A Language for Expressing User-Context Preferences in the Web

Juan Ignacio Vázquez
Faculty of Engineering, Deusto University
Avda. Universidades, 24
48007 Bilbao, Spain
+34 944 139000
ivazquez@eside.deusto.es

Diego López de Ipiña
Faculty of Engineering, Deusto University
Avda. Universidades, 24
48007 Bilbao, Spain
+34 944 139000
dipina@eside.deusto.es

ABSTRACT
In this paper, we introduce WPML (WebProfiles Markup Language) for expressing user-context preferences information in the Web. Using WPML a service provider can negotiate and obtain user-related information to personalise service experience without explicit manual configuration by the user, while preserving his privacy using P3P.

Categories and Subject Descriptors
C.2.2 [**Computer-Communication Networks**]: Network Protocols – *applications*

C.2.4 [**Computer-Communication Networks**]: Distributed Systems – *client/server*

H.3.5 [**Information Storage and Retrieval**]: Online Information Services – *web-based services*

General Terms
Management, Performance, Design, Experimentation Standardization.

Keywords
Context-aware, HTTP, profiles, Web, cookies, state management, Ambient Intelligence.

1. THE WEBPROFILES MARKUP LANGUAGE (WPML)
A service or 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 [1] 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 of *context* as the set of conditions that must be tested and probably fulfilled by the service to activate 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 my alias to be 'Mike' and talk in rooms with less than 20 people when surfing sites about music". In this case "my alias to be 'Mike' and talk in rooms with less than 20 people" are preferences to be activated in a context "when surfing sites about music".

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.

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. 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 fulfil user preferences and drive the system towards the desired state. The service can implement those changes invoking some low level functions, updating databases or files, or invoking operations on remote objects via SOAP, for instance. So, we call *configuration domains* to the set of domains of knowledge over which the service keeps control.

In order to express both the context information and the preferences we need to use XML Data Schemas that structure the involved domains of knowledge, maybe the "site information"

WWW 2005, May 10-14, 2005, Chiba, Japan.
ACM 1-59593-051-5/05/0005.

domain, and the "chat" domain in the above example. Depending on some characteristics in the site information domain we want some preferences in the chat 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>
      <context xmlns:site="http://www.webprofiles.org/dataschemas/siteinfo">
          <pattern ID="pat1" use="optional"
              match="/site:site/site:categories[site:category='music']"/>
      </context>
      <configuration xmlns:chat="http://www.webprofiles.org/dataschemas/internetservices/chat">
          <preference ID="pre1" use="optional" about="/chat:chat/chat:userinfo/chat:alias"
              operator="eq" value="Mike"/>
          <preference ID="pre2" use="optional" about="/chat:chat/chat:rooms/chat:room/chat:nusers"
              operator="lt" value="20"/>
      </configuration>
  </profile>
</wpml>
```

The <profile> element contains 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 [2], which merely conveys user-agent information using the classical attribute-value method. In the WebProfiles 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 be* of a certain value or range, using the MathML-based operators eq, neq, gt, lt, geq and leq.

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

```
<context
  xmlns:site="http://www.webprofiles.org/datasche
  mas/siteinfo"
  xmlns:chat="http://www.webprofiles.org/datasche
  mas/internetservices/chat">
  <pattern ID="pat1" use="optional"
      match="/site:site/site:categories[site:cat
  egory='music']"/>
  <pattern ID="pat2" use="required"
      match="/chat:chat/chat:rooms/chat:room
  [contains(chat:topic,'beatles')]"/>
</context>
```

With these context patterns, the associated configuration must only be applied if the site type is "music" and there is at least one chat room with the string 'beatles' in the topic, being this last pattern mandatory to exist and fulfil.

Of course, we could express our chatting preferences without any condition related to the site type, being it "music" or "politics". 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*.

WPML has some additional but powerful features such as variables and complex data structures that can be declared and used as comparison values both in patterns and preferences.

2. CONCLUSION AND FUTURE WORK

The WPML documents can be created by the user via UI wizards or even downloaded from servers that generate them based on user detected preferences. We think that RDF, OWL and other Semantic Web technologies could also be applied to declare the data structures within the context and configuration domains, instead of XML Schemas. XPath in WPML could also be substituted by other semantic alternatives, still under development and sparsely standardized, such as RxPath, RDF Path, CXPath or RPath [4].

The use of Semantic WebProfiles would allow the expression of context patterns and preferences by means of their real relationships, being the desired evolution of the WebProfiles model.

3. REFERENCES

[1] 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.

[2] W3C Device Independence Working Group. Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0. W3C Recommendation.

[3] P3P Specification Working Group. The Platform for Privacy Preferences 1.1 (P3P1.1) Specification. W3C Working Draft. 2004. http://www.w3.org/TR/P3P11/

[4] 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.