SoaM: A Web-powered Architecture for Designing and Deploying Pervasive Semantic Devices

JUAN IGNACIO VAZQUEZ, DIEGO LÓPEZ DE IPIÑA
{*ivazquez,dipina*}*@eside.deusto.es*
MoreLab - Mobility Research Lab, University of Deusto, Avda. Universidades 24, 48007 Bilbao, Spain

IÑIGO SEDANO
isedano@tecnologico.deusto.es
MoreLab - Mobility Research Lab, Tecnológico Fundación Deusto, Avda. Universidades 24, 48007 Bilbao, Spain

Received: January XX 2005; revised: November XX 2005

***Abstract*— Despite several efforts during the last years, the web model and semantic web technologies have not yet been successfully applied to empower Ubiquitous Computing architectures in order to create knowledge-rich environments populated by interconnected smart devices. In this paper we point out some problems of these previous initiatives and introduce SoaM (Smart Objects Awareness and Adaptation Model), an architecture for designing and seamlessly deploying web-powered context-aware semantic gadgets. Implementation and evaluation details of SoaM are also provided in order to identify future research challenges.**

***Index Terms*— Ubiquitous computing, semantic web, semantic devices, ubiquitous web**

I. INTRODUCTION

Ubiquitous Computing is a major field of research nowadays as surrounding environments are becoming more and more populated by small embedded devices and appliances. The term "Ubiquitous Computing" was coined by Mark Weiser in his seminal article "The Computer for the 21st Century" [45], but other terms such as pervasive computing, invisible computing or calm computing have become increasingly popular. Recently, Ambient Intelligence (AmI) has become a widely overused term, mostly within Europe, as it has been one of the main ICT research focuses in the 6th Framework Program [24] [25].

Ubiquitous computing devices are expected to be autonomous and behave in the appropriate way depending on particular situations. Therefore, they must be able to collect relevant data from any available sources in the environment, analyze this information, decide which reactive behavior is required and finally they must carry out the required operations. A device able to fulfill all this process is known to be context-aware [1].

Lassila et al. also introduced the vision of *semantic gadgets* [28]: autonomous devices that intelligently interpret context information, inferring new knowledge from existing one in order to understand the situation and react appropriately. Designing a feasible architecture for implementing semantic gadgets is certainly a challenging issue in current research that involves several areas ranging from artificial intelligence to ubiquitous communication.

Obviously, current devices and appliances do not yet exhibit the high levels of reactivity and context-awareness described in the scenarios proposed by Lassila, Weiser [45] or ISTAG [23] [24] [25], but simpler and more limited forms of behavior: automatic doors that open in the presence of people, sinks that provide water as hands are placed nearby, lights that turn on autonomously when someone gets closer, digital cameras that detect nearby printers to generate photo printouts or mobile phones that perceive surrounding partners to play a game.

Their behavior is still very basic, normally bounded in a mechanical way to particular built-in sensors and actuators, and cannot be modified easily to make devices aware of other situations. Moreover, environmental intelligence, as defined by AmI [23], should be achieved by the resulting synergies from coordinated smart objects. However, few of current devices are able to communicate with each other, and even less to collaborate in an intelligent way.

Specialized non-cooperative artifacts are the common rule (heating systems, light bulbs, temperature control systems, iris identification mechanisms, camera-based surveillance systems). While these individual behaviors are still required, rich collaboration among devices boosts the environment to a further degree of intelligence, greater than the sum of single capabilities. Nonetheless, collaboration 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 objects.

This is the motivation for the great deal of research effort devoted during the last years to make pervasive devices smarter, more reactive and more collaborative. Some devices feature specific types of perception capabilities while other feature complementary ones; some of them can perform concrete operations in particular set of environmental conditions

while other work and act over a different set of aspects.

Therefore, communication and intelligence are considered two attributes at the core of the Ubiquitous Computing / Ambient Intelligence vision. Communication contributes to information sharing and coordination of activities; intelligence contributes to analyzing, reasoning and decision taking; both together contribute to device inter-collaboration for the user's sake.

The web architecture and Semantic Web technologies seem to be one of the most suitable candidates to provide these communication and intelligent capabilities in Ubiquitous Computing environments.

Regarding the communication area, it is clear that the web model based on HTTP [15] as transport protocol, and IRIs [14] or URIs [5] for resource identification and location, is considered one of the most successful communication technologies of the last decade.

But on the other hand, how the web model can provide intelligence to pervasive computing environments is not so obvious. However, during the last years the whole web model has undergone an evolution towards a new paradigm called "the Semantic Web" [4]. The basics of the Semantic Web were outlined in the Scientific American article by Berners-Lee et al. [6].

The Semantic Web is a web of knowledge, where concepts and information are represented in a machine readable and understandable form and liked 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.

However, deploying mini-webs of knowledge, sustained by a population of embedded resource-constrained devices, is undoubtedly a challenging task.

In this paper we analyze previous efforts in trying to apply the web architecture and semantic web technologies to Ubiquitous Computing scenarios, emphasizing some features that prevented them from being truly successful. In section III and IV we introduce the Pervasive Semantic Web concept and SoaM, an architecture focused in exclusively using web technologies for empowering smart devices. Sections V and VI are devoted to implementation details and evaluation results. Finally, future research lines along with conclusions are provided.

II. Previous work

One of the pioneer projects in thoroughly applying web technologies to mobile and pervasive computing was HP's Cooltown [26] [27] [2]. Cooltown tied web resources to physical resources such as objects and places, introducing the novel concept of URL sensing as a way for seamlessly retrieving endpoints URL from the surrounding environment. It also featured ways not only for collecting data but also for posting data to existing objects, thus augmenting the interaction possibilities.

The work of Issarny et al. in WSAMI (Web Services for Ambient Intelligence) [22] introduced the application of XML web services into Ambient Intelligence environments. WSAMI promoted situation-sensitive composition of services aided by a specialized middleware component hosted in mobile devices.

Undoubtedly, the most deployed web-based architecture for devices has been Universal Plug and Play (UPnP). UPnP enables the creation of pervasive peer-to-peer connectivity of computers, devices and appliances, mainly at home environments. UPnP is strongly based on web technologies (HTTP and XML), with open specifications distributed by the UPnP Forum. The UPnP architecture [40] provides mechanisms for discovering new devices and disconnections, retrieving devices' characteristics, invoking functions and sending notifications about detected events.

Among the main advantages of UPnP are its simplicity to create an inter-device communication distributed system based on web technologies, and the evidence that it is probably the most popular architecture in the market in terms of manufactured devices and commercial success, specially Internet gateways. Among its disadvantages the most important ones are the lack of security mechanisms and inherent limitations in terms of scalability as the number of devices increases [17] [32].

Moreover, UPnP does not provide any intelligence at the device side, since it is based on traditional command and control mechanisms, where the user has complete control over devices, driving manually their behavior. There are no elements in the UPnP architecture hosting any kind of reasoning processes; UPnP devices and control points are unable to understand the information they use and, of course, no form of semantic discovery or processing is performed [33].

Regarding context-awareness, UPnP does not provide any reactivity mechanism at devices or control points. Reactive behavior must be programmed ad hoc using the platform facilities, although UPnP messages can be used both for retrieving information or invoking remote operations.

CoBrA (Context Broker Architecture) [7] [9] and SOUPA (Standard Ontology for Ubiquitous and Pervasive Applications) [8] [10] [11] [12] are part of the same effort to create a context-aware pervasive system applying Semantic Web technologies.

While CoBrA is a centralized architecture, based on a contex-broker that collects all the information from participating agents using the JADE API [3], SOUPA is a OWL ontology for defining reusable basic vocabularies and relationships for Ubiquitous Computing environments.

However, there are several drawbacks in the CoBrA model for realizing the Ambient Intelligence vision. Its centralized architecture unburden limited-devices from the inconveniences of managing knowledge and information sources, but relies in a core element, the broker, that is practically the sole responsible for implementing the full architecture. It is necessary to place a broker in every location that needs to be activated with CoBrA, increasing the cost, complexity and maintenance of the deployment process.

Moreover, CoBrA does not provide any implicit reactivity mechanism; ad hoc behaviors must be explicitly programmed and configured in the central broker. Devices are again relegated to a second class passive role in the process of taking decisions. On the other hand, SOUPA is a brilliant effort to

create a modular set of common ontologies to be applied in every pervasive computing environment.

The Gaia project [37] also applies Semantic Web technologies to create a system software infrastructure for supporting Active Spaces in ubiquitous computing environments. An active space is a "*model that maps the abstract perception of a physical space as a computing system, into a first class software entity*" [35].

Thus, the active space acts as a mapping between the real and virtual space, connecting both in such a way that real world actions affect virtual world objects and vice versa. The active space hides the complexity of the real world elements into one unique entity that provides functions for manipulating the space, discover and locate internal entities, store and retrieve information from the space and so forth.

In this way, Gaia relieves the burden of manipulating highly object-populated environments by providing a single consistent interface instead of a bulk of multiple ones [36], simplifying application programming and user interaction.

However, as CoBrA, Gaia requires a central architectural element to be deployed in the environment, thus reducing its pervasiveness and not being suitable for implementation in embedded devices. Moreover, Gaia uses CORBA as distributed computing architecture, instead of HTTP and URIs, which are more appropriate considering the application of semantic web technologies such as OWL and RDF.

The initiatives Semantic Space [44] [39] and SOCAM (Service Oriented Context-Aware Middleware) [18] [19] [20] are efforts to create a smart environment infrastructure using semantic web technologies.

They share a number of similarities with CoBrA and Gaia, also requiring a centralized control component to be deployed in the environment. SOCAM uses JavaRMI for RDF information exchange, instead of HTTP.

Finally, Task Computing [30] promotes semantic Service Oriented Architectures by providing dynamic service discovery, service publishing and management, task creation and execution on the fly [38]. It even assists users in discovering what their goals are by suggesting possible tasks that can be performed with available facilities.

All these features try to solve the frustration of users in application-rich environments, where they have to orchestrate a variety of devices and applications, in order to let them concentrate only in the final goal and accomplish it with a reduced number of simple interactions.

Task Computing pioneered the application of Semantic Web Services in Ubiquitous Computing scenarios, using semantic information to annotate service descriptions and perform service composition. However, neither reasoning nor domain ontologies are provided to understand context information. Moreover, the Task Computing Environment must be always executed in the user's computer orchestrating the process; devices are not first-class actors in this architecture, but just passive entities without autonomy or intelligence.

A. Common problems

There are several remarkable features in the above referred architectures that prevent them from successfully combining all the power of web technologies with Ubiquitous Computing:

- The architectures that are enough simple to be hosted in devices (e.g. UPnP) lack of the intelligent capabilities provided by Semantic Web technologies.
- The architectures that intensively apply Semantic Web technologies for reasoning (e.g. CoBrA, Gaia, SOCAM) do not involve devices in the process, dismissing them to a secondary, often passive, role.
- These architectures also rely on a central component that must be deployed and configured beforehand for each particular scenario, which greatly reduces the serendipitous and spontaneous nature intrinsic to Ubiquitous Computing / Ambient Intelligence visions.
- Most of these architectures do not use the web communication model, essentially HTTP, for supporting all the information exchange among participating entities. Other mechanisms (e.g. CORBA, JavaRMI) are used instead, not being as suitable for dealing with RDF and OWL-based descriptions as HTTP and accompanying technologies (HTTP authentication, HTTPS, cookies, proxies, and so forth).
- None of the above architectures focuses on devices as first-class actors in the environment with autonomy, contex-awareness and reactiveness.

Basically, all the previous initiatives substitute device collaboration for central coordination. However, Ubiquitous Computing is serendipitous in essence and decentralization is a must. The challenge that researchers are facing when applying Semantic Web technologies to Ubiquitous Computing is how to make devices web-powered, more collaborative and still enough simple.

III. The Pervasive Semantic Web

As we mentioned earlier, the web model and Semantic Web technologies feature a series of characteristics that seem to fulfill two major attributes (communication and intelligence) identified in the introduction.

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.

The web communication model, based on a network of resources linked via URIs and the HTTP communication protocol [15], has been widely employed in the past and its validity has been unquestionably proven.

Moreover, existing HTTP complementary mechanisms such as cookies [29], Basic or Digest Authentication [16], or HTTPS [34] can be also reused in the pervasive computing arena to achieve session information persistence, authentication or secure communication channels, respectively. In this way, Ubiquitous Computing architectures can take advantage of existing HTTP-related technologies to fulfill a great amount of communication requirements.

The only issue not covered by HTTP is distributed discovery of devices or services. UPnP [40] proposes SSDP, an HTTP syntax-based alternative.

On the other hand, Semantic Web technologies are a suitable candidate both for context representation and reasoning. Domain specific OWL vocabularies can help defining the terms used in particular situations in order to represent the existing knowledge in a concrete moment of time.

Not only an endless number of vocabularies can be created for representing the context information about multiple knowledge domains, but reuse of existing vocabularies must be promoted in order to share same concepts among applications and devices.

Artifacts with built-in sensors can retrieve raw data from the environment and annotate those data applying a concrete vocabulary they have been explicitly configured for when manufactured.

We deem feasible to build small "annotation processes" even in limited devices in order to characterize captured raw context data in this way and create a more expressive level of knowledge that can be shared and analyzed with other devices or entities.

Moreover, the Semantic Web not only provides a mechanism for context information representation but also for reasoning. OWL is an example of a Semantic Web technology that can be used to represent description logics formalisms in such a way that new information (new context information) can be automatically generated from existing one by applying OWL intrinsic reasoning mechanisms.

Emerging Semantic Web rules technologies such as SWRL (Semantic Web Rules Language) [21] can be also applied to generate new context information based on Horn-like clauses instead of description logics.

Using SWRL it is possible to describe a rule such as *if the color of* `light1` *is yellow and the ambient sound's volume is low then the environment is suitable for reading*, which generates new information about the suitability of the environment for certain task based on previously existing knowledge. Thus, a concrete device (for example, an electronic ink book) can behave differently depending on whether the environment is suitable for reading, taking advantage of the results obtained by the reasoning process.

Our conclusion is that the application of the web model as communication infrastructure and Semantic Web technologies for context representation and reasoning seems to provide a consistent framework for creating context-aware devices and environments.

We coined the term *Pervasive Semantic Web* for designating this ubiquitous information model [42].

The vision of the Pervasive Semantic Web pursues to create a space of knowledge, where devices are interconnected, exchanging information about environmental perceived conditions and using URIs to link resources inside and outside this space [41].

This vision determines the creation of a new type of logical environment in ubiquitous computing scenarios: a location-constrained semantic web architecture with information flows back and forth among communicating devices, sharing their knowledge about the environment and coordinating their tasks via distributed reasoning processes in order to provide an intelligent immersive experience.

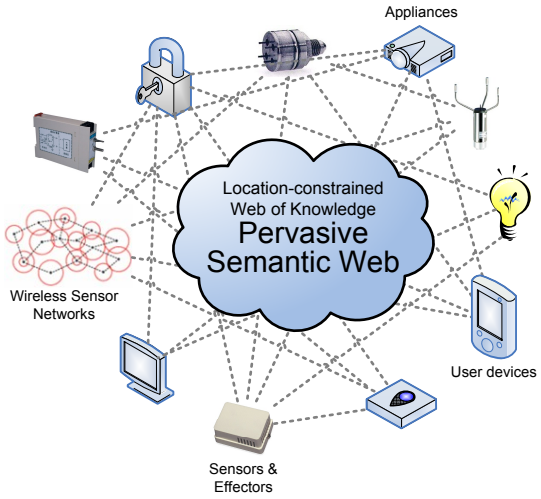

Fig. 1. The Pervasive Semantic Web.

IV. SoaM Smart Objects Awareness and Adaptation Model

Our efforts to solve the problems identified in section II by applying the Pervasive Semantic Web model produced SoaM (Smart Objects Awareness and Adaptation Model). In SoaM different kinds of entities or agents were defined in order to exchange several types of informational structures to coordinate their context-aware behavior. In this section, these information structures and entities are briefly described in order to better understand the testing and evaluation processes.

Four types of information structures were defined:

- Context Information: in SoaM devices or smart objects perceive and gather environment information via built-in sensors or similar mechanisms. This information is made available to requesting parties in a semantic annotated form using RDF/XML and OWL vocabularies, in order to promote further reasoning. The following example represents a piece of Context Information captured by a light sensor and made available to any entity in the environment.

```
<rdf:Description rdf:about="urn:uuid:light1">
  <lit:hasLuminance rdf:datatype="http://www.w3.
      org/2001/XMLSchema#int">
    30
  </lit:hasLuminance>
  <lit:hasColor rdf:resource="http://www.awareit
      .com/onto/2005/12/colors#Yellow"/>
  <rdf:type rdf:resource="http://www.awareit.com
      /onto/2005/12/light#Light"/>
</rdf:Description>
```

- Capabilities: they inform about the perception and operation capabilities of a smart object. Perception capabilities indicate the kinds of context information this object can capture, while operation capabilities indicate the types of context information that can be altered by this smart object, probably via some kind of built-in effectors.
- Constraints: smart objects' behavior 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 honoring them.

- Adaptation / Behavioral Profiles: in SoaM users or other devices do not generate low-level constraints to influence devices, but they use high-level Behavioral Profiles (also called Adaptation Profiles). A Behavioral Profile declares a desired context-aware reactivity in the device using a rule-like form containing two different sections:
  - Preconditions represent existing requirements about the environment's present state that must be met for the Behavioral Profile to be activated. They make reactivity to have a conditional nature, e.g. "my preferred temperature is 22^oC *when I am at home after 7 pm*".
  - Postconditions represent desired patterns in the environment's future state that must be accomplished by the device in order to honor the profile. Behavioral Profiles do not explicitly declare how the environment (its constituent objects) must adapt, but they just provide information about the desired environment state and let smart objects decide how to meet the requirements, depending on their internal logic and built-in actuators.

Variable substitution in Behavioral Profiles is possible to allow postcondition elements to be bounded to unknown precondition elements as in "whatever the location Alice is in, if that location is a room *x*, then that room *x* 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 she gets in.

The XML representation of this Behavioral Profile in SoaM is:

```
<adaptationProfile id="urn:uuid:prof1" expires="
    PT10M">
  <variable id="x"/>
  <precondition subject="urn:uuid:alice"
    predicate="http://www.awareit.com/onto/2005/12/
        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
        /2005/12/location#Room"/>
  </precondition>
  <postcondition subject="x"
    predicate="http://www.awareit.com/onto/2005/12/
        temperature#hasTemperature">
    <objectLiteral datatype="http://www.w3.org/2001/
        XMLSchema#int">
      24
    </objectLiteral>
  </postcondition>
</adaptationProfile>
```

Of course, this Behavioral Profile can only be honored if a smart object providing information about Alice's location is present, along with other device exhibiting operation capabilities over the room temperature.

Context Information, Constraints and Behavioral Profiles are tightly interconnected: Behavioral Profiles' preconditions are resolved against existing Context Information in order to generate Constraints (by substituting variables in the post-conditions) that can be executed by the appropriate smart objects in the environment, thus implementing the context-aware reactive behavior.

A. Smobject

A *Smobject* (a portmanteau for "smart object") is the agent we have designed for representing a context-aware reactive entity in the SoaM architecture. The way a smobject is connected to the actual physical device, sensors or actuators is out of the scope of SoaM standardization being highly platform-dependent so that designers are free to select the best choice depending on existing requirements.

Smobjects are able to capture a subset of environmental information, declared in their perception capabilities, and provide that perceived context information under request to other smobjects o entities. Smobjects are also able to act upon the same or other subset of environmental conditions, declared as operation capabilities, in order to modify them as required via constraints. Figure 2 depicts the smobject communication interfaces.

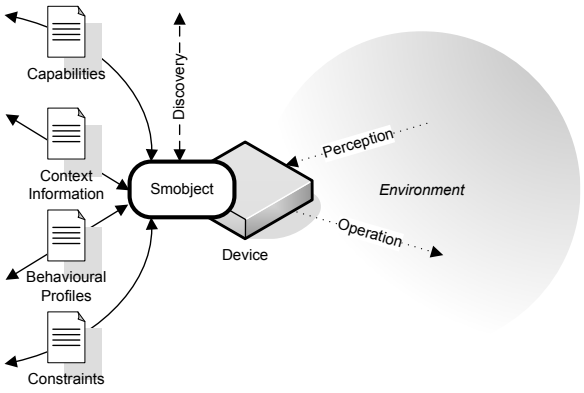

Fig. 2. Smobject communication interfaces.

We have designed the internal architecture of the smobject as a composition of different functional components. These components can be grouped into two major sets: the SmobjectBase components and the SmobjectAware components.

The SmobjectBase components are always active, providing the basic operation of the smobject: perceiving information through sensors, annotate it semantically and make it available to other entities; and receive constraints representing requested changes acting upon the effectors appropriately.

The SmobjectAware components provide full autonomous context-aware reactivity and intelligent capabilities to the smobject. These components are activated when no other more intelligent external entities exist in the environment orchestrating the smobjects.

That is, smobjects are intrinsically powered with reasoning capabilities but, since they are resource-constrained devices, their reasoning power and performance is severely limited. If other external entities present in the environment, e.g.

computers, are able to provide a higher degree of reasoning power, smobjects transfer the reasoning responsibility to them.

This flexible scheme makes smobjects able to cope and adapt dynamically to a broad range of variable scenarios, trying always to get the most from existing facilities in order to provide an intelligent experience for users.

Figure 3 illustrates the internal structure of a smobject including both SmobjectBase and SmobjectAware components.

We have designed four SmobjectBase core components in SoaM:

- Discovery Module: is the component in charge of receiving search requests from other entities and replying to them accordingly. We experimented with two alternatives for discovery: initially UPnP SSDP was extended with some additional headers, but later we designed a full semantic discovery protocol called mRDP (Multicast Resource Discovery Protocol), which is the one we finally applied [43]. Anyhow, the Discovery Module features a pluggable architecture where several discovery protocols, honoring a concrete interface, can be plugged in to use them simultaneously.
- Perceptor Manager: is the component in charge of managing the perceptor interfaces, gathering all the semantic information from them, ultimately obtained via built-in device sensors, in order to make it available to requesting parties via the HTTP interface. Perceptors are semantic gateways that annotate data obtained by sensors using RDF and OWL.
- Effector Manager: is the component in charge of managing the effector interfaces, which ultimately operate the built-in device actuators. The Effector Manager receives the constraints that drive the smobject behavior, resolves conflicts among them if required using the Conflict Resolver, to generate the applied constraints for the effector interfaces. Effector interfaces are gateways that transform semantic constraints into low-level operations over built-in actuators. The Effector Manager also deals with the constraints life-cycle, removing them periodically upon expiration.
- Conflict Resolver: is the component that performs conflict resolution among the set of received active constraints to generate the final set of applied constraints that will be effectively applied.

These components, performing the high-level management functions, are identical for every smobject whatever its purpose. Smobjects can behave very differently and be applied for a broad range of solutions by creating environment specific platform interfaces as explained below, however, the SmobjectBase core components remain immutable, providing platform-agnostic functionality.

On the other hand, the components that embody specific device functionality in SoaM are represented through two types of platform interfaces:

- Perceptors: they act as interfaces between the Perceptor Manager and the built-in device sensors. Perceptors are connected to device sensors using platform-specific libraries, APIs or system calls. Every perceptor implements at least one of the declared perception capabilities of the smobject. They collect the information read by the sensor, annotate it semantically using the appropriate knowledge domain ontology depending on the information type, and make that information available to the Perceptor Manager.
- Effectors: they act as interfaces between the Effector Manager and the built-in device actuators. Very much as perceptors, effectors are connected to device actuators using platform-specific libraries, APIs, system calls or any other mechanism. Every effector is responsible of at least one declared operation capability of the smobject. The Effector Manager provides the effectors with the constraints they are able to process depending on their operation capabilities. Afterward, these logical effectors act upon the built-in device actuators in order to carry out the desired behavior.

It is noteworthy how the whole set of perception and operation capabilities of the smobject are functionally implemented by these logical perceptors and effectors, respectively, which are connected to the actual physical device sensors and actuators. Therefore, perceptors and effectors act as *semantic gateways* to the physical components.

On the other hand, SmobjectAware components provide the smobject with intelligent capabilities and autonomous reactive behavior. Trough these components, the smobject is able to receive behavioral profiles from requesting parties, retrieve capabilities and context information from other neighbor smobjects, and finally perform reactive behavior based on all this knowledge.

There are four SmobjectAware components:

- Profiles Manager: manages the life cycle of the behavioral profiles in the smobject, removing them at expiration and renewing them under request. It receives the profiles through the HTTP interface, passing them along to the Awareness Engine for processing.
- Entity Manager: periodically searches for other smobjects in the environment using installed discovery protocols. The Entity Manager is the component in charge of communication with other smobjects, retrieving their capabilities and context information, as well as the local smobject context information provided by the Perceptor Manager, and making the resulting knowledge available to the Awareness Engine.
- Awareness Engine: whenever a behavioral profile exists, it retrieves the required context information from other smobjects (including the local smobject) via the Entity Manager and resolves the profile generating the constraints, which are passed along to the Constraints Manager.
- Constraints Manager: manages the life cycle of the constraints, dispatching, renewing and finally removing them when no longer required in the local smobject via the Effector Manager. Removal of constraints may have two different causes: either the context information has changed from the last profile resolution, or the original profile that generated the constraint has been removed or not renewed by the requester.

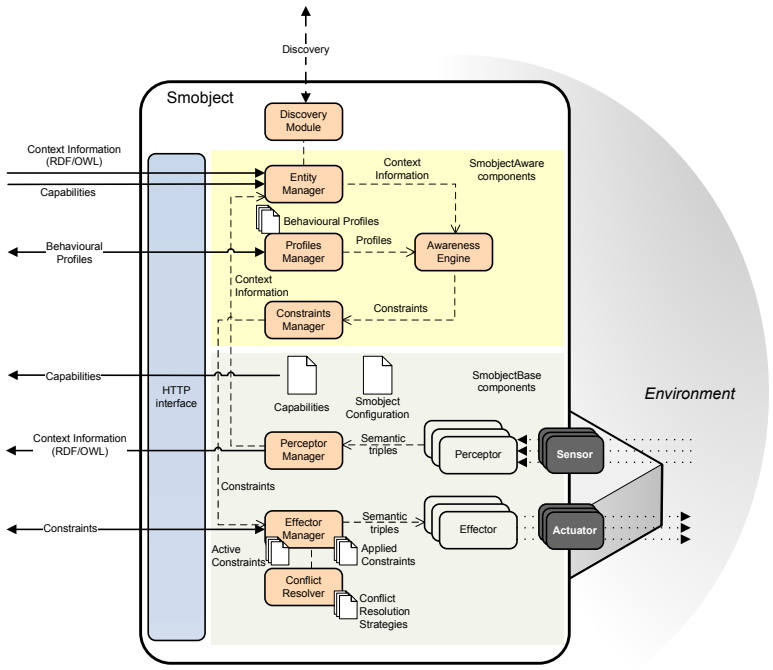

Fig. 3. Smobject internal structure.

Designed in such a way, the smobject is a full context-aware reactive entity pursuing a major goal: to adapt its behavior accordingly to changes in the environment, that is, to changes in perceived context information.

Smobjects are sensitive to environmental changes via both its *direct* perception capabilities, represented by platform interfaces to built-in sensors; and its *indirect* perception capabilities, represented by fellow smobjects in the neighborhood whose information is collected. Smobjects implement the desired behavior through direct operation capabilities over the effectors. That is, smobjects can perceive all the surrounding context information, even that provided by other smobjects, but they can only operate their own built-in actuators (perceive globally, act locally).

This model represents, from our point of view, the autonomous context-aware decentralized reactivity desired for semantic devices: driven by the expressive behavioral profiles, the smobject collects context information from the environment both directly and from other existent smobjects via HTTP requests, it resolves the profiles against this context information, and generates the constraints that ultimately affect its behavior.

B. Orchestrator

We have designed an optional entity in the SoaM architecture called *Orchestrator*, generally hosted in an advanced computing platform, that extends the functionality of the SmobjectAware components to take advantage of more intelligent and powerful reasoning mechanisms.

This entity strongly resembles others present in centralized architectures such as the context broker of CoBrA, providing similar functions. The main difference is that SoaM can operate *without* this central component, in a full decentralized way.

Basically, orchestrators are formed by the same individual components as the SmobjectAware, behaving the same way except that the “AwarenessEngine” is renamed as “Orchestrator Module” and features an attached high-level Reasoner.

Orchestrators are named this way, because they perceive and coordinate existing smobjects in the environment in order to implement a more refined and centrally directed context-awareness process. The orchestrator is the agent where the reasoning can be fully exploited and pushed to its limits, thus is the entity that may exhibit more intelligence in the SoaM architecture, as well as more computing capabilities.

Another important difference between smobjects and orchestrators is that while the Constraints Manager at the SmobjectAware discards constraints not suitable for the local smobject, the Constraints Manager at the Orchestrator processes all the constraints, finding the appropriate smobjects to inject them into.

Therefore, the main role of the orchestrator is to process behavioral profiles for the environment, applying advanced reasoning mechanisms, resolve the profiles into constraints and send those constraints to the appropriate smobjects.

An orchestrator exposes several services through its communication interfaces:

- Discovery: replies to discovery requests issued by other entities.
- Context information retrieval: provides requesting entities with the context information collected by the orchestrator from all the managed smobjects, augmented after reasoning.
- Behavioral profiles management: provides management operations over the profiles that represent the required environmental behavior the orchestrator is in charge of.

Orchestrators do not only provide these services, but also they request some from other entities:

- Discovery: orchestrators search the network continuously for smobjects and retrieve their capabilities.
- Context Information retrieval: orchestrators request context information from smobjects in order to build a comprehensive knowledge base of the environment over which reasoning can be performed.
- Constraints management: orchestrators inject constraints into smobjects to drive their behavior, managing them as needed.

Figure 4 depicts the communication interfaces of the orchestrator.

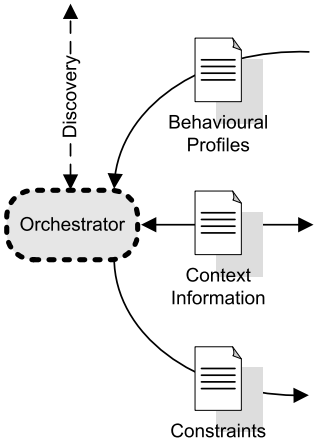

Fig. 4. Orchestrator communication interfaces.

The orchestrator implements a more sophisticated context-awareness process by managing and orchestrating the smobjects. The steps carried out by the orchestrator are:

1) Listen for behavioral profiles: the reception of a profile is the event that triggers the context-awareness process.
2) Discover existing smobjects.
3) Retrieve context information from the smobjects.
4) Apply description logics based on available ontologies and domain rules to augment the context information.
5) Resolve existing behavioral profiles against the augmented context information obtaining the constraints.
6) Identify the smobjects whose operation capabilities can honor the generated constraints and inject those constraints into them.
7) Manage and renew the constraints' influence on the smobjects as needed.

These activities are carried out continuously by the orchestrator, whenever at least one behavioral profile exists.

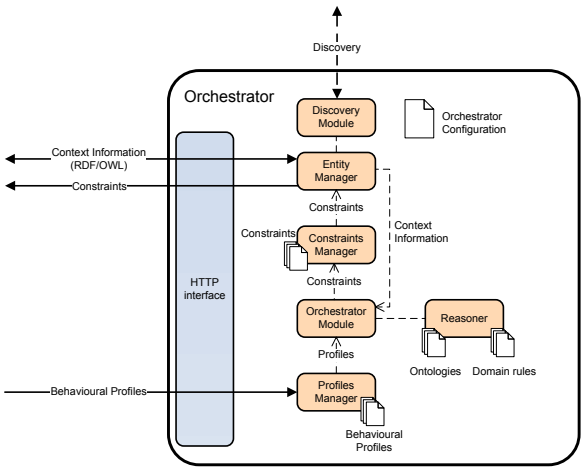

Fig. 5. Orchestrator internal structure.

The orchestrator functional components are depicted in Figure 5. The internal architecture resembles the SmobjectAware except for two new modules:

- Orchestrator Module: whenever a behavioral profile is stored, it retrieves the context information from the smobjects via the Entity Manager, resolves the profile applying description logics embodied in ontologies as well as knowledge domain rules if available, generating the constraints, which are passed along to the Constraints Manager.
- Reasoner: implements one or several reasoning mechanisms, such as description logics or inference rules. It receives context information directly obtained from environmental entities and returns that information augmented via reasoning.

Moreover, the communication flow that linked the Constraints Manager to the Effector Manager in the smobject architecture is substituted by a link between the former and the Entity Manager, since the constraints are injected through HTTP communication into external smobjects (as opposite to internal communication between SmobjectAware and SmobjectBase components within a smobject).

The mission of the orchestrator is both to augment and enrich context information through reasoning as well as to coordinate smobjects behavior to implement the required goals. The orchestrator, if present, is devoted to reasoning for the sake of other smobjects and to creating apparent high-level intelligence in the environment.

C. SoaM client

We have designed a SoaM entity called *SoaM client* that acts on behalf of a user or other entity (e.g. an smobject), disseminating behavioral profiles through the environment in

order to obtain personalized behavior and reactivity from existing smobjects.

The SoaM client does not expose any service to other entities, acting solely as a client. Its internal architecture is very simple, easily embeddable in resource-constrained platforms, since it is only composed of three components:

- Discovery Module: periodically searches the environment for orchestrators or smobjects, using the provided discovery protocols.
- Entity Manager: manages the information related to the entities found in the network, orchestrators and smobjects, retrieving their capabilities in the latter case.
- Profiles Manager: injects the client's behavioral profiles into the appropriate entities, renewing them as needed.

The SoaM client is the entity that triggers the whole process of behavioral change and adaptation in the environment by injecting the profiles into the orchestrator if present, or into the smobjects if no orchestrator exists.

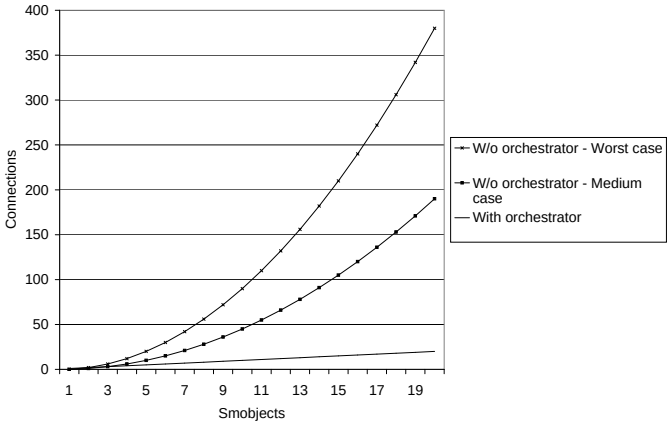

Fig. 6. Number of connections for retrieving context information with and without orchestrator.

D. Topologies

There are some remarkable differences to be aware of when deciding whether to use or not an orchestrator in the environment.

From our point of view, the following are the major advantages of the orchestrator-powered topology:

- More intelligence: the orchestrator provides reasoning mechanisms over context information that augment the knowledge base about the environment.
- Reduced network traffic: the orchestrator acts as an "information attractor", a central point where context information converges. This greatly reduces the traffic compared to the smobjects-only scenario (see below).
  In the worst case where each smobject needs context information from all the other smobjects, the amount of connections to collect the data is $n \times (n-1)$, being n the number of smobjects in the environment. In a medium case, where any smobject must collect information from 50% of the available smobjects, the number of connections is $\frac{n \times (n-1)}{2}$.
  In the presence of an orchestrator, the amount of connections to collect context information is just n: one connection per smobject from the orchestrator.
  Figure 6 compares the amount of connections required as the number of smobjects increases. While the network traffic increases exponentially with the amount of smobjects in the smobjects-only architecture, this increase is lineal in the orchestrator-powered architecture .

On the other hand, among the major advantages of smobjects-only topology of SoaM are:

- Seamless deployment: the cooperation among smobjects emerges naturally anywhere as they discover each other, there is no need for deploying or providing a central point of control. This feature supports the requirement about the serendipitous and spontaneous nature of Ubiquitous Computing.
- Fault tolerance: smobjects-only networks do not rely on a single central point of control, but the responsibilities are distributed among all the nodes. Any node is equally important and a single failure does not interrupt the process. Orchestrator-powered topologies are out of service if the orchestrator fails.
- Distributed reasoning: autonomous intelligent and collaborative entities represent much better from our point of view the vision of semantic gadgets and smart devices presented in the introduction.

The obvious conclusion is that both topologies are useful and must be selected depending on the circumstances: if an orchestrator is available, it must be used to coordinate environmental behavior with more intelligent and traffic-optimization features; in the absence of such orchestrator, smobjects are still able to autonomously form a context-awareness network and provide the required behavior.

V. Implementation and prototypes

The initial hardware and software platforms selected for hosting the first prototypes of smobjects were Digi ConnectCore 7U (also known as FS Forth UNC20) and Digi ConnectCore 9U (FS Forth UNC 90).

The ConnectCore 7U is an embedded platform powered with an ARM7 microprocessor running at 55 MHz, a μClinux kernel and a stripped-down Java Virtual Machine called mi|k|a. The ConnectCore 7U also featured 8 Mb Flash and 16 Mb RAM, 2 serial interfaces and Ethernet connectivity, easily convertible into Wi-Fi connectivity, in a relatively small form factor of 6.28 cm $\times$ 1.85 cm $\times$ 1.04 cm (12.08 cm^3, see Figure 7).

The ConnectCore 9U is an pin-to-pint compatible upgrade of ConnectCore 7U, featuring a much more powerful 180 MHz ARM9 processor, 16 MB Flash and a Linux (not μClinux) kernel in the same form factor.

All the components of the SmobjectBase and SmobjectAware sets were implemented in Java, along with some helper classes and functions. Reusability of existing Java code from desktop computers was difficult because of the limited processing power and memory of ConnectCore platforms, as

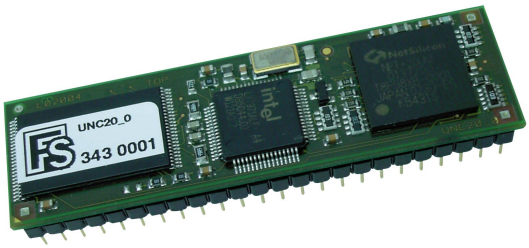

Fig. 7. Image of the ConnectCore 7U platform that hosted the smobject.

well as mi|k|a limitations in terms of core Java libraries (mi|k|a implements a subset of Java 1.3).

There were two especially challenging issues we faced when trying to implement semantic web technologies in this limited platform: RDF parsing and description logics reasoning.

RDF parsing involved a previous phase of XML parsing since we used RDF/XML serialization of context information. After testing several alternatives, *MinML* [46] was chosen as underlying XML parser and a customized version of *RDF Filter* [31] for RDF parsing. As described in the next section, RDF parsing (including XML parsing) is one of the most costly activities in terms of computing time in the smobject platform.

However, implementing a limited semantic reasoner in the smobject was even more challenging. Available RDF or OWL APIs could not be used here due to the constraints mentioned above. A simple library and reasoning engine, especially designed for dealing with RDF triples in limited platforms, was implemented, tested and improved until acceptable results were obtained.

Optimizations in terms of performance can be achieved in many algorithms by creating memory structures that speed up the process. Nevertheless, memory is a scarce resource in embedded platforms, so our design sacrificed performance in order to increase implementation feasibility.

On the other hand, since the orchestrator was intended to run on non resource-constrained devices, all the problems derived of using limited platforms and constrained Java virtual machines were avoided, as well as existing APIs could be applied.

We implemented the four main modules (Profiles Manager, Constraints Manager, Entity Manager and Orchestrator Module) in such a way that they could be run in two different modes: as different concurrent subprocesses or sequentially within the same subprocess.

As concurrent subprocesses the overall performance improved due to parallel activities; for example, the Entity Manager could update the list of smobjects, while the Constraints Manager renewed constraints on existing ones. Our design dealt with possible concurrency problems derived from simultaneous accesses from different execution threads.

We designed a pluggable architecture for reasoners, very similar to the one we used for the discovery protocols, so that any kind of reasoner could be applied without affecting other modules.

We used Jena 2 Semantic Web Framework for developing all the semantic web related activities in the orchestrator, such as serializing and deserializing RDF/XML, performing description logics reasoning or even rule reasoning, since Jena provides several built-in reasoners.

Finally, prototype SoaM clients were implemented to test the whole architecture.

VI. Evaluation

Several performance aspects of the prototype implementation were evaluated both for the smobjects-only and for the orchestration-powered topologies.

A testing scenario resembling a home environment was deployed including smobjects for representing a simulated location system, a TV set, a temperature control system, an "alerter" in a laptop and a Hi-Fi system. A number of behavioral profiles were designed to obtain automatic environmental reactivity depending on certain situations, for example "turn off the TV if I leave the TV room" or "alert me if I am leaving home and the weather forecast informs it is going to rain" (this information was obtained by a perceptor in the laptop smobject that retrieved the forecast from Weather.com, annotated it semantically and made it available to other surrounding smobjects).

Several ontologies were reused, mainly from SOUPA [13], and others were designed from scratch to embody the knowledge of some involved domains such as the TV domain, the weather domain and the location domain, the latter including concepts such as room, building or town.

A particularly interesting issue about the ontology we created for the location domain is the application of the transitive property `islocatedIn` and its inverse `contains` as well as the symmetric property `nearby` to infer the location relationships among the deployed elements. In fact, the majority of behavioral profiles we designed demanded certain reactivity of smobjects depending on their location; for example, to turn on an Hi-Fi system in a certain room nearby the user's location.

These situations promoted the application of ontologies and description logics reasoning in order to successfully obtain the required reactivity. In the concrete scenarios described above an average of 30-50 triples were selectively collected from surrounding smobjects, and augmented around 50% once OWL reasoning was applied (mainly due to inferences obtained through the location ontology). This "augmented" context information was then matched against the previously injected behavioral profiles in order to determine the constraints that finally embodied the desired environmental reactions.

The results of the tests using the completely decentralized smobjects-only architecture are depicted in Figure 8, illustrating the amounts of time required by the smobject to perform every activity.

As already mentioned two activities take most of the operating time: context information retrieval, which includes HTTP communication to poll surrounding smobjects and RDF parsing of the received data; and description logics reasoning.

Our tests also pointed out that the performance of the smobject may be severely affected by external or platform issues such as the garbage collection process, or being intensively polled by other entities during a short period of time. This

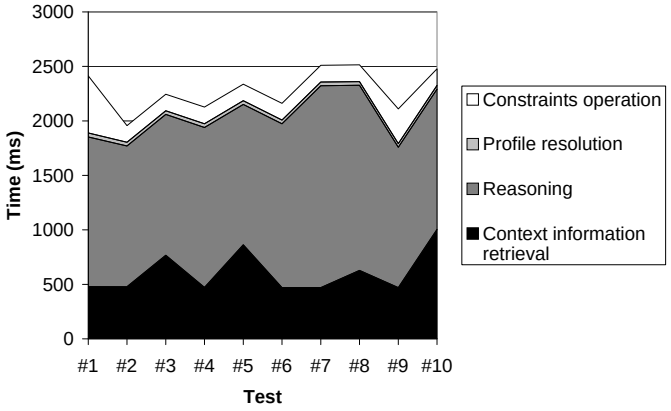

Fig. 8. Performance measures of the smobjects-powered CC9U topology.

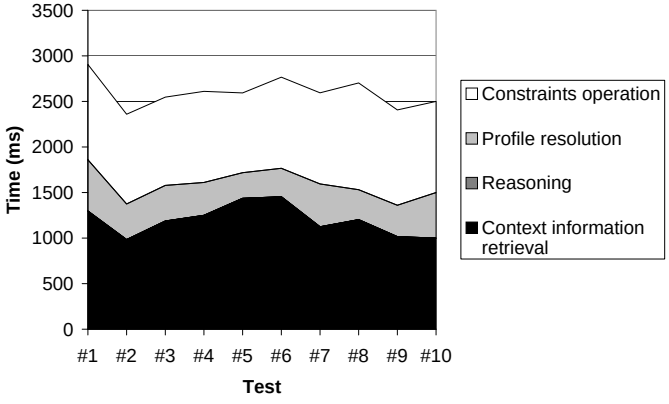

Fig. 9. Performance measures of the orchestrator-powered CC7U topology.

kind of situations arises in limited devices, very sensitive to additional work load, and must be taken into account when a constant reactivity response time is required.

While the smobject implementation in the ConnectCore 7U (CC7U) platform exhibited the above mentioned limitations, the ConnectCore 9U (CC9U) prototype performed exceptionally well. Therefore, we deployed the smobject-only topology with ConnectCore 9U devices hosting SmobjectBase and SmobjectAware components (thus featuring autonomous semantic capabilities), while the orchestrator-powered topology was deployed in an scenario populated by more limited ConnectCore 7U devices hosting SmobjectBase components coordinated by the central orchestrator.

In the smobjects-only topology, the average time required by a CC9U smobject to complete a full cycle including retrieving context information from other smobjects, generating new triples through reasoning, resolving the behavioral profiles against the context information and executing these constraints, was around 2.28 seconds for the above described scenario.

The orchestrator-powered topology performed similarly, around 2.59 seconds, being remarkably slowed by the limited CC7U smobjects, where XML processing and HTTP communication at the context information retrieval phase are costly in terms of time (see Figure 9).

The orchestrator ran in a Pentium-M 1.86 GHz with 1 GB RAM and enjoyed all the advantages of full computing resources. It is especially noteworthy how the use of the Jena API for OWL reasoning reduced the reasoning time to an almost imperceptible amount, due to the extensive optimizations for representing the RDF graph in memory used by Jena. However, profile resolution performed relatively bad compared to the other activities. The approach applied here was to convert the profile into a rule to feed the general rule engine provided by Jena, but this mechanism can probably be optimized in the future.

Since constraints operation is an activity carried out in the CC7U smobject, thus being slowed again by this resource-constrained platform, the performance of this phase is not remarkably improved.

Figure 10 compares the performance of both topologies for the described scenario in absolute terms, while Figure 11 compares them in terms of relative time required for each activity to be carried out.

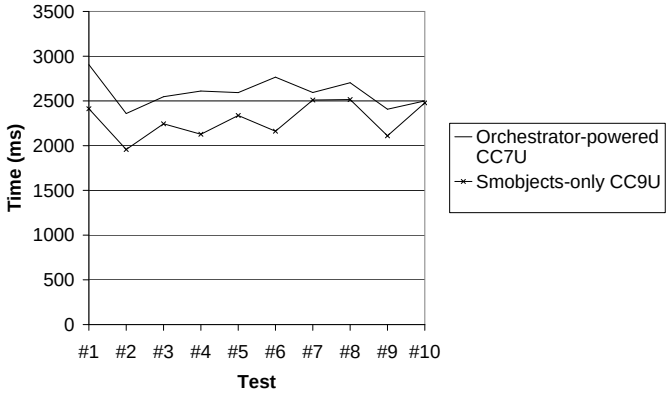

Fig. 10. Comparison of performance measures for the smobjects-only CC9U and orchestrator-powered CC7U topologies.

VII. Conclusion

The web model and Semantic Web technologies can provide the infrastructure over which intelligent Ubiquitous Computing architectures can be successfully designed. After analyzing past efforts in this direction, we created SoaM, a pure web-based architecture that relies on Semantic Web technologies for providing the required intelligent capabilities, thus creating synergies with the web communication model.

Smobjects provide semantically annotated perceptions via HTTP interfaces while they are able to operate the environment using concepts declared in OWL vocabularies about different knowledge domains. Moreover, the SmobjectAware components make smobjects full context-aware semantic entities, able to autonomously gather required context information

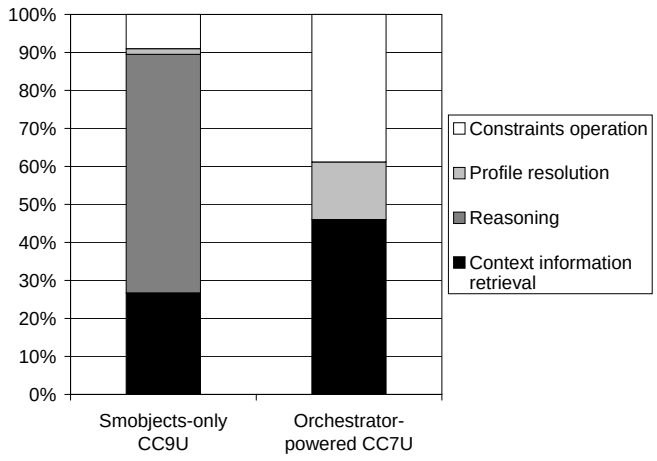

Fig. 11. Comparison of relative effort for the activities in the smobjects-only CC9U and orchestrator-powered CC7U topologies.

from fellow smobjects, reasoning upon that information and carrying out the appropriate behavior.

There are two ways of deploying SoaM in a concrete environment: a distributed smobjects-only topology and a centralized orchestrator-powered topology.

Tests demonstrated how the smobjects-only topology, hosted in powerful embedded platforms such as the ConnectCore 9U, is able to rival the centralized orchestrator-powered topology, at least in scenarios with a limited amount of exchanged semantic information among entities.

Moreover, the smobjects-only topology features a more serendipitous and spontaneous collaboration model among devices, better representing the scenarios envisioned originally by Ubiquitous Computing and Ambient Intelligence. Emerging natural collaboration among smobjects promote the vision of a Pervasive Semantic Web everywhere in order to make environments smarter and more reactive.

We consider that a decentralized web-powered infrastructure, such as the smobjects-only topology of SoaM, is the most suitable alternative to design collaborative semantic devices.

There are still some work and future research in this direction involving using and extending SoaM, for example: migrating the smobject components to a wireless sensor network platform such as Intel Mote2 (featuring 13 MHz up to 416 MHz of computing power); deciding in which scenarios the orchestrator-powered or the smobjects-only topology is more suitable; how behavioral profiles can be generated automatically by analyzing past user behaviors, or how they can be augmented with additional constructions to represent more complex situations; how applying other logics models, such as fuzzy logic, can complement description logics in Ubiquitous Computing; and finally, more research involving users has to be carried out to discover all the possibilities of intelligent semantic devices in real-world scenarios.

ACKNOWLEDGMENT

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

REFERENCES

[1] Gregory D. Abowd, Anind K. Dey, Peter J. Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a better understanding of context and context-awareness. In *HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, pages 304–307, London, UK, 1999. Springer-Verlag.

[2] John Barton and Tim Kindberg. *The cooltown user experience*. Hewlett-Packard, 2001. Technical Report HPL-2001-22.

[3] Fabio Bellifemine, Agostino Poggi, and Giovanni Rimassa. JADE - a FIPA-compliant agent framework. In *Proceedings of the Practical Applications of Intelligent Agents*, 1999.

[4] Tim Berners-Lee. *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by Its Inventor*. Harper San Francisco, 1999.

[5] Tim Berners-Lee, Roy T. Fielding, and Larry Masinter. *Uniform Resource Identifiers (URI): Generic Syntax*, 1998. IETF RFC 2396.

[6] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web: A new form of web content that is meaningful to computers will unleash a revolution of new possibilities. *Scientific American*, 284(5):28–37, May 2001.

[7] Harry Chen, Tim Finin, and Anupam Joshi. An ontology for context-aware pervasive computing environments. In *Workshop on Ontologies and Distributed Systems*. IJCAI-2003, August 2003.

[8] Harry Chen, Tim Finin, and Anupam Joshi. Using owl in a pervasive computing broker. In *Workshop on Ontologies in Agent Systems, AAMAS-2003*, Melbourne, Australia, July 2003.

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

[10] Harry Chen, Tim Finin, and Anupam Joshi. An ontology for context-aware pervasive computing environments. *Special Issue on Ontologies for Distributed Systems, Knowledge Engineering Review*, 18(3):197–207, May 2004.

[11] 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.

[12] Harry Chen, Filip Perich, Dipanjan Chakraborty, Tim Finin, and Anupam Joshi. Intelligent agents meet semantic web in a smart meeting room. In *Proceedings of the Third International Joint Conference on Autonomous Agents & Multi Agent Systems (AAMAS 2004)*, New York City, NY, July 2004.

[13] 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.

[14] Martin Duerst and Michel Suignard. *Internationalized Resource Identifiers (IRIs)*, 2005. IETF RFC 3987.

[15] 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.

[16] John Franks, Phillip M. Hallam-Baker, Jeffery L. Hostetler, Scott D. Lawrence, Paul J. Leach, Ari Luotonen, and Lawrence C. Stewart. *HTTP Authentication: Basic and Digest Access Authentication*, June 1999. IETF RFC 2617.

[17] Adrian Friday, Nigel Davies, Nat Wallbank, Elaine Catterall, and Stephen Pink. Supporting service discovery, querying and interaction in ubiquitous computing environments. *Wireless Networks*, 10(6):631 – 641, November 2004.

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

[19] 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.

[20] 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.

[21] 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.
[22] 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.
[23] European Commission IST Advisory Group (ISTAG). Scenarios for ambient intelligence in 2010. Technical report, EU Commission, 2001.
[24] European Commission IST Advisory Group (ISTAG). Ambient intelligence: from vision to reality. Technical report, EU Commission, 2003.
[25] European Commission IST Advisory Group (ISTAG). IST research content. Technical report, EU Commission, 2003.
[26] 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.
[27] 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.
[28] 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.
[29] Simon St Laurent. *Cookies*. McGraw-Hill, Inc., New York, NY, USA, 1998.
[30] Ryusuke Masuoka, Yannis Labrou, Bijan Parsia, and Evren Sirin. Ontology-enabled pervasive computing applications. *IEEE Intelligent Systems*, 18(5):68–72, September-October 2003.
[31] Meggison Technologies. *RDF Filter*, 2001. http://rdf-filter.sourceforge.net/. Accessed on 27 May 2006.
[32] Martin Modahl, Bikash Agarwalla, Gregory Abowd, Umakishore Ramachandran, and T. Scott Saponas. Toward a standard ubiquitous computing framework. In *Proceedings of the 2nd workshop on Middleware for pervasive and ad-hoc computing*, pages 135–139, New York, NY, USA, 2004. ACM Press.
[33] Davy Preuveneers and Yolande Berbers. Suitability of existing service discovery protocols for mobile users in an ambient intelligence environent. In *Proceedings of the International Conference on Pervasive Computing and Communications*, pages 760–764. CSREA Press, June 2004.
[34] Eric Rescorla. *HTTP Over TLS*, May 2000. IETF RFC 2818.
[35] 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.
[36] Manuel Román, Christopher Hess, Renato Cerqueira, Anand Ranganathan, Roy H. Campbell, and Klara Nahrstedt. Gaia: a middleware platform for active spaces. *SIGMOBILE Mob. Comput. Commun. Rev.*, 6(4):65–67, 2002.
[37] 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.
[38] 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.
[39] 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*, pages 80–97. Springer, 2005.
[40] UPnP Forum. *UPnP Device Architecture v.1.0.1 Draft*, 2003.
[41] Juan Ignacio Vazquez, Joseba Abaitua, and Diego López de Ipiña. The ubiquitous web as a model to lead our environments to their full potential. In *Proceedings of the W3C Workshop on the Ubiquitous Web*. World Wide Web Consortium, 2006. Position paper.
[42] Juan Ignacio Vazquez, Diego López de Ipiña, and Iñigo Sedano. SoaM: An environment adaptation model for the pervasive semantic web. In *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.
[43] 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 Inteligence (ECAI 2006)*, August 2006.
[44] Xiaohang Wang, Jin Song Dong, ChungYau Chin, SankaRavipriya Hettiarachchi, and Daqing Zhang. Semantic space: An infrastructure for smart spaces. *IEEE Pervasive Computing*, 3(3):32–39, 2004.
[45] Mark Weiser. The computer for the 21st century. *SIGMOBILE Mobile Computing and Communications Review*, 3(3):3–11, 1999.
[46] John Wilson. *MinML: a minimal XML parser*, November 2001. http://www.wilson.co.uk/xml/minml.htm. Accessed on 27 May 2006.

Juan Ignacio Vazquez Juan Ignacio Vazquez completed the degree of Computer Science Engineering at the University of Deusto. Since 1997 he works as a lecturer of telematics and mobile computing at the University of Deusto, and since 2003 he also works as a researcher of MoreLab - Mobility Research Lab at Tecnológico Fundación Deusto. His main research interests are web-based communications, the application of semantic web technologies to Ubiquitous Computing, and mobile computing.

Diego López de Ipiña Dr. Diego López de Ipiña completed a PhD at the University of Cambridge, UK on Sentient Computing in 2002. After that, he took part on the Trigenix start-up, specialized on providing personalized interfaces to mobile devices. Since September 2003 he is a lecturer at the University of Deusto, Spain, and researcher of MoreLab - Mobility Research Lab. His main research interests are middleware architectures for AmI, context-aware mobile computing and the synergy between Web 2.0 and Pervasive Computing.

Iñigo Sedano Iñigo Sedano completed the degree of Telecommunications Engineering at the University of Deusto. He currently works as a researcher of MoreLab - Mobility Research Lab at Tecnológico Fundación Deusto, with an special emphasis on wireless communication technologies and wireless sensor networks.