Principles and experiences on creating semantic devices

Juan Ignacio Vazquez
Faculty of Engineering
University of Deusto
48007 Bilbao
ivazquez@eside.deusto.es

Diego López de Ipiña
Faculty of Engineering
University of Deusto
48007 Bilbao
dipina@eside.deusto.es

Abstract

Current approaches for designing smart objects are failing to provide suitable mechanisms for autonomous distributed intelligence. The next wave of Ubiquitous Computing architectures must be composed of devices capable of adapting to new unforeseen situations and collaborating to determine and react to users' needs.

We argue that semantic technologies, conveniently scaled to embedded platforms, can provide suitable context representation mechanisms and the level of intelligence required for dealing with everyday situations in resource-limited devices.

Semantic devices are able to collaboratively interpret context information and choreographically react to situations that were not considered at the moment of design. Their novel future-proof nature along with their socializing capabilities determines a new exciting approach for deploying Ubiquitous Computing scenarios.

1. Introduction

Our environment is being populated by an increasing number of digital devices. Even traditional objects are being substituted by their electronic versions, demanding more skills from users.

The amount of intelligence in these devices does not match the pace at which they are being disseminated through our lives. This situation leads to interaction problems with the environment, since the user is the sole responsible for contextualizing the information and solving cooperation problems between devices; that is, the user is the intelligence provider. One of the major consequences of this situation is that people are continuously disturbed and required to configure, operate and interact with these devices.

We deem that digital objects shall be more intelligent, autonomous, and able to share, interpret and reason upon exchanged information to release users from part of these activities.

In this paper, we propose a novel approach for modeling intelligent devices sharing knowledge and promoting autonomous context-aware reactivity. In our approach, we bring Semantic Web technologies to the Ubiquitous Computing world in order to provide the means for collaboratively transforming data into knowledge and enabling intelligent reasoning mechanisms.

In our model, devices spontaneously discover each other, share context information, perform reasoning through an embedded semantic engine, and adapt their behavior dynamically to create a feel for intelligence in the environment. In order to validate our approach we have developed a number of prototypes that were deployed in experimental scenarios.

Section 2 is devoted to analyzing past efforts in applying Semantic Web technologies into Ubiquitous Computing scenarios. Section 3 introduces the four principles underlaying the *semantic device* concept, along with indications about how these principles are honored in SoaM, an architectural model for semantic devices. Section 4 describes the prototypes and scenarios that were deployed in order to evaluate our designs. Section 5 contains some recommendations we found after the evaluation, which can be useful for future designers. Finally, section 6 provides some conclusions and discussion about open research lines.

2. Related work

Semantic Web technologies, mainly RDF (Resource Description Framework) [23] and OWL (Ontology Web Language) [22], enable the creation of a shared knowledge space for information representation and distributed reasoning based on description logics. In the Semantic Web, URIs are used for representing concepts, while HTTP [3] provides a natural means for retrieving RDF-based descriptions.

Other Ubiquitous Computing initiatives have tried to embrace Semantic Web technologies in the past. Task Computing [11] [16] aimed at creating systems in which users could perform automatic composition of services, based on semantic descriptions. However, Task Computing provides a mechanism for users to control the environment, but not for automatic intelligent reactivity in devices.

Other initiatives such as CoBrA [2] and SOCAM [4] [5] provided both an architecture and an ontology (SOUPA and CONON respectively) to create environments populated by smart devices. However, both of them require a central server to be deployed where all the intelligence resides, acting the devices as simple slave entities out of the context-awareness process.

Gaia [13] [14] also follows this centralized scheme, but providing more advanced reasoning mechanisms.

Recent initiatives, such as Triple Spaces [15] and Semantic Tuple Spaces [8], have also proposed a more distributed space for context information sharing among participating objects.

In general terms, all the architectures feature a correlation among intelligence and centralization: the higher level of intelligence in the system, the more centralized it is.

A single element must be previously deployed in the environment, sometimes concentrating the majority or all of the reasoning processes, sometimes providing supporting services: the Context Broker in CoBrA; the Context Provider Lookup Service, the Context History Service and the Ontology Server in Gaia; and, the Context Interpreter, the Context Database and the Service Location Service in SOCAM.

The main drawback of existing experiences is their centralized approach: with the aim of providing more intelligence, the basic nature of Ubiquitous Computing has been abandoned. The core of their functionality must be deployed in a central server, they are not based on individual devices contributing with their capabilities to the overall environmental intelligence.

Thus, these initiatives lack of the spontaneous and serendipitous collaboration required in Ubiquitous Computing scenarios, because they did not address the problem of adapting semantic intelligence mechanisms to embedded platforms.

3. Principles of semantic devices

3.1. Definition of semantic device

Lassila and Adler [10] introduced the concept of *semantic gadget* to describe devices capable of performing "*discovery and utilization of services without human guidance or intervention, thus enabling formation of device coalitions*".

Although semantic discovery and service composition are identified as two major goals of the semantic gadget concept, no advances have been achieved so far in creating such kind of intelligent and collaborative objects.

However, some of the ideas presented in [10] contributed to clarify our vision for semantic-powered objects, and our definition for semantic device:

> *A* semantic device *is a system that is spontaneously aware of surrounding context information, capable of reasoning and interpreting this information at a semantic level, and finally able to develop a reactive behavior accordingly.*
>
> *A semantic device should be able to spontaneously discover, exchange and share context information with other fellow semantic devices as well as augmenting this context information via reasoning in order to better understand the situation and perform the appropriate reactive response.*

There are four main principles that characterize semantic devices:

- Semantic discovery
- Social behavior
- Semantic reasoning

- Choreographic reactivity

SoaM (Smart Objects Awareness and Adaptation Model) [17] [20] is an architecture for the creation of a collaborative network of individual intelligent devices based on semantic technologies.

A *smobject* (a portmanteau for "smart semantic object") is the software agent in charge of representing a semantic device in the SoaM architecture. The smobject middleware can be run in embedded platforms and promotes the creation of a local network of knowledge that supports individual decisions at devices.

In the following subsections, the four main principles of semantic devices are explained, along with the concrete strategy followed in our model for the implementation of these principles.

3.2. Semantic discovery

The goal of semantic discovery is finding resources in the network by means on "what they are" or "what they do", in an independent manner. The term "resource" must be interpreted in a broad sense as it is defined in Web standards; devices, documents, or information pieces that can be referenced via an URI, are all valid resources.

By describing resources using RDF an entity is able to find relationships among them, and even to obtain new associations by applying appropriate ontologies.

A device can issue a request to the network in order to find resources whose description matches certain conditions. Query languages such as SPARQL [24] or Plant [17] can be used to provide a syntax for creating this kind of expressive queries.

The real advantage of semantic discovery is that it dives through information relationships in order to find solutions to the query by applying a semantic reasoner. For instance, let's take the following scenario:

- Objects are tagged in such a way that a containing object (e.g., a wardrobe, or a backpack) is able to know the identity of the objects directly placed inside. An example of such system is formed by objects tagged with barcodes and barcode readers at the containers.
- A PDA is stored in a backpack.
- The backpack is placed in a room.

If a subject is provided with this information and asked to identify available PDAs in the room he will surely point at the PDA inside the backpack as one of the items. Location is a transitive property, which means that if the PDA is in the backpack, which in turn is placed in the room, the PDA is located in the room after all.

This assertion is not directly provided by the environment, but a form of reasoning has been performed to interpret context information and obtain the new fact "the PDA is stored in the room". This example clearly illustrates the basics and strengths of using description logics in order to augment available context information for evaluating required conditions.

Semantic discovery is at the heart of intelligent behavior in Ubiquitous Computing environments by providing a means for locating the most suitable resources (devices, information pieces) in order to obtain context information or coordinating an intelligent reactive response.

Semantic discovery in smobjects

In SoaM, smobjects implement semantic discovery via mRDP (Multicast Resource Discovery Protocol) [18], a lightweight protocol combining UDP for requests and HTTP for callbacks in such a way that queries can be disseminated throughout the network in order to find suitable matches.

The language for representing the queries is Plant (Pattern Language for N-Triples), a subset of the expressive power of SPARQL, which could also be used in mRDP. Plant is simpler and demands less computing resources than SPARQL, therefore being more appropriate for limited devices.

Plant queries are processed by each semantic device against its knowledge base, populated with RDF-annotated context information, replying to the requesting smobject with the matches to the query.

3.3. Social behavior

Semantic devices are inherently social: they are natively collaborative in the sense that they share all the information they can.

The Web has transitioned from a basically "one publisher - many readers" model to a more collaborative "many publishers - many readers" model, in an approach that was called Web 2.0 [12]. The major representatives of this culture are weblogs, social bookmarking, wikis, RSS feeds and so forth.

We consider that this model can be also applied to semantic devices, featuring a collaborative nature, sharing information, and creating a community of intelligent objects in the environment in order to better serve their users.

Semantic devices behave in a social way because a higher and more useful knowledge can be obtained from the generous contributions of individual entities, rather than from selfishly not sharing information (of course, taking into account privacy concerns).

Semantic devices must be social in order to enable further cooperation.

Social behavior in smobjects

In SoaM, smobjects share their perceptions with other fellow smobjects in the environment. Perceptions are retrieved through physical sensors (temperature, light, location) or virtual sensors (weather or traffic information obtained via websites, static knowledge stored in configuration files).

Perceptions are annotated using RDF and the aggregation of individual perceptions form the whole set of context information about the environment. Thus, every single smobject captures a subset of the context information via its direct perceptions, but shares this information with the others, in such a way that a common knowledge space is created by individual contributions. Smobjects build their own copy of context information by aggregating indirect perceptions obtained from others to their direct perceptions.

This vision has similarities with other alternatives already mentioned such as Triple Spaces and Semantic Tuple Spaces.

3.4. Semantic reasoning

Basically the Semantic Web is a web of knowledge, where concepts and information are represented in a machine readable and understandable form and linked via URIs. Every concept (people, places, objects, time events, verbs, and so forth) can be identified via an unique URI, in such a way that a universe of concepts can be related to each other.

Through the application of simple reasoning mechanisms based on description logics, semantic devices can augment the knowledge base with new facts. These new facts contribute to better interpret and analyze context information, thus enabling the semantic device to select the most appropriate reactive behavior to perform.

Semantic reasoning in smobjects

Smobjects can share and apply ontologies to augment the context information they have obtained from fellow smobjects. The embedded reasoner we have designed at the smobject is composed of two subelements: the MiniOwlReasoner and the MiniRuleReasoner. The MiniOwlReasoner is not a full semantic reasoner, which would be too large for an embedded platform, but a limited, yet powerful Semantic-Web oriented rule-engine.

The MiniOwlReasoner carries out the following steps:

1. It loads available ontologies.
2. It filters the ontologies, selecting the constructions the reasoner is able to deal with. The constructions the smobject's reasoner currently supports in our prototypes are:
    - rdfs:subClassOf
    - owl:sameAs
    - owl:TransitiveProperty
    - owl:SymmetricProperty
    - owl:inverseOf

    Which are the most common and used ontological predicates to create relationships among concepts in ontologies. It is noteworthy how any kind of transitive or symmetrical property can be processed by the MiniOwlReasoner; these kind of properties intrinsically embed a considerable amount of intelligence in any ontology. The MiniOwlReasoner implements a subset of OWL Lite, but it is more complex and powerful than other proposals such as RDF++ [9] by Lassila and almost equivalent to OWL Tiny [1].

3. It generates a rule for each construction, which is added to the rule base. The rules are specifically generated depending on the construction arguments. For example, if **`isLocatedIn`** is declared as a transitive property, a particular rule matching RDF statements with the **`isLocatedIn`** property is generated and added to the rules base. This mechanism performed much better in terms of time devoted to reasoning than other alternatives we also tested.
4. When context information needs to be augmented via ontologies, the stored rules in the rule base are iteratively applied until no additional information is generated.

We deem this strategy to provide a good balance between intelligence and limited resources availability. The MiniOwlReasoner can be further improved in the future to cope with additional constructions if the host platform is powerful enough to support the work load.

3.5. Choreographic reactivity

Semantic devices do not only behave in a social way for sharing information and understanding context, but they may also coordinate their reactions acting in a synchronized way.

One strategy for achieving this coordination with no additional mechanisms is by exposing their behavior as context information. In this way, the current activity or goals of a semantic device is provided to others in the form of RDF triples, using suitable ontologies and vocabularies, as if the device featured a "behavior sensor" whose information is shared with others. Fellow semantic devices would retrieve these data to be aware of a concrete device's goals and react accordingly to synchronize activities.

Of course, additional coordination mechanisms based on specific protocols could also be provided if more fine-grained coordination or higher workflow expressiveness is required.

Acting individually, but exchanging information about their goals and next activities, semantic devices can predict how others may behave and react accordingly, performing in a choreographic way.

Choreographic reactivity in smobjects

Smobjects' behavior is not hard-coded, but provided in the form of XML-based documents called "behavioral profiles". These documents comprise behavioral rules that dictate how a smobject must react to different context information.

Smobject profiles can be accessed by other smobjects in order to find out how the former reacts to different stimuli. This information is also provided by smobjects in RDF using SoaMonto, the SoaM ontology that declares different concepts for representing the state and information of smobjects.

4. Prototypes of semantic devices

Smobjects are semantic devices in the SoaM architecture. An smobject is internally composed of several functional modules. The most important from a semantic perspective are:

- Discovery module: implements the semantic discovery mechanism using mRDP.
- Perceptors: semantic gateways to connected sensors. They provide RDF annotations of directly captured information.
- Reasoner: performs semantic reasoning (MiniOwlReasoner) as well as domain rules-based reasoning (MiniRuleReasoner).
- Awareness engine: coordinates the behavior of the smobject by checking the behavioral profiles against current context information, and carrying out desired operations through the effectors.

Figure 1 illustrates the complete internal structure of the smobject. A more detailed description of the modules and activities can be found in [20] and [19].

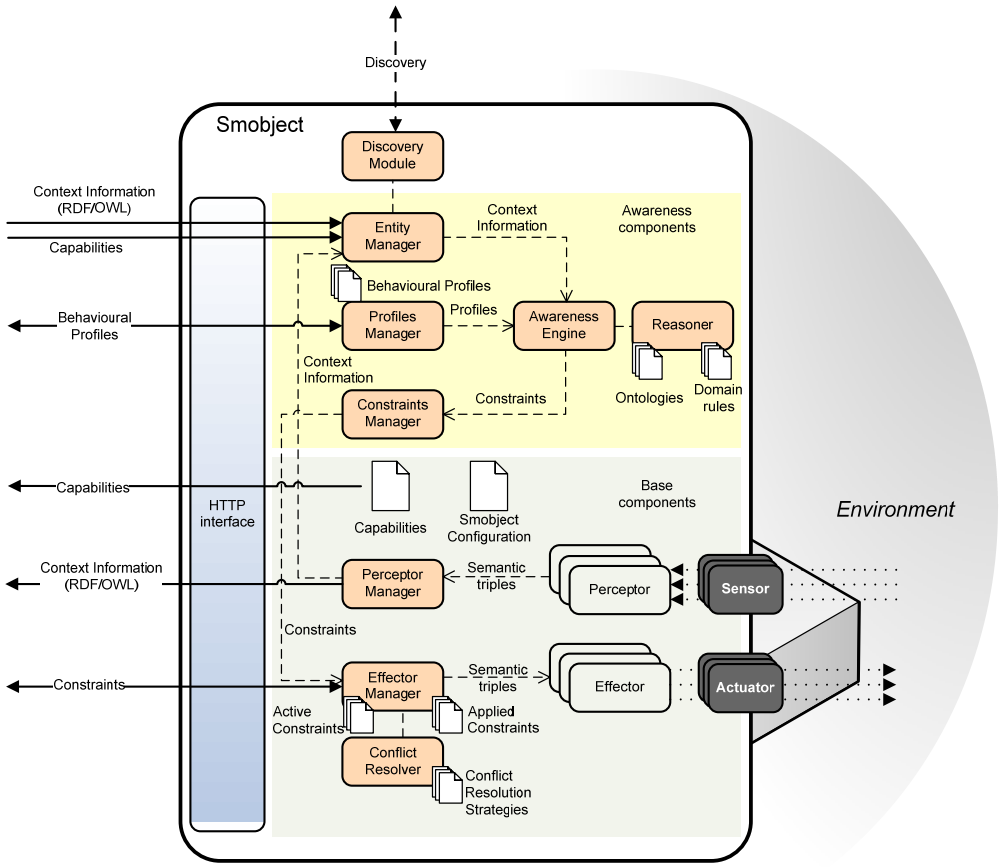

Figure 1. Smobject internal components.

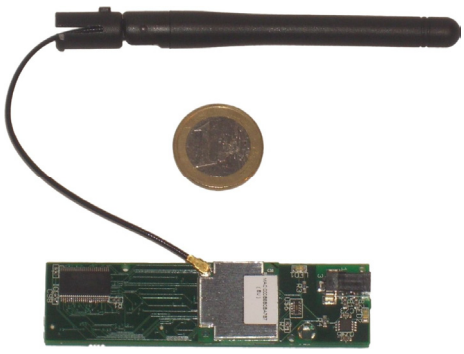

Figure 2. A Gumstix platform, with Wi-Fi connectivity, that hosted the smobject prototype.

Platform	Computing power	Size (volume)	Power-per-cm3
Laptop	2.33 GHz	2596.21 cm3	0.89 MHz/cm3
PDA	416 MHz	150.80 cm3	2.76 MHz/cm3
Digi ConnectCore 7U	55 MHz	12.08 cm3	4.55 MHz/cm3
Digi ConnectCore 9U	180 MHz	12.08 cm3	14.90 MHz/cm3
Gumstix Connex 400xm	400 MHz	10.08 cm3	39.68 MHz/cm3

Table 1. Relation among computing power and platform size.

Java was selected as the platform for implementing the smobject middleware, which was tested in several embedded platforms with limited Java VM:

- ARM7-based ConnectCore 7U with μClinux and Mika (Java VM)
- ARM9-based ConnectCore 9U with Linux and Mika (Java VM)
- Intel XScale PXA255 Gumstix with Linux and JamVM (Java VM)

Finally, Gumstix was selected as the hosting platform for the smobject prototypes, since it provided the best balance between computing power, size and connectivity (see Figure 2).

Table 1 illustrates the computer power density (per cm^3) of the platforms where the smobject middleware was deployed. Performance measures of the smobject in some of these platforms are discussed in [21] and [20].

Several scenarios suitable for semantic devices evaluation were designed and deployed using the smobject prototypes as described below (see Figure 3, more detailed descriptions can be found in [17]).

4.1. SmartPlants: autonomous objects that interact with their environment

One of the scenarios we envisioned at the beginning of the research was to create an artifact that could be attached to real objects, augmenting their perceptions and providing them with intelligent capabilities. An additional challenge was to attach this kind of artifact to living entities, such as plants, in a way that could result in intelligent behavior carried out by the entities from the user's point of view.

Creating this kind of "smart plants" raised several new important implications such as:

- They could become first-class citizens in the environment, rather than passive elements. They could inject their preferences into their location to influence temperature, humidity or lighting settings.
- They could be perceived as autonomic systems [7] in a twofold view: as normal living beings they try to survive and adapt to environmental conditions; but also, as augmented intelligent entities they can interact and communicate with surrounding objects to create a more suitable and healthy environment.

The plant was reactive via a number of behavioral profiles, in such a way that it was continuously aware of the temperature and light conditions about different nearby locations provided by a wireless sensor network. The plant asked the user to me moved to the most suitable place using a voice generated via a TTS (text-to-speech) embedded engine.

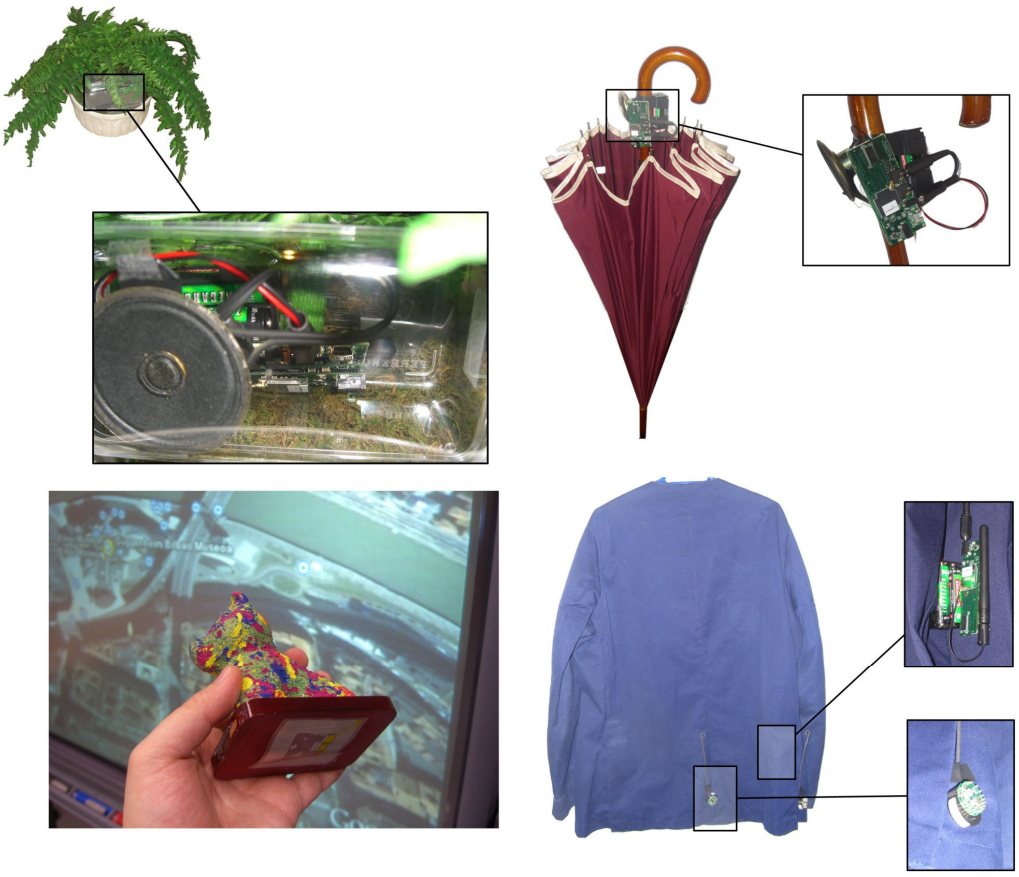

Figure 3. Different prototypes of semantic devices.

4.2. Aware-Umbrella: a reactive device integrating local and global communication

An additional challenge was the ability to seamless integrate local and global information sources in order to augment local intelligence and knowledge by injecting externally obtained context information. The most probable, but not unique, source for this information is the Internet, and particularly, available dynamic web services.

Our goal in this scenario was to design some kind of smart object that could be aware of both environment-provided and Internet-provided information in order to take decisions and look more intelligent from a users' perspective.

Our choice was to create a smart umbrella that could obtain current weather information from both surrounding sensors and the Internet, as well as to obtain the weather forecast for the next hours through the Internet. The smart umbrella reacted when the user was leaving home without taking it by issuing a synthesized voice alert.

The umbrella obtained context information from the Internet through a "virtual software sensor", a small piece of code that connected to the Internet to get weather information about the town (provided by the location) and semantized these data using a weather ontology. The umbrella finally checked the state of the door in case the

user was leaving when raining in order to decide whether to issue the voice alert.

4.3. WorkSafe: a protective working agent enforcing user care in dangerous environments

We also wanted to demonstrate how smobjects could be used in heterogeneous working environments to provide workers with intelligent safety measures. We designed an scenario populated with different electrical tools, a number of containers filled with flammable and chemical products and sensorized workwear. All these objects were transformed into smobjects, so they were able to exchange information about themselves and their perceptions.

This kind of intelligent environment works in the periphery of attention to preserve workers' safety, not only alerting about possible dangerous situations but also actively reacting (e.g. switching off electrical tools if flammable substances are nearby) to unforeseen hazards.

5. Recommendations about designing semantic devices

Based on our experiences, we identified a number of recommendations that may be useful for those researchers intending to design semantic devices using SoaM or any other similar architectures.

5.1. Use semantic devices to react to unforeseen situations

The real strength of semantic devices is interpreting a new situation to perform actions intelligently. In the protective working environment scenario the main behavior of a display is "if there is a danger, alert the user". The interpretation of a situation as dangerous is performed in real time, based on available context information, ontologies and domain rules.

In this way, unforeseen situations are supported as long as semantic information about their constituent concepts is provided.

5.2. Use natural language annotations in context information

By using properties such as `rdfs:label` throughout the context information, a natural dialog with users can be easily built if required. Certain situations require alerting the user about some concrete fact, e.g. when a smart plant requests for a location change, or when a dangerous situation is detected in industrial premises.

In these cases, natural language annotations of context information served to dynamically build alerting messages that could be synthesized (via a TTS engine) or displayed to the user.

5.3. Complement ontologies with domain rules

Description logics provide an appropriate level of context interpretation, but domain rules are generally also required. For most of the situations and scenarios we have been dealing with, ontological knowledge and domain rules acted in complementary ways to augment the context information and take the appropriate decisions.

We found that few scenarios required only description logics in order to implement the expected intelligent behavior. This issue does not require extra implementation work since, at least in our case, the same rule engine core was used for ontological and domain rules-based reasoning.

SWRL [6] and the current work in the W3C Rule Interchange Format Group may help here.

5.4. A little optimization in reasoning goes a long way

The most costly activity in terms of time and energy in semantic devices is reasoning. The reactivity level of an entity depends on its ability to immediately perceive changes in context information, produce a new set of facts inferred from those changes, and modify its behavior accordingly.

As usual with embedded programming, we found that small optimizations in the reasoning engine remarkably reduced the overall processing time of context information.

5.5. Use energy efficient wireless bearers

We used Wi-Fi as wireless bearer in order to support TCP/IP and HTTP communication

protocols and being completely synergistic with the Web and the Semantic Web model. However, communication bearers such as Zigbee should be used for energy efficiency purposes.

The challenge here is to design a layer with the expressive power of HTTP over this kind of bearers, so that the Semantic Web connective model can be seamlessly deployed over a different protocol family.

6. Conclusion

We consider the concept of "semantic device" to be of foremost importance for Ubiquitous Computing. It comprises some of the fundamental aspects researchers have been looking for during the last years, especially intelligent context-awareness and serendipitous collaboration.

In our research we investigated the foundations of these new wave of social objects and developed an architectural model for implementing them. We deployed experimental prototypes in several scenarios which were clear candidates for being populated by semantic devices, and described a set of recommendations based on our experiences.

Our future research lines include the design of a Zigbee-friendly alternative for HTTP, in such a way that the current Semantic Web model, which is natively linked to the Web communication architecture can be implemented over a more energy efficient bearer. This research involves finding new mechanisms for serializing RDF in a compressed form suitable for wireless nodes.

Moreover, the results would pave the way for semantic sensor network nodes: entities that generate self-descriptive information flows that can be dynamically discovered and aggregated.

References

[1] Dave Beckett. Swad-Europe deliverable 3.11: Developer workshop report 4 - workshop on semantic web storage and retrieval. Technical report, World Wide Web Consortium, 2004.

[2] Harry Chen, Tim Finin, and Anupam Joshi. An intelligent broker for contextaware systems. In *Adjunct Proceedings of Ubicomp 2003*, pages 183-184. UbiComp, UbiComp, October 2003.

[3] Roy T. Fielding, James Gettys, Jeffrey C. Mogul, Henrik Frystyk, Larry Masinter, Paul J. Leach, and Tim Berners-Lee. *Hypertext Transfer Protocol - HTTP/1.1*, 1999. IETF RFC 2616.

[4] Tao Gu, Hung Keng Pung, and Da Qing Zhang. Toward an OSGI-based infrastructure for context-aware applications. *IEEE Pervasive Computing*, 3(4):66-74, 2004.

[5] Tao Gu, Edmond Tan, Hung Keng Pung, and Daqing Zhang. Contextpeers: Scalable peer-to-peer search for context information. In *Proceedings of the International Workshop on Innovations in Web Infrastructure (IWI 2005), in conjunction with the 14th World Wide Web Conference (WWW 2005)*, 2005.

[6] Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, and Mike Dean. *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*. World Wide Web Consortium, May 2004. W3C Member Submission.

[7] Jeffrey O. Kephart and David M. Chess. The vision of autonomic computing. *Computer*, 36(1):41-50, 2003.

[8] Deepali Khushraj, Ora Lassila, and Tim Finin. stuples: Semantic tuple spaces. In *Proceedings of Mobiquitous 2004: The First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*, pages 268-277, 2004.

[9] Ora Lassila. *Semantic Web, Quo Vadis?*, October 2006. Keynote of the Ninth Scandinavian Conference on Articial Intelligence (SCAI 2006).

[10] Ora Lassila and Mark Adler. Semantic gadgets: Ubiquitous computing meets the semantic web. In *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential*, pages 363-376, 2003.

[11] Ryusuke Masuoka, Yannis Labrou, Bijan Parsia, and Evren Sirin. Ontology enabled pervasive computing applications. *IEEE Intelligent Systems*, 18(5):68-72, September-October 2003.

[12] Tim O'Reilly. *What Is Web 2.0. Design Patterns and Business Models for the Next Generation of Software*, September 2005. http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html. Retrieved on January 2007.

[13] Manuel Román and Roy H. Campbell. Gaia: enabling active spaces. In *Proceedings of the 9th workshop on ACM SIGOPS European workshop*, pages 229-234, New York, NY, USA, 2000. ACM Press.

[14] Manuel Román, Christopher Hess, Renato Cerqueira, Anand Ranganathan, Roy H. Campbell, and Klara Nahrstedt. A middleware infrastructure for active spaces. *IEEE Pervasive Computing*, 1(4):74-83, 2002.

[15] Omair Shafiq, Ioan Toma, Reto Krummenacher, Thomas Strang, and Dieter Fensel. Using triple space computing for communication and coordination in semantic grid. In *Proceedings of the 3rd Semantic Grid Workshop colocated with the 16th Global Grid Forum*, 2006.

[16] Zhexuan Song, Ryusuke Masuoka, Jonathan Agre, and Yannis Labrou. Task computing for ubiquitous multimedia services. In MUM '04: *Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia*, pages 257-262, New York, NY, USA, 2004. ACM Press.

[17] Juan Ignacio Vazquez. *A Reactive Behavioural Model for Context-Aware Semantic Devices*. PhD thesis, University of Deusto, 2007.

[18] Juan Ignacio Vazquez and Diego López de Ipiña. mRDP: An HTTP-based Lightweight Semantic Discovery Protocol. *The International Journal of Computer and Telecommunications Networking*, (to be published), 2007.

[19] Juan Ignacio Vazquez, Diego López de Ipiña, and Iñigo Sedano. SoaM: An environment adaptation model for the Pervasive Semantic Web. In *Proceedings of the 2nd Ubiquitous Web Systems and Intelligence Workshop (UWSI 2006), colocated with ICCSA 2006. Lecture Notes in Computer Science - LNCS*, volume 3983, pages 108-117, May 2006.

[20] Juan Ignacio Vazquez, Diego López de Ipiña, and Iñigo Sedano. SoaM: A Web-powered architecture for designing and deploying pervasive semantic devices. *IJWIS - International Journal of Web Information Systems*, (to be published), 2007.

[21] Juan Ignacio Vazquez, Iñigo Sedano, and Diego López de Ipiña. Evaluation of orchestrated reactivity of smart objects in Pervasive Semantic Web scenarios. In *Proceedings of The Second International Workshop on Semantic Web Technology For Ubiquitous and Mobile Applications (SWUMA'06) at the 17th European Conference of Artificial Intelligence (ECAI 2006)*, August 2006.

[22] World Wide Web Consortium. *OWL Web Ontology Language Guide*. World Wide Web Consortium, February 2004. W3C Recommendation.

[23] World Wide Web Consortium. *RDF Primer*. World Wide Web Consortium, February 2004. W3C Recommendation.

[24] World Wide Web Consortium. *SPARQL Query Language for RDF*. World Wide Web Consortium, April 2006. W3C Candidate Recommendation.