- The average network latency during the experiments was of 50 ms. In the next subsection we analyze how latency changes influence the results.
- The simulations used do not take into account node mobility or connectivity ranges. For the current state of the system, it is assumed that nodes are

Scenario	Approach	# Of Context Providers	Network Latency (ms)	Context Consumer's Hardware	# Of Experiments Performed
A	Centralized	40, 60, 80, 100, 120, 160, 200, 250, 300	50	Intel Core Duo P8700 with 4 GB of memory	100
B	Distributed (mixed)				
C	Distributed (serialized)				
D	Distributed (parallel)				

Table 5.4: Configuration of the experiments

static and that they do not disappear due to mobility or connectivity range problems.

As can be seen in Figure 5.15 the centralized approach (Scenario A) inference time degrades drastically as the number of Context Providers (and hence the data to be processed) increases. As we discussed in the introduction the response time to context changed is an important factor in smart environments. Users need to see the reactions of all these changes to happen in a timely manner, otherwise the usability of the whole system will suffer.

In Figure 5.16, the differences between the centralized (Scenario A) and distributed (Scenario B, C and D) approaches are seen. For a small number of context providers the centralized approach is much more efficient. The time gained parallelizing the inference process is minimal and much more time is lost due to the network latency. But as the number of Context Providers increases the distributed approach becomes more efficient. Even with 250 Context Providers the inference time for the distributed approach is under 2 s, while the inference time of the cent-

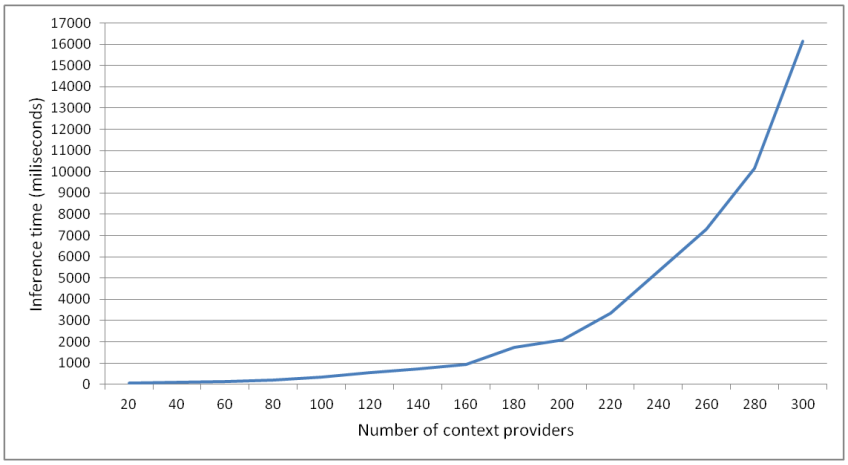

Figure 5.15: Inference times (in milliseconds) for a centralized approach where only one inference engine processes all the information from the Context Providers.

ralized approach is of 6,386 ms. This difference is even bigger with 300 Context Providers, where the centralized approach is five times slower.

The architecture of the distributed approaches will be dictated by the specific application domain. In order to evaluate the differences, three different distributed architectures have been tested. A completely serialized architecture (Scenario C) where the output of each Context Consumer is the input of the next Context Consumer, an architecture where context data is processed in a completely parallel manner (Scenario D) and a scenario that mixes both approaches (Scenario B). As expected, among the distributed scenarios, D achieves the fastest inference times and C the slowest ones, being B's results between both of them. It has to been taken into account that Scenarios C and D are extremes of the possible distributed architectures and will not be usually encountered in real life problems. It is interesting to note that even the slowest distributed scenario performs better that the centralized one once the number of the Context Providers increases. As can be seen in Figure 5.15 the inference time increases exponentially with the number of Context Providers. As is discussed in the next sub-section, even if the process is completely serialized, splitting the inference problem helps reducing the global inference time.

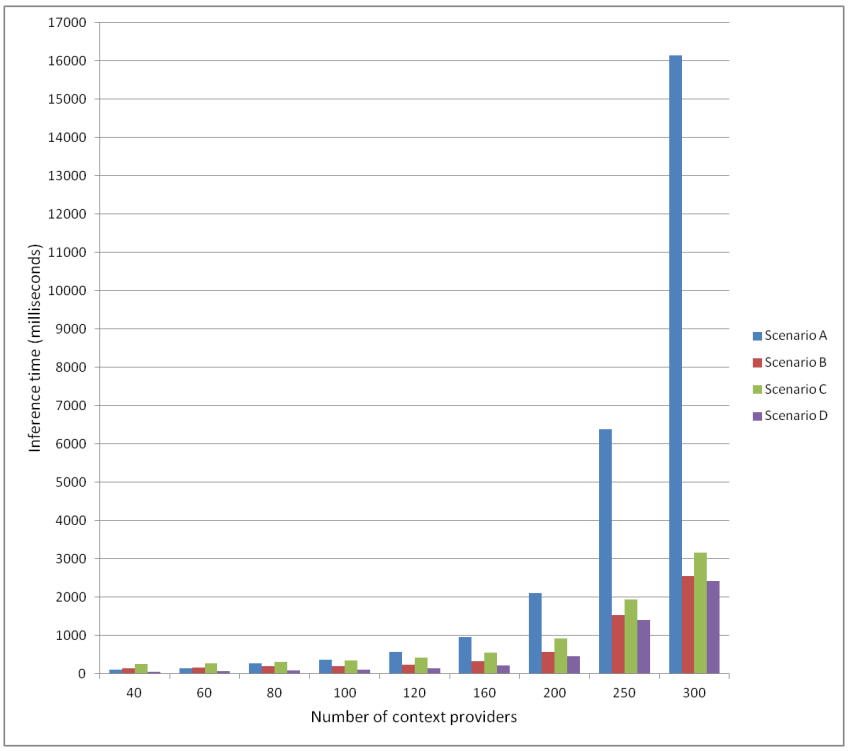

Figure 5.16: Comparison of inference times (in milliseconds) between the centralized and distributed approaches.

5.2.1 Discussion

As result of the evaluation of the system we have identified four factors that influence the global performance of the system:

1. *The hardware:* Obviously the computational capabilities of the hardware where the inference is carried on will determine how fast that reasoning is performed.

2. *The inference engine:* The used semantic inference engine will also result in a faster or slower reasoning process. The performance of different semantic reasoning engines was analyzed in (Lopez-de Ipina et al., 2008).

3. *The architecture of the reasoning engine ecosystem:* The final architecture of the distributed reasoning ecosystem will depend on the specific domain of the problem. The architecture will dictate how much of the reasoning process can be parallelized.

4. *The network latency:* In the case of the distributed reasoning process the network latency is an important factor to take into account. The latency time must be added to the total inference time.

Points one and two are common for any reasoning architecture (centralized or distributed). The advantages or drawbacks that arise from the design choices in those features will be shared by both approaches. Points three and four are more interesting, because it is in those design choices where one approach will differ from the other. Our analysis focuses in these last two factors. In the case of the architecture of the reasoning engine ecosystem, it is important to design it to parallelize as much inference as possible. The distribution of the inference can be done according to three aspects as explained in Chapter 3: the location where the data is originated on, the context type expressed by the data and the certainty factor associated to the data. We acknowledge that there are scenarios where the restrictions imposed by the domain makes impossible to parallelize the inference process (e.g., if there is only one type of sensors in a single location), but usually smart environments are a host of a diverse ecosystem of sensors and devices and encompass a space divided in a series of rooms (a house, a building, an office,

etc.). The next step is to analyze the *"inference flow"* and identify which sub-units of the inference problems can be parallelized and which ones depend on a previous inference, an example of this can be seen on Figure 4.1 in Chapter 4. Usually there are several inference sub-units in a problem than can be processed in parallel, reducing the total inference time for the whole system.

The second factor, network latency, penalizes the distribution of the inference problem. High network latency can make unfeasible to share the inference between multiple reasoning engines. It is easy to see how these two factors work against each another. On one hand, it is desired to have as many inference sub-units as possible distributed among several reasoning engines. On the other one having multiple reasoning engines that have to communicate with each other can be counterproductive depending on the existing network latency. It is important to reach equilibrium between them to maximize the achieved improvement. This improvement can be expressed as:

$$T_{gain} = T_{par} - T_{lat} \tag{5.8}$$

Where:

- T_{gain} is the total time gained distributing the inference.
- T_{par} is the time gained with the parallelization of the different inference sub-units and the reduction of triples for each reasoning engine.
- T_{lat} is the time lost in the communication between the reasoning engines due to the network latency.

It must be taken into account that other benefits arise also from the distribution of the reasoning process. The spatial decoupling assures that at least some of the inference will take place if some of the reasoning engines fail (due to connectivity problems, hardware failure...). This increases the overall robustness of the system, making it more reliable. In the end, in order to decide if a distributed or centralized approach is more appropriate for a specific problem domain, users much ask themselves the following questions:

1. What is the average network latency?

2. Can the inference be parallelized or is a completely serial process in this domain?

3. How much semantic data must be processed? Only the measures from a few sensors or the data produced by all the users and sensors of a smart building?

4. Is important the robustness of the system? Partial solutions of the inference problem are useful?

5.2.1.1 Current limitations

The system has several limitations in its current state:

- There is no mechanism to automatically assess the certainty factor of the context providers. This process must be done manually by the user deploying the system, testing each sensor type to evaluate the uncertainty in their measures.

- The computational capabilities of each device are not taken into account in the negotiation process. This can result in computationally constrained devices been burdened with a large fraction of the inference problem.

- Related with the previous point, the system does not allow to dynamically reassign the rules from one reasoning engine to another according to their capabilities.

Future work (as discussed in Chapter 6) will address these limitations of the system. Implementing these characteristics will lead to a more robust system that can adapt itself to the devices present in the environment for a more efficient reasoning process.

This is the end. My only friend, the end.

The Doors

CHAPTER

6

Conclusions

In this chapter the main results of this research are reviewed, highlighting the specific contributions and identifying further research areas. In Section 4.4 a global description of the research work is given. The achieved contributions are described in Section 6.2 and a complete list of the relevant publications is given in Section 6.3. New research areas and ideas are presented in Section 6.4 and the final remarks are given in Section 6.5.

6.1 Discussion

In this thesis, several contributions have been provided: an ontology that models the ambiguity of smart environments, a flexible semantic context management system that takes into account the ambiguity (which is composed by the uncertainty and the vagueness) and provides a complete picture of the environment and a multi-agent architecture that allows to split the inference problem among multiple reasoning engines.

The proposed context manager provides three main advantages:

1. It takes into account the *uncertainty and vagueness of the context information* as explained in Chapter 3. Using this information, the presented context

manager is able to better model the environment, leading to more informed decisions as proved in the previous chapter. Uncertainty allows the system to evaluate the quality of the context information, discarding those measures and inputs with a low certainty level and prioritizing the ones that are more reliable. This results in a more reliable and robust context manager. Vagueness is used among other things to model the perceptions of the users. Using fuzzy sets the context manager can better ascertain the context information as users sense it.

2. As described in Chapter 3 the presented context manager provides a *data fusion mechanism* that is applied in the case that multiple data sources for the same measurement exist in one room. This allows creating a unified view of the environment, avoiding conflicting reactions for the same measures.

3. The context manager can *distribute an inference task among an ecosystem of reasoning engines*. The proposed solution has tackled the problems of semantic inference in smart environments. While, as discussed previously, ontologies are one of the most expressive models to represent the context information, semantic inference can lead to slow reasoning times. The proposed distributed reasoning engine ecosystem solves this problem taking advantage of all the computational capabilities that are usually present in smart environments. To do this, the inference process is divided into smaller sub-units and distributed among several semantic reasoning engines. As explained in Chapter 4, this provides several benefits. The temporal decoupling allows faster inference times as shown in Chapter 5, considerably reducing the total inference time for the whole problem. This is also achieved reducing the number of triples that each reasoning engine has to process. As seen in Chapter 5, the inference time increases exponentially as the data to be processed does. The information is also compartmentalized in different reasoning engines depending on the interests of each one. The spatial decoupling makes the system more reliable, allowing to achieve partial solutions to the inference problem even if one of the nodes suffers some kind of problem. Finally, the presented architecture is able to reorganize itself to respond to

the appearance of new reasoning engines or context providers, making the context manager more adaptable.

The presented context manager has also some constraints:

1. The inclusion of two new aspects in the context model (uncertainty and vagueness) requires some extra knowledge from the system deployers to represent them. A broader knowledge of the specific domain might be necessary to correctly model the system rules. In any case this same problem arises when using other fuzzy or probabilistic inference engines.

2. The main problem to deploy the system in real environments is running the existing semantic reasoning engines into embedded devices. As already stated by Seitz and Schönfelder(Seitz and Schönfelder, 2011), none of the existing semantic reasoning engines can be used in embedded platforms. In order to solve this problem we are currently developing an embedded semantic reasoning engine, also based in CLIPS(cli), which implements the OWL 2 RL profile(owl, c). This reasoning engine will be able to run not only in embedded Linux platforms but also on Android based systems. On the other hand there are several examples of the Jade framework running in portable and mobile platforms like Java ME and Android, so once the semantic reasoning engine is finished we plan to perform a more in-depth test using more computationally constrained devices. The results in more computationally limited devices will surely produce higher inference times, but it will have the same effect in the centralized (only one reasoning engine doing all the inference) and distributed (using our system to split the reasoning problem between different reasoning engines) approaches. Thus, the results obtained in the evaluation chapter can be used as an approximation to the expected results once semantic reasoners are widely available in embedded platforms.

6.2 Contributions

A summary of the contributions explained in this thesis is presented in this section:

- An in-depth state of the art was presented in Chapter 2.
  - Analyzing the current approaches to modeling and reasoning with uncertain and vague context information.
  - Studying the existing architectures for distributed reasoning.
- An ontology for intelligent environments that models the uncertainty and vagueness of the context information was presented in Chapter 3.
  - Allows to model intelligent environments taking into account different types of entities: users, devices, measures, location...
  - Models the vagueness in the perception of the context information using fuzzy sets.
  - Models the uncertainty originated from the capture of the context information by imperfect sensors.
- A reasoning engine for context management that combines semantic reasoning, vagueness (as fuzzy sets) and uncertainty (as a certainty factor) was presented in Chapter 3.
  - Reasons over the spatial and semantic information of the context information.
  - Processes the uncertainty and vagueness of the data to provide a vision of the context information closer to the ambiguous nature of the reality.
  - Applies a data fusion process to provide a holistic picture of the context information of each location.
- An architecture to split the inference task among several reasoning engines was described in Chapter 4.
  - Allows splitting the global inference problem of an intelligent environment among the existing reasoning engines.

  - Attains the temporal decoupling of the different inference units. This allows the inference to be done concurrently in various reasoning engines. The parallelization of the inference process reduces the time required to reach certain conclusions.
  - Attains the spatial decoupling of the inference process. This increases the general robustness of the system making it more fault-tolerant. Problems in one reasoning engine will not stop all the inference, as opposed to a centralized approach.
  - Reduces the number of triples and rules that each reasoning engine has to manage. This allows the use of more computationally constrained devices to carry out the inference process.
  - Compartmentalizes the information according to the different interests of the reasoning engines. This reduces the amount of data that is shared over the network.
  - Allow the dynamic modification of the created reasoning organization. Devices in modern smart environments can change their location frequently (e.g., mobile phones, motes...). The created hierarchy must change to adapt itself to these modifications.

- Finally the previous contributions where combined to offer an implementation of a distributed context management system that provides a more realistic and complete vision of the context information.

6.3 Relevant Publications

The contributions of this thesis have been presented to the scientific community in a series of international forums, such as: journals and conferences.

6.3.1 International JCR Journals

The current versions of the distributed reasoning engine ecosystem and the context ambiguity modeling were published in the following journals:

- Aitor Almeida and Diego López-de-Ipiña. A Distributed Reasoning Engine Ecosystem for Semantic Context-Management in Smart Environments MDPI. vol. 12, no. 8, pp. 10208-10227, DOI: 10.3390/s120810208, ISSN 1424-8220, JCR Impact Factor (2011): 1.739, Q1. Basel, Switzlerland, July 2012.

- Aitor Almeida and Diego López-de-Ipiña. Assessing Ambiguity of Context Data in Intelligent Environments: Towards a More Reliable Context Managing System Sensors (Journal). Volume 12, Issue 4, pp 4934-4951. MDPI. JCR Impact Factor (2011): 1.739, Q1. DOI: 10.3390/s120404934. April 2012.

A first approach to modeling and assessing vagueness in context management applied to adaptive interfaces was published in the following journals:

- Aitor Almeida, Pablo Orduña, Eduardo Castillejo, Diego López-de-Ipiña and Marcos Sacristan. A method for automatic generation of fuzzy membership functions for mobile device's characteristics based on Google Trends. Computers in Human Behavior, Volume 29, Issue 2, pp 510–517. Impact Factor (2011): 2.293, Q1. DOI: 10.1016/j.chb.2012.06.005. March 2013.

- Aitor Almeida, Pablo Orduña, Eduardo Castillejo, Diego López-de-Ipiña, Marcos Sacristán. Imhotep: an approach to user and device conscious mobile applications. Personal and Ubiquitous Computing (Journal). Volume 15, Issue 4, pp 419–429. Springer. Impact Factor (2009): 1.554, Q2. ISSN: 1617-4909. DOI: 10.1007/s00779-010-0359-8. January 2011.

6.3.2 International Conferences

The current versions of the distributed reasoning engine ecosystem and the context ambiguity modeling were presented in the following conferences:

- Aitor Almeida and Diego López de Ipiña. An Inference Sharing Architecture For a More Efficient Context Reasoning. Proceedings of the 4th International Workshop on Sensor Networks and Ambient Intelligence (SENAMI 2012). Lugano, Switzerland, 2012. ISBN: 978-1-4673-0906-6

- Aitor Almeida and Diego López de Ipiña. An Approach to More Reliable Context-Aware Systems By Assesing Ambiguity: Taking Into Account Indetermination and Vagueness in Smart Environments. Proceedings of the 2nd International Conference on Pervasive Embedded Computing and Communication Systems (PECCS 2012). Rome, Italy, 2012. ISBN: 978-989-8565-00-6

- Aitor Almeida, Diego López-de-Ipiña. Modelling and Managing Ambiguous Context in Intelligent Environments. Proceedings of the 5th International Symposium of Ubiquitous Computing and Ambient Intelligence (UCAMI 2011). Riviera Maya, Mexico, December 2011. ISBN:978-84-694-9677-0.

A first approach to modeling and assessing vagueness in context management applied to adaptive interfaces was presented in the following conferences:

- Aitor Almeida, Pablo Orduña, Eduardo Castillejo, Diego López-de-Ipiña, Marcos Sacristán. An approach to automatic generation of fuzzy membership functions using popularity metrics. 4th World Summit on the Knowledge Society. Mykonos, Greece, September, 2011. Information Systems, E-learning, and Knowledge Management Research, Communications in Computer and Information Science, Vol. 278, December 14, 2012. ISBN 978-3-642-35878-4

- Aitor Almeida, Pablo Orduña, Eduardo Castillejo, Diego López-de-Ipiña, Marcos Sacristán. Adaptative applications for heterogeneous intelligent environments. ICOST 2011: 9th International Conference on Smart Homes and Health Telematics. Montréal, Canada, June 2011. LNCS6719, Toward Useful Services for Elderly and People with Disabilities, Springer, ISBN: 978-3-642-21534-6, pp. 1-8

- Aitor Almeida, Pablo Orduña, Eduardo Castillejo, Diego López de Ipiña, Marcos Sacristán. A user-centric approach to adaptable mobile interfaces Proceedings of the II International Workshop of Ambient Assisted Living (IWAAL 2010), p.p. 153-160 Valencia, Spain, September 7-10, 2010 (ISBN: 978-84-92812-67-7)

The first approach to developing a framework for context-aware Intelligent Environments was presented in the following conferences:

- Aitor Almeida, Diego López de Ipiña, Unai Aguilera, Iker Larizgoitia, Xabier Laiseca, Pablo Orduña and Ander Barbier. An Approach to Dynamic Knowledge Extension and Semantic Reasoning in Highly-Mutable Environments Proceedings of 3rd Symposium of Ubiquitous Computing and Ambient Intelligence 2008, Advances in Soft Computing, vol. 51, Springer, ISSN: 1615-3871, ISBN: 978-3-540-85866-9, , University of Salamanca, SPAIN, 22-24 October, 2008

- D. López de Ipiña, A. Almeida, U. Aguilera, I. Larizgoitia, X. Laiseca, P. Orduña, A. Barbier, J.I. Vazquez. Dynamic Discovery and Semantic Reasoning for Next Generation Intelligent Environments. The 4th IET international conference on Intelligent Environments IE08. 2008. University of Washington, Seattle, USA. ISBN: 978-0-86341-894-5

- Aitor Almeida, Diego López de Ipiña, Unai Aguilera, Iker Larizgoitia, Xabier Laiseca, Pablo Orduña, Ander Barbier. Dynamic Ontology Enrichment and Reasoning in AmI Environments Proceedings of the 3rd Workshop on Artificial Intelligence Techniques for Ambient Intelligence (AITAmI2008), Patras, Greece. 21st-22nd of July 2008. Co-located event of ECAI 2008. ISBN: 978-960-6843-07-5.

6.4 Future Work

Several areas were more in-depth research can be pursued have been identified. Below each area is described, providing ideas about the possible research and in some cases explaining the steps that have been already taken.

6.4.1 Improving the Existing Semantic Reasoning Engine Surveys

The review of the state of the art could be improved by an in-depth study of the existing semantic reasoning engines. At the time of writing this thesis, there is no sur-

vey that compares the capacities and limitations of the existing reasoning engines. An analysis of both their qualitative (supported semantic expressivity, reasoning algorithm, rule support, supported ontology serialization formats, used software license...) and quantitative (inference times, consumed memory...) features would greatly help the community when selecting a semantic reasoning engine for their applications.

6.4.2 Creating a Semantic Reasoning Engine for Mobile Devices

At the time of writing this thesis and as stated by Seitz and Schönfelder(Seitz and Schönfelder, 2011), there is no semantic reasoning engine that fully supports OWL DL(owl, f) or OWL 2 Profiles(owl, c) in mobile devices. This could be done using Pocket KRHyper(Sinner and Kleemann, 2005) or a port of CLIPS(cli) for Android and translating the rules of the OWL inference to the target reasoning engine. We have already started working on a adaptation of the OWL 2 RL profile that runs in a customized version of CLIPS for Android. Having a semantic reasoning engine that runs on mobile devices will help us to take advantage of all the inference capabilities of the Intelligent Environments, making the inference splitting architecture even more useful.

6.4.3 Measuring Automatically the Certainty Factor of the Sensors

The proposed system could be improved creating a mechanism that automatically assesses the certainty factor of a sensor comparing its data with the one provided by other sensors. This will allow the system to identify and discard malfunctioning sensors automatically. The whole process could be enhanced including a feedback based learning mechanism that fine-tunes over the time the certainty and vagueness data of the modeled intelligent environment.

6.4.4 Creating a Process to Build the Fuzzy Membership Functions

Currently creating the fuzzy membership functions for our system involves a knowledge eliciting process with the domain experts to model them. Turning this into an automatic process would ease the system deployment, making it possible to perform it without the intervention of the domain experts. A method to automatically generate the fuzzy membership functions of certain parts of the context information(Almeida et al., 2013) as already been developed in the previous work of this thesis, but it is necessary to generalize it for other aspects of the context (other devices, users, measures...).

6.4.5 Improving the Negotiation Process for the Inference Splitting

A more complex negotiation process where the computational capabilities of each device are taken into account to better divide the inference according to them should be developed. This will allow to build a more efficient reasoning architecture that will improve the global inference time. It would also be interesting to evaluate how dynamically reassigning the rules from one reasoning engine to another according to their capabilities would affect the system. To do this it would be necessary to automatically map the *"inference flow"* of the domain, identifying those inference sub-units that can be parallelized and organizing the rules so that the reasoning architecture can achieve the maximum efficiency.

6.5 Final Remarks

With this thesis I have tried to make significant contributions in a research area that I deem will further help a more widespread adoption of Intelligent Environments. I hope that the presented conclusions and outlined future work will help other researches into further extending the state of the art in this area, providing useful guidelines for future research.

Bibliography

ANTLR parser generator. http://www.antlr.org/.

Apache jena - reasoners and rule engines: Jena inference support. http://jena.apache.org/documentation/inference/index.html.

CLIPS: a tool for building expert systems. http://clipsrules.sourceforge.net/.

Jade - java agent DEvelopment framework. http://jade.tilab.com/.

jFuzzyLogic: open source fuzzy logic library and FCL language implementation. http://jfuzzylogic.sourceforge.net/html/index.html.

OWL 2 web ontology language document overview (second edition). http://www.w3.org/TR/owl2-overview/.

OWL 2 web ontology language manchester syntax. http://www.w3.org/TR/2009/NOTE-owl2-manchester-syntax-20091027/.

OWL 2 web ontology language profiles (second edition). http://www.w3.org/TR/owl2-profiles/.

OWL 2 web ontology language structural specification and functional-style syntax. http://www.w3.org/TR/2009/REC-owl2-syntax-20091027/.

OWL 2 web ontology language XML serialization. http://www.w3.org/TR/2009/REC-owl2-xml-serialization-20091027/.

OWL web ontology language guide. http://www.w3.org/TR/owl-guide/.

OWL web ontology language semantics and abstract syntax. http://www.w3.org/TR/owl-semantics/.

Pellet: OWL 2 reasoner for java. http://clarkparsia.com/pellet/.

RDF semantics. http://www.w3.org/TR/rdf-mt/.

RDF/XML syntax specification (revised). http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/.

Turtle - terse RDF triple language. http://www.w3.org/TeamSubmission/turtle/.

(2002). FIPA ACL message structure specification.

(2007). *Fuzzy Control Programming*. Number Part 7 in Programmable Controllers. American National Standards Institute (ANSI), Washington, DC, USA.

Adjiman, P., Chatalic, P., Goasdoué, F., Rousset, M. C., and Simon, L. (2006). Distributed reasoning in a peer-to-peer setting: Application to the semantic web. *Journal of Artificial Intelligence Research*, 25(1):269–314.

Almeida, A. and López-de Ipiña, D. (2012). Assessing ambiguity of context data in intelligent environments: Towards a more reliable context managing system. *Sensors*, 12(4):4934–4951.

Almeida, A., López-de Ipiña, D., Aguilera, U., Larizgoitia, I., Laiseca, X., Orduña, P., and Barbier, A. (2009). An approach to dynamic knowledge extension and semantic reasoning in highly-mutable environments. In Corchado, J. M., Tapia, D. I., and Bravo, J., editors, *3rd Symposium of Ubiquitous Computing and Ambient Intelligence 2008*, number 51 in Advances in Soft Computing, pages 265–273. Springer Berlin Heidelberg.

Almeida, A., Orduña, P., Castillejo, E., López-De-Ipiña, D., and Sacristán, M. (2011). Imhotep: an approach to user and device conscious mobile applications. *Personal Ubiquitous Comput.*, 15(4):419–429.

Almeida, A., Orduña, P., Castillejo, E., López-de Ipiña, D., and Sacristán, M. (2013). A method for automatic generation of fuzzy membership functions for mobile device's characteristics based on google trends. *Computers in Human Behavior*, 29(2):510–517.

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

Baltrunas, L., Kaminskas, M., Ludwig, B., Moling, O., Ricci, F., Aydin, A., Lüke, K.-H., and Schwaiger, R. (2011). InCarMusic: context-aware music recommendations in a car. In Huemer, C. and Setzer, T., editors, *E-Commerce and Web Technologies*, number 85 in Lecture Notes in Business Information Processing, pages 89–100. Springer Berlin Heidelberg.

Black, M. (1937). Vagueness. an exercise in logical analysis. *Philosophy of science*, 4(4):427–455.

Bloch, I. (1996). Information combination operators for data fusion: a comparative review with classification. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 26(1):52 –67.

Bobillo, F. and Straccia, U. (2009). An OWL ontology for fuzzy OWL 2. In Rauch, J., Raś, Z. W., Berka, P., and Elomaa, T., editors, *Foundations of Intelligent Systems*, number 5722 in Lecture Notes in Computer Science, pages 151–160. Springer Berlin Heidelberg.

Brown, P. (1996). The stick-e document: a framework for creating context-aware applications. *Proceedings of the Electronic Publishing*, 8(2):259–272.

Cabitza, F., Sarini, M., and Dal Seno, B. (2005). DJess - a context-sharing middleware to deploy distributed inference systems in pervasive computing domains. In *International Conference on Pervasive Services, 2005. ICPS '05. Proceedings*, pages 229 – 238, Santorini, Greece.

Carriero, N. and Gelernter, D. (1989). Linda in context. *Commun. ACM*, 32(4):444–458.

Chen, H. (2004). An intelligent broker architecture for pervasive context-aware systems (phd thesis).

Chen, H., Perich, F., Finin, T., and Joshi, A. (2004). SOUPA: standard ontology for ubiquitous and pervasive applications. In *The First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004*, pages 258 – 267.

Chu, M., Mitter, S., and Zhao, F. (2003). An information architecture for distributed inference on ad hoc sensor networks. In *Proceedings of Forty-first Annual Allerton Conference on Communication, Control, and Computing*, volume 41, pages 90–99, Monticello, IL, USA.

Costa, P. C. G. d., Laskey, K. B., and Laskey, K. J. (2008). PR-OWL: a bayesian ontology language for the semantic web. In Costa, P. C. G. d., d'Amato, C., Fanizzi, N., Laskey, K. B., Laskey, K. J., Lukasiewicz, T., Nickles, M., and Pool, M., editors, *Uncertainty Reasoning for the Semantic Web I*, number 5327 in Lecture Notes in Computer Science, pages 88–107. Springer Berlin Heidelberg.

Dey, A. K. (2001). Understanding and using context. *Personal and Ubiquitous Computing*, 5(1):4–7.

Dey, A. K., Abowd, G. D., and Salber, D. (2001). A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human–Computer Interaction*, 16(2-4):97–166.

Dey, A. K. and Mankoff, J. (2005). Designing mediation for context-aware applications. *ACM Trans. Comput.-Hum. Interact.*, 12(1):53–80.

Ding, Z., Peng, Y., and Pan, R. (2006). BayesOWL: uncertainty modeling in semantic web ontologies. In Dr, Z. M., editor, *Soft Computing in Ontologies and Semantic Web*, number 204 in Studies in Fuzziness and Soft Computing, pages 3–29. Springer Berlin Heidelberg.

Engelmore, R. and Morgan, T. (1988). *Blackboard system*. Addison-Wesley, Wokingham, England.

Fukushige, Y. (2004). Representing probabilistic knowledge in the semantic web. In *Proceedings of the W3C Workshop on Semantic Web for Life Sciences*, page 27–28, Cambridge, MA, USA.

Garlan, D., Siewiorek, D., Smailagic, A., and Steenkiste, P. (2002). Project aura: toward distraction-free pervasive computing. *IEEE Pervasive Computing*, 1(2):22 –31.

Gu, T., Kwok, Z., Koh, K. K., and Pung, H. K. (2007). A mobile framework supporting ontology processing and reasoning. In *Proc. of the 2nd Workshop on Requirements and Solutions for Pervasive Software Infrastructures (RSPSI'07), in conjunction with the 9th International Conference on Ubiquitous Computing (Ubicomp'07)*, Austria.

Gu, T., Pung, H., and Zhang, D. (2004). Toward an OSGi-based infrastructure for context-aware applications. *IEEE Pervasive Computing*, 3(4):66 – 74.

Guan, D., Yuan, W., Cho, S. J., Gavrilov, A., Lee, Y.-K., and Lee, S. (2007). Devising a context selection-based reasoning engine for context-aware ubiquitous computing middleware. In Indulska, J., Ma, J., Yang, L. T., Ungerer, T., and Cao, J., editors, *Ubiquitous Intelligence and Computing*, number 4611 in Lecture Notes in Computer Science, pages 849–857. Springer Berlin Heidelberg.

Guestrin, S. F. C., Paskin, M., and Sukthankar, R. (2007). Distributed inference in dynamical systems. In *Advances in Neural Information Processing Systems 19: Proceedings of the 2006 Conference*, volume 19, page 433. MIT Press.

Hong, J.-y., Suh, E.-h., and Kim, S.-J. (2009). Context-aware systems: A literature review and classification. *Expert Systems with Applications*, 36(4):8509–8522.

Hsieh, F.-S. and Chiang, C. Y. (2009). Workflow planning in holonic manufacturing systems with extended contract net protocol. In Chien, B.-C., Hong, T.-P., Chen, S.-M., and Ali, M., editors, *Next-Generation Applied Intelligence*, number 5579 in Lecture Notes in Computer Science, pages 701–710. Springer Berlin Heidelberg.

Huang, W., Zhang, X., and Wei, X. (2011). An improved contract net protocol with multi-agent for reservoir flood control dispatch. *Journal of Water Resource and Protection*, 3(10):735–746.

Kagal, L., Finin, T., and Joshi, A. (2003). A policy based approach to security for the semantic web. In Fensel, D., Sycara, K., and Mylopoulos, J., editors, *The Semantic Web - ISWC 2003*, number 2870 in Lecture Notes in Computer Science, pages 402–418. Springer Berlin Heidelberg.

Knabe, T., Schillo, M., and Fischer, K. (2002). Improvements to the FIPA contract net protocol for performance increase and cascading applications. In *In International Workshop for Multi-Agent Interoperability at the German Conference on AI*, Aachen, Germany.

Korpipaa, P., Mantyjarvi, J., Kela, J., Keranen, H., and Malm, E. (2003). Managing context information in mobile devices. *IEEE Pervasive Computing*, 2(3):42 – 51.

Lee, C.-S., Jian, Z.-W., and Huang, L.-K. (2005). A fuzzy ontology and its application to news summarization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 35(5):859 –880.

Lopez-de Ipina, D., Almeida, A., Aguilera, U., Larizgoitia, I., Laiseca, X., Orduna, P., Barbier, A., and Vazquez, J. (2008). Dynamic discovery and semantic reasoning for next generation intelligent environments. In *2008 IET 4th International Conference on Intelligent Environments*, pages 1 –10.

Lukasiewicz, T. and Straccia, U. (2008). Managing uncertainty and vagueness in description logics for the semantic web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(4):291–308.

Nottelmann, H. and Fuhr, N. (2006). Adding probabilities and rules to OWL lite subsets based on probabilistic datalog. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 14(01):17–41.

Orchard, R. (1998). FuzzyCLIPS version 6.04A user's guide.

Oren, E., Kotoulas, S., Anadiotis, G., Siebes, R., ten Teije, A., and van Harmelen, F. (2009). Marvin: Distributed reasoning over large-scale semantic web data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(4):305–316.

Ouelhadj, D., Garibaldi, J., MacLaren, J., Sakellariou, R., and Krishnakumar, K. (2005). A multi-agent infrastructure and a service level agreement negotiation protocol for robust scheduling in grid computing. In Sloot, P. M. A., Hoekstra, A. G., Priol, T., Reinefeld, A., and Bubak, M., editors, *Advances in Grid Computing - EGC 2005*, number 3470 in Lecture Notes in Computer Science, pages 651–660. Springer Berlin Heidelberg.

Ouelhadj, D., Hanach, C., and Bouzouia, B. (1998). Multi-agent system for dynamic scheduling and control in manufacturing cells. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 3, pages 2128–2133, Multi-agent system for dynamic scheduling and control in manufacturing cells. IEEE.

Ouelhadj, D., Hanachi, C., and Bouzouia, B. (2000). Multi-agent architecture for distributed monitoring in flexible manufacturing systems (FMS). In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 3, pages 2416–2421, San Francisco, California, USA. IEEE.

Parry, D. (2004). A fuzzy ontology for medical document retrieval. In *Proceedings of the second workshop on Australasian information security, Data Mining and Web Intelligence, and Software Internationalisation - Volume 32*, ACSW Frontiers '04, page 121–126, Darlinghurst, Australia, Australia. Australian Computer Society, Inc.

Paskin, M., Guestrin, C., and McFadden, J. (2005). A robust architecture for distributed inference in sensor networks. In *Fourth International Symposium on Information Processing in Sensor Networks, 2005. IPSN 2005*, pages 55 – 62, Los Angeles, California, USA.

Perich, F. (2004). Mogatu BDI ontology. *University of Maryland, Baltimore County*.

Preuveneers, D., Bergh, J. V. d., Wagelaar, D., Georges, A., Rigole, P., Clerckx, T., Berbers, Y., Coninx, K., Jonckers, V., and Bosschere, K. D. (2004). Towards an extensible context ontology for ambient intelligence. In Markopoulos, P., Eggen, B., Aarts, E., and Crowley, J. L., editors, *Ambient Intelligence*, number 3295 in Lecture Notes in Computer Science, pages 148–159. Springer Berlin Heidelberg.

Roman, M., Hess, C., Cerqueira, R., Ranganathan, A., Campbell, R., and Nahrstedt, K. (2002). A middleware infrastructure for active spaces. *IEEE Pervasive Computing*, 1(4):74 –83.

Sachs, M., Dan, A., Nguyen, T., Kearney, R., Shaikh, H., and Dias, D. (2000). Executable trading-partner agreements in electronic commerce. *IBM TJ Watson Research Center*.

Salber, D., Dey, A. K., and Abowd, G. D. (1999). The context toolkit: aiding the development of context-enabled applications. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, CHI '99, page 434–441, New York, NY, USA. ACM.

Sandholm, T. (1993). An implementation of the contract net protocol based on marginal cost calculations. In *Proceedings of the National Conference on Artificial Intelligence*, pages 256–256, Washington DC, USA. JOHN WILEY & SONS LTD.

Satyanarayanan, M. (2001). Pervasive computing: vision and challenges. *IEEE Personal Communications*, 8(4):10 –17.

Seitz, C. and Schönfelder, R. (2011). Rule-based OWL reasoning for specific embedded devices. In Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., and Blomqvist, E., editors, *The Semantic Web – ISWC 2011*, number 7032 in Lecture Notes in Computer Science, pages 237–252. Springer Berlin Heidelberg.

Serafini, L. and Tamilin, A. (2005). DRAGO: distributed reasoning architecture for the semantic web. In Gómez-Pérez, A. and Euzenat, J., editors, *The Semantic*

Web: Research and Applications, number 3532 in Lecture Notes in Computer Science, pages 361–376. Springer Berlin Heidelberg.

Sinner, A. and Kleemann, T. (2005). KRHyper – in your pocket. In Nieuwenhuis, R., editor, *Automated Deduction – CADE-20*, number 3632 in Lecture Notes in Computer Science, pages 452–457. Springer Berlin Heidelberg.

Smith, R. (1980). The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12):1104 –1113.

Smith, R. G. and Davis, R. (1981). Frameworks for cooperation in distributed problem solving. *IEEE Transactions on Systems, Man and Cybernetics*, 11(1):61 –70.

Sousa, J. and Garlan, D. (2002). Aura: An architectural framework for user mobility in ubiquitous computing environments. *Computer Science Department*.

Steventon, A. and Wright, S. (2006). *Intelligent spaces*. Computer Communications and Networks. Springer Verlag, London.

Stoilos, G., Simou, N., Stamou, G., and Kollias, S. (2006). Uncertainty and the semantic web. *IEEE Intelligent Systems*, 21(5):84 –87.

Strang, T. and Linnhoff-Popien, C. (2004). A context modeling survey. In *In: Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing, Nottingham/England*, Nottingham, UK.

Studer, R., Benjamins, R., and Fensel, D. (1998). Knowledge engineering: Principles and methods. *Data & Knowledge Engineering*, 25(1):161–197.

Tho, Q., Hui, S., Fong, A., and Cao, T. H. (2006). Automatic fuzzy ontology generation for semantic web. *IEEE Transactions on Knowledge and Data Engineering*, 18(6):842 –856.

Udrea, O., Subrahmanian, V., and Majkic, Z. (2006). Probabilistic RDF. In *2006 IEEE International Conference on Information Reuse and Integration*, pages 172 –177.

Uschold, M. and Gruninger, M. (1996). Ontologies: principles, methods and applications. *The Knowledge Engineering Review*, 11(02):93–136.

Wang, X., Zhang, D., Gu, T., and Pung, H. (2004). Ontology based context modeling and reasoning using OWL. In *Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, 2004*, pages 18 – 22.

Want, R., Hopper, A., Falcão, V., and Gibbons, J. (1992). The active badge location system. *ACM Trans. Inf. Syst.*, 10(1):91–102.

Weigand, H., Verharen, E., and Dignum, F. (1998). A Language/Action perspective on cooperative information agents. *Accounting, Management and Information Technologies*, 8:39–59.

Weiser, M. (1991). The computer for the 21st century. *Scientific American*, 265(3):94–104.

Wood, A., Stankovic, J., Virone, G., Selavo, L., He, Z., Cao, Q., Doan, T., Wu, Y., Fang, L., and Stoleru, R. (2008). Context-aware wireless sensor networks for assisted living and residential monitoring. *IEEE Network*, 22(4):26 –33.

Xu, L. and Weigand, H. (2001). The evolution of the contract net protocol. In Wang, X. S., Yu, G., and Lu, H., editors, *Advances in Web-Age Information Management*, number 2118 in Lecture Notes in Computer Science, pages 257–264. Springer Berlin Heidelberg.

Yamabe, T., Takagi, A., and Nakajima, T. (2005). Citron: a context information acquisition framework for personal devices. In *11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, 2005. Proceedings*, pages 489 – 495.

Yang, Y. and Calmet, J. (2005). OntoBayes: an ontology-driven uncertainty model. In *International Conference on Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce*, volume 1, pages 457 –463.

Declaration

I herewith declare that I have produced this work without the prohibited assistance of third parties and without making use of aids other than those specified; notions taken over directly or indirectly from other sources have been identified as such. This work has not previously been presented in identical or similar form to any examination board.

The dissertation work was conducted from 2007 to 2013 under the supervision of Diego López de Ipiña at the University of Deusto.

Bilbao,

This dissertation was finished writing in Bilbao on April 8^{th}, 2013