Evaluation of Orchestrated Reactivity of Smart Objects in Pervasive Semantic Web Scenarios

Juan Ignacio Vazquez and **Iñigo Sedano** and **Diego López de Ipiña**[1]

Abstract. The Ubiquitous Web concept encourages the application of existing well-known and state-of-the-art web technologies in Ubiquitous Computing scenarios. A particular interpretation of the Ubiquitous Web is the Pervasive Semantic Web, where the joint synergies of the Semantic Web model and Ubiquitous Computing technologies are integrated in order to create intelligent environments populated by smart objects.

In this paper we present a general vision and preliminary results on the evaluation of SOAM – Smart Objects Awareness and Adaptation Model –, a pure 100% Web-based environment reactivity model that uses orchestration mechanisms to coordinate existing smart objects in Pervasive Semantic Web scenarios in order to achieve automatic adaptation to user preferences.

1 INTRODUCTION

Ubiquitous Computing is aimed at creating intelligent spaces to empower users in everyday tasks at home, work, street, vehicle and so forth, aligned with the vision of Ambient Intelligence [11] [12] [13]. In this vision, environments are proactive perceiving users' surrounding information, often referred to as context, and reacting in the appropriate way to facilitate users' activities. In fact, more and more smart spaces engineers and designers are starting to think that the most valuable resource in such environments is not computing power, communication or storage capabilities, but user interaction [6] [24] [23].

Environment adaptation can be achieved from two different approaches::

1. **Active influence**: any mechanism in which the agent explicitly commands other agents or objects to change their state or perform an action. Examples of active mechanisms s are configuration processes and operation invocation techniques such as CORBA, RMI or SOAP.
2. **Passive influence**: any mechanism in which an agent disseminates certain information, expecting that other agents change their state or perform an action at their discretion to create a more adapted environment [27].

While active influence represents traditional "command and control" mechanisms, agents applying passive influence do not command the target agents or objects to do anything concrete; it simply publishes/broadcasts desired preferences expecting that the others react, changing their state in a positive way [26].

[1] MoreLab – Mobility Research Lab. University of Deusto, Avda. Universidades, 24, 48014 Bilbao, Spain. email: {ivazquez, dipina}@eside.deusto.es, isedano@tecnologico.deusto.es

For instance, by broadcasting a preference stating "my preferred ambient temperature is 24^oC", a user does not control explicitly any device, but some of latter could be user-aware and set the environment's temperature accordingly .

Passive mechanisms are not intrusive, but probably their effects are less predictable.

We have experimented with active mechanisms in pervasive computing environments in the past, for example, applying the EMI^{2}lets middleware [16], with positive results. But as any other active influence model, it requires explicit user intervention continuously to control the environment.

Passive mechanisms are complementary to active ones and can serve to automatically adapt the environment in a initial phase, while allowing users to customise it later via explicit interaction. For instance, when a user enters a car, the temperature, radio station and driving settings, could be automatically configured to his preferences/characteristics without explicit command, being the user free to change them explicitly afterwards.

Passive mechanisms have not been very much explored in pervasive computing scenarios; most of existing systems use several forms of active mechanisms, such as WebServices/SOAP (UPnP [5], Task Computing [17] [18] [22], WSAMI [10]), Jini, CORBA, and so forth.

We have designed a passive influence model called SOAM – Smart Objects Awareness and Adaptation Model – that applies Semantic Web technologies to create context-aware environments that react automatically to user preferences, technically called adaptation profiles, without explicit user intervention.

But there are still some problems that other similar architectures exhibit and we tried to solve in our model. In this paper we explain the basics of SOAM and provide initial results on the evaluation of this approach to create more intelligent and reactive environments.

2 THE PERVASIVE SEMANTIC WEB VISION

The World Wide Web Consortium is currently promoting the Ubiquitous Web vision: webs of interconnected services and resources everywhere in the real world, thus augmenting users' experiences.

As a particular approach to the Ubiquitous Web, we coined the term *Pervasive Semantic Web* to designate the result of applying Semantic Web technologies to Pervasive Computing scenarios in order to perform reasoning processes. The main representatives of those technologies are RDF (Resource Description Framework)[30] and OWL(Ontology Web Language)[29].

These scenarios are populated by different kinds of devices with a number of capabilities such as temperature sensing, video capturing, door opening, and so on. Our strategy is based on using ontologies to represent knowledge about different domains, so that we use appro-

priate ontologies for temperature, physical access control, location and so on.

In the Pervasive Semantic Web, devices are interconnected, hosting knowledge about environmental perceived conditions and using references to link resources inside and outside this space. This vision determines the creation of a new type of logical environment in ubiquitous computing scenarios: a personal area semantic web with information flows back and forth among communicating devices, sharing their knowledge about users inside the environment and coordinating their tasks via distributed reasoning procedures in order to provide an ambient intelligence experience.

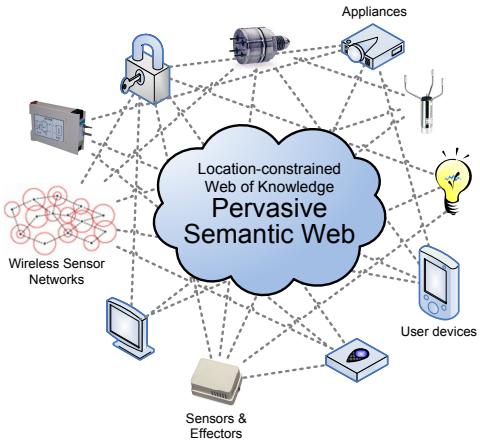

Figure 1. The Pervasive Semantic Web.

2.1 Challenges

As we mentioned before, people need to be released from the burden of interacting with the environment all the time and concentrate on their goal, not on eliminating the barriers. Surrounding objects should be able to perceive users' goal, existing barriers and perform the required operations to facilitate human activities. This is the final goal of Ambient Intelligence.

After analysing the scenarios depicted by Weiser et al. in [28] and the EU ISTAG in [11] [12] [13], as well as proposed ubiquitous computing architectures integrating semantic web technologies such as CoBrA [3] [1] [2], Gaia [19] [21] [20], Task Computing [17] [18] [22] or Semantic Spaces/SOCAM [8] [25], we have found several issues that drew our attention:

1. Our environments are getting more and more populated by electronic / automatic devices, sometimes with advanced complex interfaces to operate them. Since they are neither context-aware nor user-aware, they use to require manual operation (e.g.: PDA, home multimedia devices, mobile phones, MP3 players, light control systems, access control systems, and so forth).
2. The scenarios depicted in those visionary papers are far from being reached in real life nowadays: existing devices and environments are not very intelligent.
3. Existing experimental proposals integrate different unrelated technologies to solve a problem, making the resulting system heavier and highly coupled.
4. There is a need of a suitable discovery protocol that can take advantage of semantic web information models during discovery.

We can only expect a high degree of intelligence in the environment if such environment is populated by intelligent devices. In this type of scenario, devices are able to gather existing information, share this information with others, analyse and reason upon the data, and determine the best reactive behaviour to perform.

Of course, such environment should not be only composed of individual autonomous devices acting by themselves, but a certain level of coordination should arise, required to perform collaborative tasks in consistent ways. Some devices feature specific types of perception capabilities while other feature complementary ones; some can perform concrete operations over some set of environmental conditions while other work and act over a different set of aspects.

Specialised appliances and heterogeneous objects are the common rule (heating systems, light bulbs, temperature control systems, iris-based identification mechanisms, camera-based surveillance systems), but although concrete features are different, coordination and collaboration can be achieved among them to truly realise the concept of Ambient Intelligence.

While individual behaviours are still required, coordination boosts devices to a further degree of intelligence, greater than the sum of single capabilities. Nonetheless, coordination requires some kind of communication framework, and thus, communication capabilities among objects. Presently, only very advanced appliances have the ability to carry out any sort of communication processes with fellow devices.

Therefore, communication and intelligence have been considered two attributes at the core of the Ubiquitous Computing / Ambient Intelligence vision. Communication contributes to information sharing and coordination of activities, while intelligence contributes to analysing, reasoning and decision taking.

The Web model and Semantic Web technologies feature a series of characteristics that seem to fulfil these two major attributes.

The joint application of the Web model and the Semantic Web in pervasive computing scenarios results in a coherent architectural model, since core technologies such as URI or namespaces constitute their technological basis.

It is difficult to find a flexible mechanism for representing how devices must behave under certain situations or contexts. Existing approaches embody the behaviour in the form of small programs hosted in the device [1]. This approach exhibits a severe limitation when these programs need to be updated, as well as the behaviour of devices is not self-descriptive.

Other approaches embody the behaviour in the form of rules [19] [8], but they do not provide a way for entities to dynamically influence each other by injecting rules depending on the existing context. This ability would allow a device to influence others in its surroundings, modifying their behaviour to make them context-aware and implementing the passive influence concept. Of course, security and authorisation mechanisms should me provided as well.

On the other hand, most of the previous work involving integration of Semantic Web into Ubiquitous Computing architectures exhibit a technological inconsistency mixing non-Web mechanisms with Semantic Web technologies:

- The Context Broker Architecture (CoBrA) uses JADE API and FIPA ACL (Agent Communication Language) for information exchange.
- Gaia relies on CORBA for communication and transport of RDF payload.

- The Service Oriented Context-Aware Middleware uses JavaRMI as protocol for RDF information exchange.

Not only these approaches seem somehow artificial, but they require the use of multiple libraries and process layers, resulting in larger platform sizes, which is a major concern for limited device platform deployment. Also, potential integration problems between semantic web technologies and these communication mechanisms can arise, since they were not designed to work together.

Despite HTTP is somehow simpler and less powerful than JADE, CORBA or JavaRMI, it is a natural vehicle for communication in Ubiquitous Computing scenarios (as demonstrated in projects such as Cooltown [14] [15]), specially if Semantic Web technologies are being applied.

One of the main problems in adopting HTTP as a Pervasive Computing protocol is the lack of a suitable ubiquitous discovery mechanism. This concern is also present in the previous architectures, that end up applying other architectures' discovery protocols such as SSDP [7] or Bluetooth SDP for this goal.

But, since RDF and OWL are being used for creating rich information models, we came up with the possibility of designing a discovery protocol able to explore these models instead of performing a mere attribute-value matching.

Therefore, in our research we faced three main challenges:

- To completely embody influence over the environmental and device behaviour in the form of RDF behavioural profiles: "adaptation profiles" that can be exchanged among existing entities.
- To create a pure 100% Ubiquitous Computing architecture exclusively based on Web technologies to achieve technological consistency, and reduce both platform size and potential integration problems. HTTP security mechamisms such as basic or digest authentication could provide the security level needed for the profiles exchange.
- To create a light and suitable discovery protocol able to take advantage of the RDF representation of devices, services, people and any other kind of resource.

SOAM – Smart Object Awareness and Adaptation Model – has been designed in such as way that these requirements are fulfilled, completely based on the Web model, thus seamlessly integrating semantic web technologies, and featuring a passive influence model based on "adaptation profiles".

3 SOAM: SMART OBJECTS AWARENESS AND ADAPTATION MODEL

SOAM defines different kind of entities or agents that exchange several types of informational structures in order to coordinate its behaviour. In this section, those information structures and entities are briefly described in order to better understand the testing and evaluation processes.

3.1 Structures

Four types of information structures are needed in SOAM:

- Context Information: in SOAM, devices or smart objects perceive and gather environment information via built-in sensors or any other mechanism. This information is made available to requesting parties in a semantic annotated form using RDF/OWL, in order to allow further reasoning. Figure 2 represents an example of Context Information captured by a light sensor and made available to any entity in the environment.
- Capabilities: they inform about the perception capabilities and operation capabilities of a smart object. Perception capabilities indicate which kind of context information this object can capture, while operation capabilities indicate which kind of context information can be altered by this smart object, probably via some kind of effectors.
- Constraints: smart objects' behaviour can be influenced by other agents using some data constructions called constraints, which declare valid ranges on the desired state of the environment, so that the object is in charge of driving adaptation honouring them.
- Adaptation Profiles: in SOAM, users or other devices do not generate low-level constraints to obtain environment adaptation, but in turn they use high-level Adaptation Profiles. An Adaptation Profile is a preference or environment adaptation requirement that contains two different sections:
  - Preconditions: represent existing requirements about the environment's present state, that must be met for the Adaptation Profile to activate. It makes the adaptation to have a conditional nature. Often, adaptation requirements are not fixed, e.g. a user does not need his preferred temperature to be always 22^oC, but maybe only when he is at the car.
  - Postconditions: represent desired patterns in the environment's future state that must be met for the adaptation to be considered as honoured. Adaptation Profiles do not explicitly declare how the environment (its constituent objects) must adapt, but it just inform about the desired environment state and let smart objects decide how to meet the requirements, depending on their internal logic.

Variable substitution in Adaptation Profiles is possible to allow postcondition elements to be bounded to precondition elements as shown in figure 3.

This Adaptation Profile can be read as "*whatever the location Alice is in, if that location is a room, then that room should have a temperature of* 24^oC", which is a very simple but powerful mechanism for Alice to force every room to be aware of her preferred temperature as Alice gets in. Of course, this Adaptation Profile can only be honoured if a smart object exhibiting operation capabilities over the room temperature is present.

Adaptation Profiles are resolved against existing Context Information in order to generate Constraints that can be sent to appropriate smart objects in the environment.

3.2 Architecture

SOAM architecture is divided into three different types of entities: Smobjects, Orchestrators, and Adaptation User-Agents.

A *Smobject* (a short for "smart object") is an agent representing an intelligent device, several devices or event part of a device. Smobjects capture a subset of environmental conditions, provide that perceived Context Information under request, and act upon the same or other subset of environmental conditions in order to modify them as needed via Constraints.

The way a smobject is connected to the actual device, sensors or effectors is out of the scope of SOAM standardization, being highly platform-dependent so that designers are free to select the best choice depending on solution requirements.

```
<rdf:Description rdf:about="urn:uuid:light1">
  <lit:hasLuminance rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
    30
  </lit:hasLuminance>
  <rdf:type rdf:resource="http://www.awareit.com/onto/lighting#Light"/>
</rdf:Description>
```

Figure 2. An example Context Information.

```
<adaptationProfile id="urn:uuid:prof1" expires="PT10M">
  <variable id="x"/>
  <precondition subject="urn:uuid:alice"
    predicate="http://www.awareit.com/onto/location#isLocatedIn" >
    <objectVariable ref="x"/>
  </precondition>
  <precondition subject="x"
    predicate="http://www.w3.org/1999/02/22-rdf-syntax-ns#type">
    <objectResource ref="http://www.awareit.com/onto/location#Room"/>
  </precondition>
  <postcondition subject="x"
    predicate="http://www.awareit.com/onto/temperature#hasTemperature">
    <objectLiteral datatype="http://www.w3.org/2001/XMLSchema#int">
      24
    </objectLiteral>
  </postcondition>
</adaptationProfile>
```

Figure 3. An example Adaptation Profile with one bounded variable.

Smobjects also act as control agents for the device or devices. They need to access built-in sensors, effectors, communication ports, maybe storage facilities if available on the device, and so forth.

An *Orchestrator* is an agent that perceives and coordinates existing smobjects in the environment to perform the adaptation process following an Adaptation Profile. Orchestrators transform Adaptation Profiles into concrete Constraints that are sent to appropriate smobjects to drive their behavior as explained below.

An *Adaptation User-Agent* is a piece of software, acting on behalf of a user or other client device, that stores the owner's Adaptation Profiles and negotiates with surrounding Orchestrators the adaptation process by exchanging those profiles.

Adaptation User-Agents silently start the process of adapting the environment by finding an available Orchestrator to which they send the Adaptation Profiles.

The process of environment adaptation is described in several phases with communication flows illustrated in Figure 4.

1. Smobjects discovery and Capabilities retrieval: the environment Orchestrator periodically scans the environment for smobjects (via SSDP [7] multicast protocol or mRDP in the future as explained below) and retrieving their Capabilities using HTTP. The overall sum of individual Capabilities of the existing smobjects form somehow the Capabilities of the environment.
2. Orchestrator discovery: a user shows up in the environment powered with an Adaptation User-Agent that, on behalf of the user, looks up for an existing Orchestrator using SSDP.
3. Environment capabilities retrieval: the Adaptation User-Agent communicates with the Orchestrator and retrieves the Capabilities of the environment (the overall set of existing smobjects' Capabilities).
4. Adaptation Profiles injection: the Adaptation User-Agent selects the suitable Adaptation Profiles that can be processed by the environment and sends them to the Orchestrator.
5. Reasoning: periodically, the Orchestrator retrieves Context Information from existing smobjects, augmenting it by performing reasoning with description logics and stored inference rules. After that, the Orchestrator checks if Adaptation Profiles' preconditions are matched against the augmented Context Information. When this happens, postconditions are resolved and concrete Constraints are finally generated.
6. Constraints processing: the Orchestrator sends the obtained Constraints to suitable smobjects, based on declared Capabilities. A smobject processes the Constraints message and establishes the internal mechanisms for the desired behaviour.
7. Smobject adaptation: eventually, the smobject expresses the new behaviour, e.g. increasing the temperature, changing the TV channel, and so on, via effectors or connected devices.
8. Environment adaptation: from a higher viewpoint, individual adaptation of smobjects creates an emerging coordinated environment adaptation honouring user's Adaptation Profiles.

The overall result of applying SOAM is a context-aware reactive environment, where smart objects are orchestrated to meet user's adaptation requirements.

4 IMPLEMENTATION AND EVALUATION

For testing purposes, we have implemented a prototype of SOAM with the following characteristics:

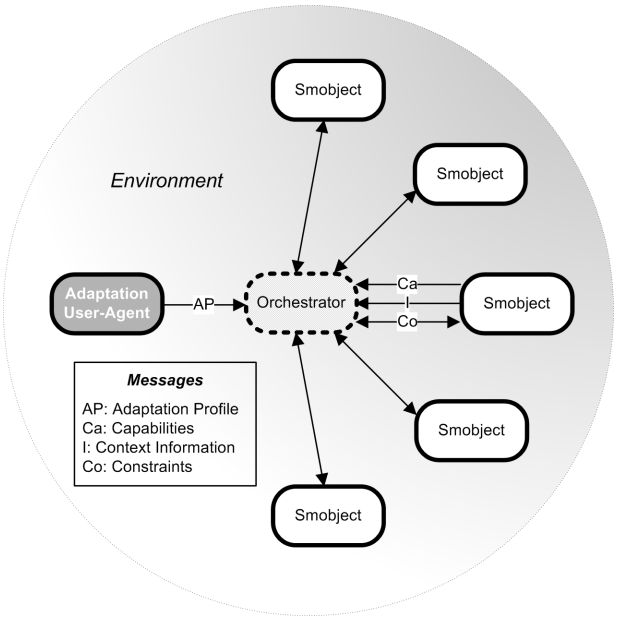

Figure 4. Diagram illustrating SOAM architecture.

- Smobjects: embedded platforms based on AMTEL ARM7 with 55 MHz of processor speed and 16 Mb RAM running μClinux. The Smobject middleware was implemented in Java and runs on top of Mika (a Java VM for embedded platforms).
- Orchestrator: a platform based on Intel Pentium-M 1.86 GHz with 1Gb RAM, running Windows XP. The Orchestrator middleware was implemented in Java using Cybergarage libraries for UPnP communication and Jena libraries for Semantic Web reasoning processes.

In order to test the execution and performance of the implementation, we created an hypothetical scenario emulating a smart living room and featuring the following devices: a Hi-Fi system, a TV set, an ambient microphone, a laptop, a temperature control system, and a light.

The scenario was completed with a test user called Bob featuring four adaptation profiles:

- AdaptationProfile1: If Bob is located in *x*, then *x* must have a temperature of 24^{o}C.
- AdaptationProfile2: If Bob is working with the laptop in *x*, and *x* has an ambient sound *y*, then *y* must be classical music with a volume level of 3.
- AdaptationProfile3: If Bob is talking in *x*, and *x* has an ambient sound of *y*, then *y* must have a volume level of 0.
- AdaptationProfile4: If Bob is watching TV in *x*, and *x* has an ambient light *y*, then *y* must illuminate *x* with yellow colour and have a luminance level of 10.

As Bob entered the room, his adaptation profiles were injected into the Orchestrator, which discovered the smobjects and retrieved its capabilities during the initialization phase. Given an arbitrary initial state of those smobjects, the Orchestrator retrieved the Context Information and reasoned over it using description logics and knowledge rules. The resulting enriched information was checked against Bob' Adaptation Profiles in order to generate associated Constraints that were sent to appropriate smobjects.

For instance, AdaptationProfile1 was activated at all times and the constraint requesting a temperature of 24^{o}C in room1 was sent to the temperature control system smobject. When Bob logged in the laptop, AdaptationProfile2 was activated, generating constraints for the Hi-Fi system smobject, requesting a classical music background sound with a volume level of 3.

Figure 5 illustrates the measured times for different phases (smobjects discovery and capabilities retrieval, context information retrieval, reasoning at the orchestrator and constraints processing at the smobject) that were performed during 16 different tests, obtaining the average times shown in Figure 6.

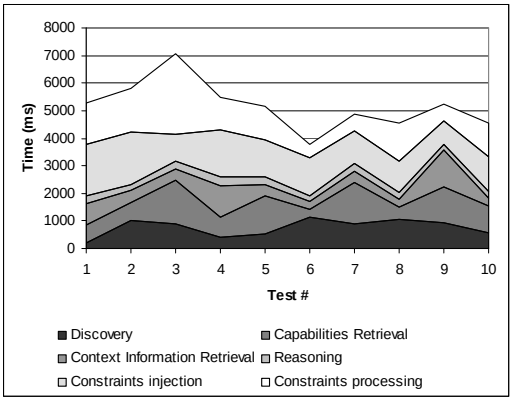

Figure 5. Measured times for the different SOAM phases.

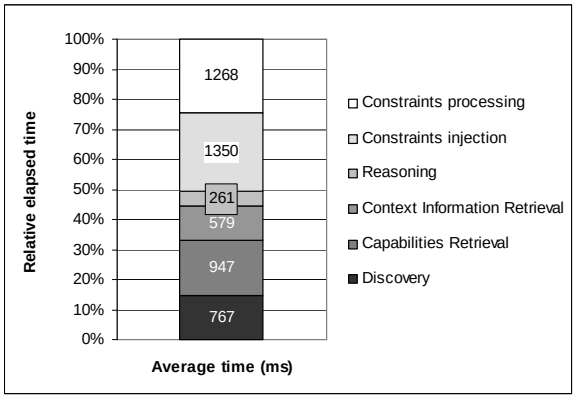

Figure 6. Average measures and relative elapsed time for the different SOAM phases.

As we can notice the most costly phases are Constraints injection and processing at the smobject, due probably to platform limitations, despite an ARM7-based platform running at 55 MHz seems to be quite average for any embedded device.

On the other hand, reasoning at the Orchestrator is surprisingly efficient, probably due to platform features and the absence of com-

munication latencies during this phase (communication is not needed during reasoning).

Considering that smobjects discovery, Capabilities retrieval and Context Information retrieval are performed by the Orchestrator in the background continuously, the average time for environment adaptation requests (reasoning + Constraints injection and processing) in our tests is about 2.87 seconds. Our conclusion is that when Bob enters the room and exposes his adaptation profiles, existing devices can react appropriately in a minimum time of about 2.87 seconds, which is a quite promptly environment response. Maximum response time depends on particular device operation, e.g. maybe 10 minutes are required for the heating system to achieve the desired temperature in a big room.

In most cases, an environment is populated by several users at a time, e.g. intelligent meeting rooms, public transport vehicles, shops, and so forth. These users could broadcast their Adaptation Profiles to the existing environment Orchestrator in order to obtain a common personalized ambient. In case of conflict, the Orchestrator could decide which constraints to apply, depending on priorities or other available user information.

When many users disseminate their Adaptation Profiles throughout a concrete environment the Orchestrator performance can be severely affected. In order to determine how the amount of Adaptation Profiles is related to the performance of the reasoning process for obtaining the constraints, we carried out several tests, where the Orchestrator was provided with 4, 8, 16, 32, 64, 128 and 256 Adaptation Profiles. Obtained results are shown in Figure 7.

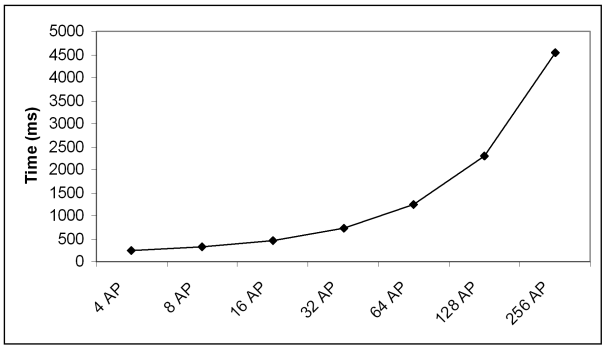

Figure 7. Orchestrator performance during reasoning with different amounts of Adaptation Profiles.

As we can see, the Orchestrator performance is barely affected with 4, 8, 16 and even 32 Adaptation Profiles, while from that point ahead, the response time of the reasoning process almost doubles.

Initial estimation of the amount of Adaptation Profiles an environment must support is difficult. Considering that every user can disseminate an average of 5 Adaptation Profiles, an Orchestrator similar to the one we tested could support about 50 users, featuring a reasoning response time under 5 seconds.

We think that those response times can be improved via optimization during evolution of our implementation. This improvement becomes a must when other less powerful platforms are intended to host the orchestration process, as we are planning.

5 mRDP: MULTICAST RESOURCE DISCOVERY PROTOCOL

The earlier versions of the SOAM platform, as many other architectures, used SSDP for discovery. But as we stated earlier, in order to fully take advantage of the potential of RDF during discovery, a new protocol should be designed.

From our point of view such a protocol must meet the following requirements:

- Able to query advanced devices (such as orchestrators) representing their internal state / managed information as RDF models. Those models can be queried through languages such as SPARQL, whose protocol uses HTTP interfaces.
- Able to cope with limited devices lacking RDF information models. The desired protocol must be also able to deal with simpler, non SPARQL-based queries.
- Since a centralised device or entity registry is not suitable for pervasive computing scenarios, the protocol must feature multicast communication.

Considering these requirements we designed a multicast protocol over UDP, able to convey SPARQL queries as well as more simpler attribute-value queries in a basic XML-based language called TiRDL (Tiny Resource Discovery query Language).

The resulting protocol is mRDP (Multicast Resource Discovery Protocol). mRDP conveys SPARQL or TiRDL queries in UDP multicast request packets. Receiving entities process the query and inject the results into a requester provided callback URI using HTTP.

mRDP features two different commands:

- IDENTIFY: for resolving queries against existing RDF models in the environment and obtaining the URI of the resource satisfying the query.
- LOCATE: for locating SPARQL or HTTP endpoints where information about a concrete resource can be provided.

SPARQL queries make mRDP extremely powerful: any mRDP client application can search in the vicinity for any kind of resource, including devices, services, documents, people information and so on.

The steps involved in the mRDP discovery process are illustrated in the following example:

1. A PDA application wants to display data about the owners of surrounding devices.
2. The PDA application creates the SPARQL request using the appropriate vocabularies and ontologies (e.g.: `SELECT ?owner WHERE (?owner <po:owns> ?resource . ?resource <rdf:type> <do:Device>.)`).
3. The request is multicasted through the wireless network in a mRDP `IDENTIFY` packet that includes a callback URI where the application listens for responses.
4. Devices and appliances receive the request packet, execute the query against their RDF information model, and, if matched, they respond with an HTTP request to the callback URI conveying the values for `?owner` as well as SPARQL and / or HTTP URLs where more information about that resource can be downloaded.
5. The PDA application collects all the possible values for `?owner` and retrieves the information from the supplied SPARQL or HTTP URLs. Let us suppose that one of those values is the URI of *Bob*, `http://people.com/bobby`.

6. The PDA must retrieve all the possible information about the different values of `?owner`, not just that supplied by those devices that processed the query. For instance, a device not owned by *Bob*, may contain available information about him. So, the PDA application multicasts a `LOCATE` command about `http://people.com/bobby`.
7. All devices containing information about `http://people.com/bobby`, respond with SPARQL or HTTP interfaces where that information can be downloaded from.
8. The PDA application downloads all the information about *Bob* from the available surrounding sources, displaying it on the screen.

The mRDP protocol itself is not complex at all and it represents a kind of "multicast SPARQL protocol". The mRDP client is very light and can store predefined SPARQL queries as strings, without any need for understanding the underlying syntax. The mRDP server takes advantage of an integrated SPARQL engine for executing the query and retrieving the results. Simpler queries, non SPARQL-based, are also allowed via TiRDL for implementing mRDP servers in more limited devices.

6 CONCLUSIONS AND FUTURE WORK

SOAM is an effort to create a passive influence model for automatic environment adaptation to required conditions. SOAM applies Semantic Web technologies for knowledge representation and reasoning (vocabularies and ontologies) and it can take advantage of existing standard ontologies, such as SOUPA[4], or CONON [9] for this purpose.

We think that SOAM has some advantages over other alternatives, such as its novel approach based on passive influence through exchange of "adaptation profiles", its technological consistency relying completely on web technologies and the capability and semantic representation both of context information and the adaptation requirements.

Moreover, mRDP constitutes an innovative approach for resource discovery in ambient intelligence environments, it is also strongly based on web and semantic web technologies, and it will be incorporated into the architecture.

SOAM has been implemented and tested, obtaining some interesting results about how distributed smart objects can be orchestrated to achieve coordinated adaptation of the environment via passive influence. To the extent of our knowledge other alternatives lack of such testing and concrete results.

There are some open research issues related to SOAM architecture and passive influence models that still need to be investigated.

For dealing with conflict resolution when incompatible adaptation profiles are injected, devices can be preconfigured with several strategies depending on the particular device and situation: the first profile wins, the last wins, find a balanced value, privileges, and so forth.

The centralised topology is one of the weakest points in all the ubiquitous computing architectures that integrate semantic web technologies: the requirement of a central reasoner managing all the context information impose some deployment requirements that are most of times unacceptable in this kind of scenarios.

One of the future challenges of SOAM is to solve this problem through a distributed choreographical model, without a central entity, meeting much better the requirements of real pervasive computing environments.

ACKNOWLEDGEMENTS

This work has been partially supported by the Department of Industry, Commerce and Tourism of the Basque Government under a SAIOTEK grant, and the Cathedra of Telefonica Moviles at the University of Deusto, Bilbao, Spain.

REFERENCES

[1] Harry Chen, Tim Finin, and Anupam Joshi, 'A context broker for building smart meeting rooms', in *Proceedings of the Knowledge Representation and Ontology for Autonomous Systems Symposium, 2004 AAAI Spring Symposium*, eds., Craig Schlenoff and Michael Uschold, pp. 53–60, Stanford, California, (March 2004). AAAI, AAAI Press, Menlo Park, CA.

[2] Harry Chen, Tim Finin, and Anupam Joshi, 'Semantic web in in the context broker architecture', in *Proceedings of the Second Annual IEEE International Conference on Pervasive Computer and Communications*. IEEE Computer Society, (March 2004).

[3] Harry Chen, Tim Finin, and Anupam Joshi, *Ontologies for Agents: Theory and Experiences*, chapter The SOUPA Ontology for Pervasive Computing, Whitestein Series in Software Agent Technologies, Springer, July 2005.

[4] Harry Chen, Filip Perich, Tim Finin, and Anupam Joshi, 'Soupa: Standard ontology for ubiquitous and pervasive applications', in *First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous'04)*, Boston, MA, (August 2004).

[5] UPnP Forum, *UPnP Device Architecture v.1.0.1 Draft*, 2003.

[6] David Garlan, Dan Siewiorek, Asim Smailagic, and Peter Steenkiste, 'Project aura: Toward distraction-free pervasive computing', *IEEE Pervasive Computing*, **1**(2), 22–31, (2002).

[7] Yaron Y. Goland, Ting Cai, et al., *Simple Service Discovery Protocol/1.0. Operating without an Arbiter*, 1999. Internet Draft.

[8] Tao Gu, Hung Keng Pung, and Da Qing Zhang, 'A service-oriented middleware for building context-aware services', *Journal of Network and Computer Applications*, **28**(1), 1–18, (2005).

[9] Tao Gu, Xiao Hang Wang, Hung Keng Pung, and Da Qing Zhang, 'An ontology-based context model in intelligent environments', in *Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference*, (2004).

[10] Valerie Issarny, Daniele Sacchetti, Ferda Tartanoglu, Francoise Sailhan, Rafik Chibout, Nicole Levy, and Angel Talamona, 'Developing ambient intelligence systems: A solution based on web services', *Automated Software Engineering*, **12**(1), 101–137, (2005).

[11] European Commission IST Advisory Group (ISTAG), 'Scenarios for ambient intelligence in 2010', Technical report, EU Commission, (2001).

[12] European Commission IST Advisory Group (ISTAG), 'Ambient intelligence: from vision to reality', Technical report, EU Commission, (2003).

[13] European Commission IST Advisory Group (ISTAG), 'IST research content', Technical report, EU Commission, (2003).

[14] Tim Kindberg and John Barton, 'A web-based nomadic computing system', *Computer Networks: The International Journal of Computer and Telecommunications Networking*, **35**(4), 443–456, (2001).

[15] Tim Kindberg, John Barton, Jeff Morgan, Gene Becker, Debbie Caswell, Philippe Debaty, Gita Gopal, Marcos Frid, Venky Krishnan, Howard Morris, John Schettino, Bill Serra, and Mirjana Spasojevic, 'People, places, things: web presence for the real world', *Mobile Networks and Applications*, **7**(5), 365–376, (2002).

[16] Diego López de Ipiña, Juan Ignacio Vazquez, Daniel Garcia, Javier Fernandez, and Ivan Garcia, 'A reflective middleware for controlling smart objects from mobile devices', in *Proceedings of sOc-EUSAI 2005: Smart Objects & Ambient Intelligence*, (2005).

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

[18] Ryusuke Masuoka, Bijan Parsia, and Yannis Labrou, 'Task computing - the semantic web meets pervasive computing', in *Proceedings of 2nd International Semantic Web Conference (ISWC2003)*, Sanibel Island, Florida, (October 2003).

[19] Robert E. McGrath, Anand Ranganathan, M. Dennis Mickunas, and Roy H. Campbell, 'Investigations of semantic interoperability in ubiq-

uitous computing environments', in *Proceedings of the 15th International Conference Parallel and Distributed Computing and Systems (PDCS)*, (2003).

[20] Anand Ranganathan, *A Task Execution Framework for Autonomic Ubiquitous Computing*, Ph.D. dissertation, Department of Computer Science, University of Illinois at Urbana-Champaign, 2005.

[21] Anand Ranganathan, Jalal Al-Muhtadi, and Roy H. Campbell, 'Reasoning about uncertain contexts in pervasive computing environments', *IEEE Pervasive Computing*, **03**(2), 62–70, (2004).

[22] Evren Sirin, James Hendler, and Bijan Parsia, 'Semi-automatic composition of web services using semantic descriptions', in *Web Services: Modeling, Architecture and Infrastructure workshop in ICEIS 2003*, Angers, France, (April 2003).

[23] Joo Sousa and David Garlan, *The Aura Software Architecture: an Infrastructure for Ubiquitous Computing*, 2003. Carnegie Mellon Technical Report, CMU-CS-03-183.

[24] Joo Pedro Sousa and David Garlan, 'Aura: an architectural framework for user mobility in ubiquitous computing environments', in *Proceedings of WICAS3: the IFIP 17th World Computer Congress - TC2 Stream / 3rd IEEE/IFIP Conference on Software Architecture*, pp. 29–43, Deventer, The Netherlands, The Netherlands, (2002). Kluwer, B.V.

[25] Joo Geok Tan, Daqing Zhang, Xiaohang Wang, and Heng Seng Cheng, 'Enhancing semantic spaces with event-driven context interpretation', in *Proceedings of Pervasive 2005: Third International Conference on Pervasive Computing*, volume 3468 of *Lecture Notes in Computer Science*, pp. 80–97. Springer, (2005).

[26] Juan Ignacio Vazquez and Diego López de Ipiña, 'A language for expressing user-context preferences in the web', in *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*, pp. 904–905, New York, NY, USA, (2005). ACM Press.

[27] Juan Ignacio Vazquez and Diego López de Ipiña, 'An interaction model for passively influencing the environment', in *Adjunct Proceedings of the 2nd European Symposium on Ambient Intelligence*, (2004).

[28] Mark Weiser, 'The computer for the 21st century', *SIGMOBILE Mobile Computing and Communications Review*, **3**(3), 3–11, (1999).

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

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