Environment Adaptation Meeting User Preferences

Juan Ignacio Vazquez and Diego López de Ipiña

Faculty of Engineering, Deusto University, Avda. Universidades, 24,
48007 Bilbao, Spain
{ivazquez, dipina}@eside.deusto.es

Abstract. Automatic adaptation of the environment to user preferences relieves users from continuously re-configuring their physical context as periodic everyday activities are performed. However, the problem of environment adaptation has been traditionally solved via ad-hoc context-specific solutions for concrete problems and situations. In this paper we present the WebProfiles Markup Language, a mechanism for Ubiquitous Computing, enabling user preferences representation capabilities for influencing surrounding devices and services, so that the environment meets the user and knows him as if specifically designed for that purpose.

1 Introduction

Context-awareness mechanisms and, in particular, user-related information awareness, are one of those required extensions for the Ubiquitous Computing to fulfill present and future services demands. Context-awareness would allow a service to perceive user-related and task-related information that can be used to provide a more suitable and effective outcome for that user. Context information can be provided by the user-agent explicitly (user-related data), or can be extracted by the service from other available sources in a scenario dependent paradigm.

On the other hand, the Web model, including HTTP technology, has proven to be suitable to support communication needs in Ubiquitous Computing scenarios [1,2], so that the same communication model can be applied to provide both global scale and local scale services.

HTTP context awareness is a broad concept than can embrace the traditional HTTP state management mechanism [3], which has been very criticized over the years, despite the Web would not be as powerful as it is without those small chunks of information called cookies [4]. These pieces of data allow a web service to recognize immediately a visiting user and parameterize the nature of the information being presented based on past visits and interaction, and it can be considered a very simple form of user-awareness mechanism.

In order to materialize new capabilities we have created the WebProfiles model: an HTTP extension that supports context information management as well as a negotiation process that allows clients and service providers to establish the appropriate informational environment for the service execution in Ambient Intelligence or Ubiquitous Computing scenarios. The goal is to extend Personal Area Networks to create what we call *Personal Area Webs*, applying the successful web communication model around the user physical circumstances in order to weave context-aware user-device relationships.

In our vision, Personal Area Webs exist around the user and move with him, linking that user with the surrounding devices, and creating a services-rich digital ecosystem extended with user awareness capabilities.

The WebProfiles model shares many similarities with both OPS (Open Profiling Standard) [6] and the initial specifications of the Platform for Privacy Preferences (P3P) [7], inheriting some of their characteristics, but we stress the use of user-related context in the form of preferences about service characteristics. While CC/PP [8] seems to be a good initial alternative, it is too oriented to express device information and concrete data instead of conditional preferences as explained below.

Special attention must be paid to WS-Context [9], an ongoing work to define a mechanism for context information sharing among multiple coordinated services for executing a task. This specification is tightly linked to the Web Services technologies such as SOAP [10], WSDL [11] and more concretely to WS-CAF (Composite Application Framework), WS-Coordination and WS-Transactions.

In section 2, we introduce the concept of context awareness and its implications for personal area web-based services. In section 3, we present our main contribution in this paper, the WebProfiles Markup Language (WPML), the part of the WebProfiles model that supports user preferences representation, that is, the context data used to automatically adapt the environment. Finally in section 4, we present some open issues about the evolution of the WebProfiles model, WPML and personal area user-awareness in general.

2 User Context Awareness in the Personal Area Web

It is not easy to find a widely accepted definition for "context", since it is very dependable of the framework in which is applied. One of the most precise and open statements we can mention is found in the WS-Context specification 0 and declares that a "context contains information about the execution environment of an activity".

That is, a context is an information entity that can be used to provide additional data for some process execution. Probably, that execution could be performed without that supplementary information, but surely its influence can be used to establish a user-adapted execution framework more precisely.

Probably, and important part of the context information for a service is related to the user, expressing data about him, his preferences maybe depending on other context information, and so on. We can define *user context information* as the subset of the context information influencing a service that model user-related aspects.

When coping with personal area web services and web processes, it is often necessary to exchange a large amount of data to execute a service. The service provider needs to be supplied with all the data the user keeps that are relevant to the situation. For example, if a user wants to configure his temperature preferences every time he enters a new room in a building, he must repeat similar interactions over the temperature control devices once and again at every location.

Web-based surrounding devices populated with embedded web services are not aware of users' context, provoking unnecessary interactions refinements over the time that end up in entering the same data manually along different devices repeatedly.

HTTP state management mechanism has provided a simple method for a service provider to recognize the user in subsequent visits via cookies. Nevertheless, cookies are used primary for client identification, not for context information representation due to format limitations and security considerations.

Our goal was to find a mechanism as simple as cookies but able to cope with user context information sharing between embedded clients and servers, where user preferences could be formally defined and structured so that they could be passed forward to validated devices in order to obtain a more personalized service execution.

That is, prior to actual service interaction between the user and the service provider (a web-enabled device), the user-agent and the embedded server negotiate and set up an information-rich context in such a way that it seems that the service provider *knows the user beforehand*, despite the latter has never interacted with the device before. Further interactions can be accomplished inside that mutual knowledge framework.

Figure 1 illustrates the interaction process between a client, also called user-agent (some kind of process running within the user's mobile phone or PDA), and a service provider (a surrounding device) in the usual way, without previous context negotiation.

Data are supplied by the client as needed, increasing the number of interactions. This diagram is familiar in the traditional Web paradigm, since several extensions implement similar mechanisms, such as HTTP Authentication, where the client supplies authentication data under demand in a client-driven negotiation.

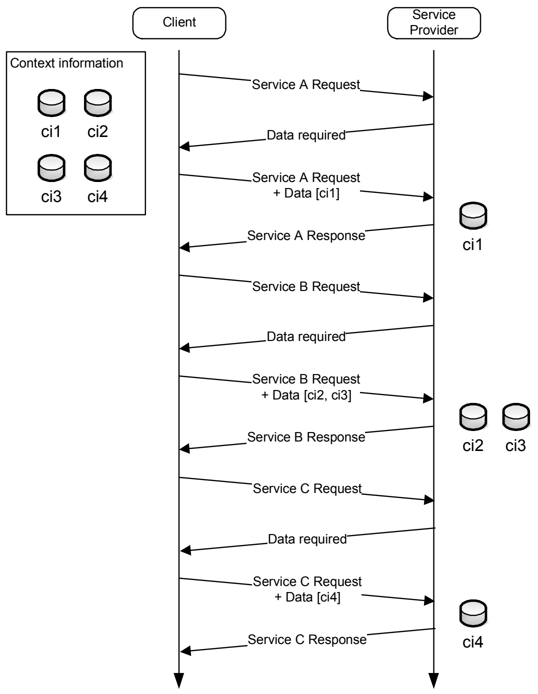

Fig. 1. Service interaction without previous context negotiation

Figure 2 illustrates the same services requests with a previous context negotiation process.

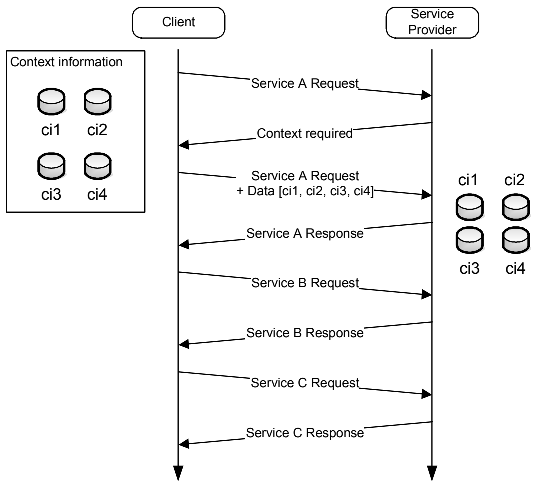

Fig. 2. Service interaction with previous context negotiation

As we can see, context is established in the initial phases of the communication process. The service provider obtains immediately a perfect knowledge about required user information, which can be applied to carry out a personalized service execution. Moreover, the number of interactions decreases dramatically, resulting in saved time and communication efforts.

Of course, these advantages depend significantly in how accurately the user context and preferences information can be identified beforehand. Imprecise negotiation can result in a large amount of unusable exchanged data along with a lack of relevant information that forces extra interactions. How the WebProfiles model represents the user context in the form of preferences is described in the next section.

3 The WebProfiles Markup Language (WPML)

A service or a system can be probably represented at any time via state information, which evolves along the state space that represents all the possible situations under which the service can be found.

After all, expressing and transmitting user preferences is a way of influencing the state of the service or system when interacting with the user [5] to meet his desires or requirements.

But the reality is a bit more complex. Probably the user wants his preferences to be applied in a context-sensitive way, that is, depending on the service actual state or information, the preferences can vary.

Here, we redefine the concept and define *context* as the set of conditions that must be tested and probably fulfilled by the service to activate the user preferences. Thus, the context represents the surrounding information that must be checked to determine the need for setting up some concrete preferences.

On the other hand, we define *configuration* as the set of related preferences that express user requirements or predilections for some features of the service operation.

Finally, we define *profile* as the association of a context to a configuration, that is, the set of conditions under which some preferences must be activated. In fact, an accepted configuration provokes a change in the service state related to the user, creating a *new context* closer to the user's desires, so the whole process can be called *context negotiation*.

Via context negotiation the user (or user-agent) expresses and transmits profiles that must be processed by the target service, influencing its behaviour and state, thus achieving user-aware web services.

For example, a user preference can represent "I want the temperature of my present location to be between 20ºC and 30ºC when outside because out of this range I use to get ill, so this is mandatory, and between 25ºC and 30ºC when at home at night, mandatory too." In this case "temperature of my present location to be between 20ºC and 30ºC" is the preference to be activated in a context "when outside" and "between 25ºC and 30ºC" is a preference to be activated in the context "when at home at night".

Both contexts and configurations are expressed with two complementary mechanisms. First, data structures of XML Data Schemas are used to identify the concepts about which conditions and preferences are going to be expressed. Second, we have developed an XML-based language called WPML (WebProfiles Markup Language) to relate configurations to contexts in which those preferences must be activated, that is, to represent profiles.

In order to express both the context information and the preferences we need to use XML Data Schemas that structure the involved domain of knowledge, maybe the “location” domain, the “time” domain, and the “ambient conditions” domain, which includes the temperature, in the above example. Depending on some characteristics in the location and time domain we want some preferences in the ambient/temperature domain. Since every domain is identified via a unique namespace, no ambiguities must arise when generating our profile.

The above example can be represented in WPML in the following way:

```
<?xml version="1.0" encoding="UTF-8"?>
<wpml xmlns="http://www.webprofiles.org/schemas/wpml10"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.webprofiles.org/schemas/wpml10
  http://www.webprofiles.org/schemas/wpml10.xsd" querylang="xpath">
  <profile uri="urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6">
    <context
      xmlns:loc="http://www.webprofiles.org/dataschemas/location">
      <pattern ID="pat1" use="required"
        match="/loc:location[@loc:type!='Home']"/>
    </context>
    <configuration
  xmlns:amb="http://www.webprofiles.org/dataschemas/ambientconditions
  ">
      <preference ID="pre1" use="required"
        about="/amb:ambient/amb:temperature/text()"
        operator="gt" value="20"/>
      <preference ID="pre2" use="required"
        about="/amb:ambient/amb:temperature/text()"
        operator="lt" value="30"/>
    </configuration>
  </profile>
</wpml>
```

The `<profile>` element contains an attribute `uri`, with a unique universal identifier for referencing this profile and two elements: `<context>` and `<configuration>`. The `<context>` expresses a set of patterns (the technical word we use for conditions) in domains to activate preferences. Those patterns are expressed using XPath and are considered to be fulfilled if the XPath expression yields an object when evaluated. The `<configuration>` element contains the user preferences, addressing them also via XPath but expressing ranges via the `operator` and `value` attributes.

This is a remarkable difference with other systems like CC/PP [12], which merely conveys user-agent information using the classical attribute-value method. In the Web-Profiles model, we can express *ranges* of values that are preferred by the user for a concrete attribute, thus allowing more expressive power about real preferences. We can even represent our desire for a concrete attribute *not to have* a certain value or range, using the MathML-based operators `eq`, `neq`, `gt`, `lt`, `geq` and `leq`.

Of course, we could express our temperature preferences without any condition related to the location. In those cases where preferences are not attached to existing context conditions, the `context` section can be omitted, so that only the configuration information is conveyed. We call this type of profiles, *context-less profiles*.

The "`required`" value at the `use` attribute in the pattern element indicates that the condition must be present and fulfilled, considering it as failed if not present or nor checkable. An "`optional`" value there indicates that the condition must only be fulfilled if present, which is not mandatory

Several patterns must be provided in the same or different domains. For example,

```
<context xmlns:loc="http://www.webprofiles.org/dataschemas/location"
  xmlns:time="http://www.webprofiles.org/dataschemas/time">
  <pattern ID="pat1" use="required"
      match="/loc:location[@loc:type='Home']"/>
  <pattern ID="pat2" use="required"
      match="/time:time[@time:hours<6 or @time:hours>20"/>
</context>
```

With these context patterns, the associated configuration must only be applied if the location type is "Home" and it is sooner than 6:00 or later than 20:00 which can be considered "night time". Both patterns are mandatory to exist and fulfill.

Again, we want to stress that there is a subtle but important difference among the context-related information structures and the "configuration of preferences"-related structures. Context information represents *state information* that the service provider is able to check, either directly from databases or files, or indirectly by requesting the state from some originating sources. In both cases, that state information must be structured in XML format meeting the requirement of an associated grammar, possibly in the form of a XML Schema. That XML formatted state information is the target of the XPath expressions in the context section of the profile. So, we call *context domains* to the set of domains of knowledge the service is aware of.

On the other hand, preferences configuration information represents *domains over which the service keeps control* to make changes to fulfill user preferences and drive the system towards the desired state. The service can implement those changes invoking some low level functions on actuators, or invoking operations on other devices, for instance. The selected mechanism is up to the service and out of the scope of the WebProfiles model. So, we call *configuration domains* to the set of domains of knowledge over which the service keeps control

XPath is the preferred element addressing language as well, but WPML is open for other mechanisms such as XQuery, just establishing the `querylang` attribute at the `<wpml>` element (in our current implementation only XPath is supported).

4 Conclusions and Future Work

The WebProfiles Markup Language constitutes a convenient extension to the HTTP protocol in order to support automatic customization and adaptation in Ubiquitous

Computing environments populated with devices and embedded services, where these are automatically configured to match user preferences and requirements.

Our current implementation of the negotiation mechanism takes the form of a background process for PocketPC which discovers surrounding devices using UPnP, and negotiates user preferences represented via WPML to adapt the environment, made up of small HTTP server-embedded devices.

The WPML documents can be created by the user via UI wizards or inferred from past manual interactions with devices which can be recorded and analyzed by pattern-detection or heuristic tools.

The use of well-known standards such as HTTP, XML or XML Schemas guarantees the stability and coherence of the model itself, while retaining the extensibility that can be added by using accompanying web technologies such as HTTPS. The WebProfiles model relies also on P3P technology for validating the use of the preferences by the device against user privacy policy.

Finally, we think that RDF [14], OWL [15] and other Semantic Web technologies could also be applied to declare the relationships about context conditions and preferences, instead of XML Schemas. XPath in WPML could also be substituted by other semantic alternatives, still under development and sparsely standardized, such as CXPath [16], RDF Path [17] or RPath [13]. SWRL [18] is a suitable alternative for representing rules about context conditions and desired configurations without any need for path-based languages, to create Semantic WebProfiles that would allow the expression of user-preferences by means of their real relationships.

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 Telefónica Móviles España at Deusto University, Bilbao, Spain.

References

1. UPnP Forum. UPnP Device Architecture v1.0. 2000. http://www.upnp.org.
2. Issarny, V., Sacchetti, D. et al. Developing Ambient Intelligence Systems: A Solution based on Web Services. Journal of Automated Software Engineering. 2004.
3. Kristol, D., and Montulli, L. RFC 2965: HTTP State Management Mechanism. IETF RFC. 2000.
4. St. Laurent, S. Cookies. Computing Mcgraw-Hill. 1998.
5. Vázquez, J.I., and López de Ipiña, D. An Interaction Model for Passively Influencing the Environment. Adjunct Proceedings of the 2nd European Symposium on Ambient Intelligence (Eindhoven, The Netherlands). 2004.
6. Hensley, P. et al. Implementation of OPS Over HTTP. 1997. http://www.w3.org/TR/NOTE-OPS-OverHTTP.html
7. P3P Specification Working Group. The Platform for Privacy Preferences 1.1 (P3P1.1) Specification. W3C Working Draft. 2004. http://www.w3.org/TR/P3P11/

8. Coyle, K. P3P: Pretty Poor Privacy? A Social Analysis of the Platform for Privacy Preferences (P3P). 1999. http://www.kcoyle.net/p3p.html
9. Little, M., Newcomer, E. and Pavlik, G. (eds.). Web Services Context Specification (WS-Context). OASIS Committee Draft v.0.8. 2004.
10. XML Protocol Working Group. SOAP Version 1.2 Part 0: Primer. W3C Recommendation. 2003.
11. Web Services Description Working Group. Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. W3C Working Draft. 2004.
12. W3C Device Independence Working Group. Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0. W3C Recommendation.
13. Matsuyama, K. et al. A Path-Based RDF Query Language for CC/PP and UAProf. Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, 2004.
14. RDF Core Working Group. RDF Primer. W3C Recommendation. 2004. http://www.w3.org/TR/rdf-primer/
15. Web Ontology Working Group. OWL Web Ontology Language Overview. W3C Recommendation. 2004. http://www.w3.org/TR/owl-features/
16. Camillo, S.D., et al. Querying Heterogeneous XML Sources through a Conceptual Schema-Conceptual Modeling - ER 2003, 22nd International Conference on Conceptual Modeling. LNCS 2813. Springer. 2003.
17. http://infomesh.net/2003/rdfpath/
18. DARPA DAML. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. Draft version 0.7. 2004.