

# A Passive Influence Model for Adapting Environments based on Semantic Preferences

Juan Ignacio Vazquez, Diego López de Ipiña, and Iñigo Sedano

University of Deusto, Avda. Universidades 24, 48007 Bilbao, Spain  
ivazquez@eside.deusto.es, dipina@eside.deusto.es,  
isedano@tecnologico.deusto.es

**Abstract.** Creation of intelligent environments has been a major subject of research during the last years through different approaches that model adaptation capabilities of the environment and devices. Often, designed solutions are too much problem-oriented and expose many ad-hoc features, being difficult to export to other cases or scenarios, and contributing to a lack of standardized models of representing the adaptive nature of smart spaces. In this paper, we present a model, based on the joint synergies of Ubiquitous Computing and Semantic Web, for passively influencing the environment and get it adapted to user preferences without explicit intervention.

## 1 Introduction

The final goal of Ubiquitous Computing and Ambient Intelligence [1] is creating intelligent spaces to empower users in everyday tasks at home, work, street, vehicle and others. In this vision, environments are proactive perceiving users' surrounding information, often referred as context, and reacting in the appropriate way to facilitate user's 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 [2].

Users can perceive intelligence in the environment by observing required interactions. Engineers try to increase systems' perceived intelligence by reducing interactions often following a very simple rule: more system autonomy about detecting and reacting to users' goals leads to less users' interactions. In this way, an automatic door can be considered intelligent since it can easily detect users' goals (pass through it) and react appropriately (to open). The final objective of smart environment designers is to extrapolate the simplicity of the automatic doors to every object, thus creating some kind of invisible but present surrounding intelligence.

We distinguish two different approaches for any agent to change and adapt the environment:

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 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 [8].

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 [9].

For instance, by broadcasting a preference stating “my preferred temperature for my location is 24°C”, a user does not control explicitly any device, but they can be aware and change their state 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<sup>2</sup>lets middleware [10], 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 initially, while allowing users to customize it later via explicit methods. 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, but the user is free to change them explicitly afterwards.

Passive mechanisms have not been very much explored in pervasive computing scenarios; most of existing systems use forms of active mechanisms, such as WebServices/SOAP (UPnP[7], Task Computing [3], WSAMI), Jini, CORBA, and so forth.

In this paper, we introduce a model for passive influence: SOAM - Smart Objects Awareness and Adaptation Model. SOAM applies Semantic Web technologies to create context-aware environments that react automatically to user preferences, technically called adaptation profiles, without explicit user intervention.

We think that passive influence models must be explored in order to build more autonomous and reactive environments, and we have tried to design such a model as explained below.

## 2 SOAM: Smart Objects Awareness and Adaptation Model

### 2.1 The Pervasive Semantic Web Vision

We coin 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)[5] and OWL(Ontology Web Language)[6].

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

Our vision is to create some kind of Personal Area Web, where 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.

This point of view about future living and working environments is shared with other research groups that provided similar viewpoints [4], but until now there are no practical results or architectures developed and tested.

SOAM – Smart Object Awareness and Adaptation Model – is intended to fill this gap by providing a comprehensive architecture, easily replicable, to test automatic environment adaptation scenarios by applying semantic annotation techniques.

## 2.2 Smobjects: Smart Objects

In SOAM, a major agent is what we denominate *smobject* as a short for “smart object”. Smobjects are agents representing an intelligent device, several devices or event part of a device. Smobjects capture a subset of environmental conditions, provide that perceived information under request, and act upon the same or other subset of environmental conditions in order to modify them and adapt them as needed.

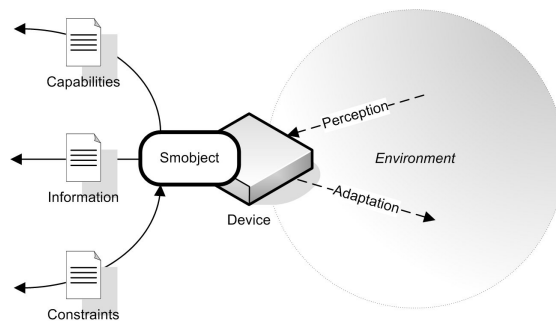
Smobjects, as agents, are conceptual entities connected to one or several physical devices. The way a smobject is connected to the actual device, sensors or effectors is out of the scope of SOAM standardization, being highly platform-dependent so that designers are free to select the best choice depending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

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.

Smobjects exchange three different kinds of information:

- **Context information:** smobjects provide context information about perceived state of the environment via semantically annotated documents under request. Information is gathered through built-in sensors in the bounded device or via other connected smobjects that provide it. Perceived information is provided by the smobject to requesting parties.

- **Capabilities:** a smobject is capable of perceiving only some concrete environmental conditions depending on the bounded-device built-in sensors, and it is also capable of operating over some (same or other) conditions depending on the bounded-device built-in effectors. Perception and operation capabilities are exhibited by the smobject to other parties.
- **Constraints:** smobjects 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 smobject is in charge of driving adaptation honoring them. Smobject’s behaviour is defined by active constraints, which represent existing influences over the smobject, and have a limited lifespan. Since Constraints refer to current Context Information, smobjects’ behavior becomes naturally context-aware (as shown in figure 3). Smobjects can provide information about their active constraints to requesting parties, as well as accept constraints from other entities that desire to influence the smobject’s behaviour.



**Fig. 1.** Smobject interfaces to access data entities.

### 2.3 Context Information

Context Information is probably the most important data structure a smobject can provide. Context Information is constructed using RDF serialized in the form of XML, thus representing semantic knowledge about certain factors. It conveys perceived information captured through device’s sensors, annotated via RDF and OWL. Captured data semantics is highly knowledge domains -dependent, for example temperature measures, an item location or user’s personal information.

An example of a Context Information message conveying knowledge about luminance is shown in figure 2. In this example, the smobject is installed in a lighting device called `light1`, and it provides information about `light1`’s state upon request (luminance, light color, and so forth).

As we can notice, a smobject normally does not only provide information about the perceptions obtained by sensors, but also the device identification and

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

**Fig. 2.** An example Information structure provided by a smobject.

type, that is, the full semantic description of available data. This annotation is particularly useful to automate processes depending on device identification, type or other device parameters.

## 2.4 Capabilities

A smobject can exhibit perception capabilities on some domains and operation capabilities on the same or different domains. Perception capabilities represent sensing mechanisms the smobject is able to access on the host device about some domains (for example, lighting conditions), while operation capabilities represent control mechanisms the smobject features about some domains. Capabilities are exchanged using XML.

## 2.5 Constraints

Smobjects receive requests to perform environment adaptation through effectors. These requests come in the form of Constraints, represented by statement patterns in the desired behavior. A smobject can receive a number of this kind of constraints over the time, so its behavior is influenced and driven by them. In fact, a smobject is in charge of managing the active Constraints and trying to perform in such a way that Constraints are honored.

XML is used to send constraints to smobjects and influence their behavior. Figure 3 illustrates an example of active Constraints on `room1`.

The first constraint could be read as “*Alice requested room1 to have luminance less than 80*”, and the second as “*Alice requested room1 to have a temperature of 24° C*”.

Constraints are injected into smobjects to constrain its behavior and thus have the environment conveniently adapted. Constraints are generated and sent to appropriate smobjects depending on smobject capabilities and user preferences about the environment. Those preferences come in the form of Semantic Adaptation Profiles.

```

<constraintsCollection xmlns="http://www.awareit.com/soam">
  <constraint id="urn:uuid:alice_const1" requester="urn:uuid:alice"
    subject="urn:uuid:room1"
    predicate="http://www.awareit.com/onto/lighting#luminance"
    operator="http://www.awareit.com/onto/soam#lessThan">
    <objectLiteral datatype="http://www.w3.org/2001/XMLSchema#int">
      80
    </objectLiteral>
  </constraint>
  <constraint id="urn:uuid:alice_const2" requester="urn:uuid:alice"
    subject="urn:uuid:room1"
    predicate="http://www.awareit.com/onto/temperature#hasTemperature">
    <objectLiteral datatype="http://www.w3.org/2001/XMLSchema#int">
      24
    </objectLiteral>
  </constraint>
</constraintsCollection>

```

Fig. 3. Example Constraints to influence a smobject.

## 2.6 Adaptation Profiles

The goal of SOAM is to achieve a comprehensive model for automatic passive adaptation of the environment to user preferences, needs and behavioral patterns. Smobjects are the entities in charge of performing the final operation to achieve adaptation, but constraints must be generate by other agent.

Adaptation Profiles are the information elements that conveys user's adaptation requirements that eventually drive smobjects behavior, through constraints. Adaptation Profiles are stored and exchanged with the environment via the user's personal device.

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°C, 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 smobjects) must adapt, but just inform about the desired environment state and let smobjects 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 4.

```

<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>

```

**Fig. 4.** An example Adaptation Profile with one bounded variable.

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° C*”, 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 honored if a smobject exhibiting operation capabilities over the room temperature is present.

Both Adaptation Profiles and Constraints have a leasing time when accepted by Ambient Adapters and smobjects respectively, that must be renewed periodically by the requesting party as indicated in the HTTP response message.

There is a special agent in SOAM called Orchestrator in charge of receiving user Adaptation Profiles, obtaining constraints by evaluating the Adaptation Profiles against existing Context Information (provided by smobjects), and sending those constraints to appropriate smobjects to drive their behavior. The Orchestrator acts as central entity that “translates” user profiles into smobject (or environmental) constraints.

The overall result is that the environment adapts automatically to user preferences without explicit intervention, which fulfills our goal of achieving an exemplar architecture of a passive influence model.

### 3 Conclusions and Future Work

SOAM is an effort to create a passive influence model for automatic environment adaptation to user’s preferences. 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[11], for this purpose.

We are currently implementing SOAM in order to benchmark passive influence mechanisms against active influence ones, during execution of different tasks in several scenarios. Smobject agents are implemented in Mika (a Java VM) for  $\mu$ CLinux in ARM7-based UNC20 embedded platforms.

There are some open research issues related to SOAM architecture and passive influence models that still need be studied such as conflict resolution with multiple users' disjoint requirements, and the possibility of distributed orchestration, without a central entity.

## 4 Acknowledgements

This work has been partially supported by the Department of Industry, Commerce and Tourism of the Basque Government under the SAIOTEK grant S-OD04UD02, and the Cathedra of Telefonica Moviles at Deusto University, Bilbao, Spain.

## References

1. K. Ducatel, M. Bogdanowicz, F. Scapolo, J. Leijten and J-C. Burgelman. *Scenarios for Ambient Intelligence in 2010. Final Report*. IST Advisory Group. EC (2001).
2. Project Aura. <http://www.cs.cmu.edu/aura/>
3. R. Masuoka and Y. Labrou. *Task Computing - Semantic-web enabled, user-driven, interactive environments*. WWW Based Communities For Knowledge Presentation, Sharing, Mining and Protection (The PSMP workshop) within CIC 2003, Las Vegas, USA (2003)
4. Ora Lassila. *Using the Semantic Web in Mobile and Ubiquitous Computing*. Proceedings of the 1st IFIP WG12.5 Working Conference on Industrial Applications of Semantic Web, pp. 19-25. Springer (2005).
5. World Wide Web Consortium. *RDF Primer. W3C Recommendation*. World Wide Web Consortium (2004).
6. World Wide Web Consortium. *OWL Web Ontology Language Semantics and Abstract Syntax. W3C Recommendation*. World Wide Web Consortium (2004).
7. UPnP Forum. *UPnP Device Architecture 1.0*. UPnP Forum (2003).
8. J. I. Vazquez and D. Lopez de Ipiña. *An Interaction Model for Passively Influencing the Environment*. Adjunct Proceedings of the 2nd European Symposium on Ambient Intelligence, Eindhoven, The Netherlands (2004).
9. J. I. Vazquez and D. Lopez de Ipiña. *A language for expressing user-context preferences in the web*. WWW 2005: Special interest Tracks and Posters of the 14th international Conference on World Wide Web (Chiba, Japan) pp. 904-905. ACM Press (2005).
10. D. Lopez de Ipiña, J. I. Vazquez, D. Garcia, J. Fernandez and I. Garcia. *A Reflective Middleware for Controlling Smart Objects from Mobile Devices*. sOc-EUSAI 2005: Smart Objects & Ambient Intelligence, Grenoble, France (2005).
11. H. Chen et al. *SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications*. Proceedings of Mobiquitous 2004: International Conference on Mobile and Ubiquitous Systems: Networking and Services, Boston, USA (2004).