An ontology based on the CC/PP framework to support content adaptation in context-aware systems

José Viterbo F, Marco Antonio Casanova, Hana K. Rubinsztejn, Markus Endler

1Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro (PUC-RJ)
R. Marquês de São Vicente, 225
22453-900, Rio de Janeiro, Brasil
{viterbo,casanova,hana,endler}@inf.puc-rio.br

Abstract. The ability to react to dynamic changes of the environment with minimal human interference is a fundamental requirement in ubiquitous applications for mobile computing. A common element in the architecture of such applications is a proxy, an element in charge of executing a number of content adaptations on behalf of one or several client applications running on mobile devices. These adaptations are triggered by specific conditions involving the mobile devices on which the applications execute. This paper proposes an extension to the CC/PP ontology to assist content adaptation in context-aware systems describing not only the static and dynamic characteristics of mobile devices but also the proper conditions that trigger pre-defined adaptations.

1. Introduction

Portable devices such as notebooks, PDA’s and smartphones are becoming more and more popular, as their computational power increases and their prices fall. Moreover, the quick spread of wireless networks has enabled the data access in places as diverse as university campi, airports, coffee houses and family homes. The ubiquitous computing concept, presented by Mark Weiser [Weiser 1991], now sets, as a fundamental requirement, the design of applications capable of responding to dynamic changes in their environments with minimal human interference. Users should be able to take full advantage of the local capabilities within a given environment, and be able to seamlessly roam between several environments, even as resources (like available bandwidth) change [Sousa and Garlan 2002]. With that purpose as target, the development of context-aware systems has drawn great attention recently.

Context-aware applications may be implemented in a number of ways, but the current trend suggests the use of a middleware infrastructure capable of collecting, managing and making available context information about mobile devices and their environment, thus reducing the complexity necessary to develop context-aware applications [Henricksen and Indulska 2005]. Several systems have been proposed in order to provide such infrastructures [Baldauf and Dustdar 2004]. Among them, MoCA (Mobile Collaboration Architecture) [Rubinsztejn et al. 2004] is an architecture that supports the

*Work partially funded by projects CNPq 55.2068/02-2 (ESSMA) and 479824/04-5 (Ed. Universal).
†Author supported by scholarship CNPq 140441/2005-0.
‡Author supported by scholarship CNPq 140249/2003-6.
development of context-aware applications for mobile computing. Besides a set of APIs to build such systems, MoCA provides efficient services to collect, store and distribute context information associated with mobile devices. This information comprises not only raw data related to the device’s resources and the wireless links (802.11), but also the location information inferred based on the 802.11 signal strength values measured from all access points available in the environment.

As a part of MoCA, the ProxyFramework [Rubinsztejn et al. 2005] is being developed to allow the generation of instances of proxies for different context- and location-aware applications. The framework has to be extended and customized to produce concrete proxy instances according to specific application requirements. A proxy is a common element in the architecture of distributed applications for mobile networks. It intercepts the messages exchanged between the mobile clients and servers, and is in charge of executing a number of transformations, adaptations or management functions on behalf of one or several clients, such as content adaptation, protocol translation, caching, personalization, user authentication, handover management, etc. The main advantage of using such an intermediary is to bridge the wired-wireless gap, and make all mobility, connectivity and context-dependent issues transparent to the application developer. Although each distributed application for such networks has specific adaptation and transformation requirements, there are a number of recurrent components and interaction patterns used for implementing the commonly found adaptation and management functions.

Nevertheless, adaptive and context-aware applications need not only an appropriate way for expressing information about device’s characteristics, but also a common representation for exchanging this information among the involved elements. Composite Capabilities/Preference Profiles (CC/PP) [Klyne et al. 2004] defines an XML-based standard for exchanging context descriptions, as well as vocabularies for expressing information related to device capabilities and user preferences. It was originally designed to support content negotiation between Web browsers and servers, so that servers and proxies could customize Web content to the target device’s capabilities.

This work presents an extension of CC/PP, consisting of a set of classes, properties and vocabularies to express a variety of context information types used within the MoCA architecture. We also defined a way to describe the adaptations available to be applied to a given service and how the profiles containing these descriptions may be composed and processed to guide content adaptation.

Section 2 discusses the basic concepts involved in the implementation of the proposed system. Section 3 contains some related work. Section 4 presents our extended ontology. Section 5 discusses the use of the ontology with the help of some examples. Finally, Section 6 brings the conclusions about the proposed system.

2. Basic Concepts

2.1. The CC/PP Framework

The Composite Capabilities/Preference Profiles (CC/PP) [Klyne et al. 2004] is a standard proposed for describing device capabilities and preferences focusing on wireless devices such PDAs and smartphones. Since most adaptations depend only on the capabilities of the devices and the preferences of the users, a CC/PP profile can be used to allow
servers to deliver content specifically adapted for such devices. In fact, it was originally designed for this purpose. When a device makes a request using a protocol such as HTTP, it sends its CC/PP profile in the request. The server can then filter, translate and adapt the content to meet the requirements of the requesting device. CC/PP is based on the Resource Description Framework (RDF) and serialized in the XML format.

A CC/PP profile is constructed as a two-level hierarchy: a profile has a number of components and each component, a number of attributes. Following this scheme, statements are made about the components that have named attributes and values for those attributes. For instance, a profile may specify that a mobile device has a hardware component that has as attributes a screen width resolution whose value is 320 pixels and a screen height resolution whose value is 200 pixels. It often occurs that the profiles of many devices are the same or similar. For this reason, attributes of a component may be included directly in a profile document, or may be specified by reference to a default profile. These default values may be provided by a hardware or software vendor, stored separately in a Web server and accessed via a URL. If the profile received from a client device contains its own values for any of the default component attributes, then non-default values always take precedence. Therefore, clients need only to specify those components and attributes which vary from the defaults, minimizing the use of bandwidth between the device and the source server.

Through the introduction of new vocabularies, new attributes can be defined as needed. Similarly, new relationships can be introduced through new vocabulary items. However, the introduction of such relationships must be carefully conducted to ensure that their semantics are adequately and consistently defined. New vocabularies are introduced through XML namespaces. Their relationship to other CC/PP vocabulary items can be defined through new RDF schema statements. Reusing existing vocabularies leverages work that has already been undertaken and reduces the opportunity for different attribute names to be quasi-synonyms. As names using different namespaces may be freely mixed in a profile, introducing additional attributes and relationships does not require the definition of a complete new vocabulary.

2.2. The MoCA Architecture

The MoCA architecture provides support for developing and executing distributed context-aware applications, particularly those that comprise mobile devices interconnected through wireless infrastructured LANs (802.11b/g). The services provided by MoCA support the collection, distribution and processing of context information acquired directly from the mobile devices or inferred through context services. In addition, MoCA offers a set of APIs to support the design and implementation of context-aware applications that use the MoCA’s services.

Among the services and functionalities provided by MoCA, a Context Information Service (CIS) is responsible for collecting, storing and processing raw context data probed at the mobile devices, and a Location Inference Service (LIS) is responsible for inferring the location of a mobile device from the information about RF signal patterns collected from reference points [Nascimento et al. 2006]. A service called Monitor must be executed on each mobile device to collect raw context data and send it to CIS. Table 1 shows the most relevant context variables managed by CIS and available for querying.
<table>
<thead>
<tr>
<th>Tag</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Integer</td>
<td>CPU usage (in percentage)</td>
</tr>
<tr>
<td>FreeMemory</td>
<td>Long</td>
<td>Total memory available (Kbytes)</td>
</tr>
<tr>
<td>EnergyLevel</td>
<td>Integer</td>
<td>Energy level available (in percentage)</td>
</tr>
<tr>
<td>LinkQuality</td>
<td>Integer</td>
<td>Quality of network connection</td>
</tr>
<tr>
<td>APMacAddress</td>
<td>String</td>
<td>MAC address of current access point</td>
</tr>
<tr>
<td>OnLine</td>
<td>Boolean</td>
<td>True if the mobile device is online</td>
</tr>
<tr>
<td>IPChange</td>
<td>Boolean</td>
<td>True if the device changes its IP</td>
</tr>
<tr>
<td>APChange</td>
<td>Boolean</td>
<td>True if the device changes access point</td>
</tr>
</tbody>
</table>

Table 1. Context information items provided by CIS

Besides context management, CIS and LIS also implement context monitoring, allowing clients to register their interest in specific context states (involving one or several context variables) modeled as logical expressions, and to be asynchronously notified whenever the corresponding context-expression is satisfied [Sacramento et al. 2004]. This functionality reduces the cost to build client applications, since they are relieved from managing the context information delivery.

2.3. The ProxyFramework
MoCA’s ProxyFramework [Rubinsztejn et al. 2005] is a white-box framework for developing and customizing proxies according to the specific needs of a context-aware application. It facilitates the programming of distributed, self-adaptive applications for mobile networks, where adaptations should be triggered by context-change events. The proxy not only intermediates the communication between the application server and its mobile clients, but also serves as the interface with MoCA services. The ProxyFramework is intended to accommodate a number of basic management and adaptation functions that an application proxy might be required to execute on behalf of each of its mobile clients. In fact, the ProxyFramework defines only abstract interfaces of proxy components and templates describing how these components interact. In order to implement application-specific adaptation and management functions, these components have to be extended or specialized by the application developer.

Application developers must also define the set of conditions under which a proxy must apply the adaptations it is able to provide. At runtime, the proxy decides which adaptors must be used based on the context information and the predefined conditions. As CC/PP was designed to guide the adaptation of content delivered by servers to mobile devices, its use seems to be adequate to provide a common structure to describe the conditions that trigger proxy adaptations, according to specific vocabularies properly defined.

3. Related Works
The User Agent Profile (UAProf) [WAPForum 2000] is a concrete implementation of CC/PP developed by the Open Mobile Alliance (OMA) aimed at WAP-enabled mobile terminals. For instance, the UAProf specification defines HardwarePlatform, SoftwarePlatform and BrowserUA to be subclasses of the generic CC/PP component class. The UAProf specification also defines attributes for each of these specialized classes.

Indulska et al. (2003) presented another ontology model based on the CC/PP standard, together with a context management system that supports a pervasive system
infrastructure. This work defined a set of CC/PP components and attributes to express a variety of context information types and relationships between context descriptions, and also an extended vocabulary describing a wide range of context information. Such components include network interfaces of devices, quality of service, location, connection status and application requirements, among other features. This context information was used by the proposed infrastructure to support a wide range of adaptation techniques used by applications when their computational context changes.

To overcome the limitations of the two-level hierarchy in CC/PP, this extension proposed that some of the described components may have attributes that are themselves components. Figure 1 shows an example where the NetworkInterface component is modeled as an attribute of the standard UAProf NetworkCharacteristics component. The NetworkInterface component has an attribute called QoS, which itself is a structured attribute with three attributes, PacketLoss, Bandwidth and Latency. The value of each structured attribute is a URI that points to the complete description of the complex attribute. Usually, the URI is a fragment identifier that points to another CC/PP component within the same profile.

4. The Proposed Ontology

Following the W3C recommendation to improve interoperability, we built our ontology by extending the one proposed in [Indulska et al. 2003] into three directions. First, we introduced new vocabularies to represent the context information provided by MoCA. Second, we proposed a way to represent the dynamic behaviour of some attributes. Finally, we extended the ontology to also describe the dynamic adaptations of a given service.

4.1. The MoCA Vocabulary

Our first concern was to create an ontology appropriate to be used within the MoCA architecture, so we defined an extended vocabulary to include new components and attributes
to represent the type of context information managed by MoCA. For that purpose, we defined the new component class *DeviceStatus* associated with the attribute classes *CpuUsage*, *FreeMemory* and *EnergyLevel*. We have defined the attribute class *APMacAddress* associated with the component class *NetworkInterface*, and the attribute class *LinkQuality* associated with the component class *QoS*. We defined *IsOnLine*, *IPChange*, *APChange* and *Roaming* as attribute classes related to the component class *DisconnectionStatus*. Finally, we defined *SymbolicLocation* as a new component class and *SymbolicArea* as an attribute class related to this component class.

### 4.2. Dynamic Attributes

Our next step was to add to the ontology a way of stating that some attributes are dynamic and, like the default description of components, their values must be acquired somewhere.

![Figure 2. Extended ontology to represent the dynamic behaviour of attributes](image)

We then proposed the new basic class *DynamicBehaviour*, with data type properties *hasSource*, *hasTag*, *hasInitialValue* and *hasProtocol*, that completely describe how the value of a dynamic attribute may be obtained. In our ontology, this new class is related to the *Attribute* basic class by a *hasDynamicBehaviour* property, as shown in Figure 2. Since the previously defined classes and properties are maintained, an application that is not able to cope with dynamic attributes (that is, not enabled to query CIS, for instance) still can process a description to get the static capabilities.

### 4.3. Describing Adaptations

In our approach, the adaptations are implemented at the application level, which means that application developers must define a set of conditions under which a given adaptation must be applied. After defining the states that are of particular interest for a given application (acceptable conditions for the network, device, user, etc) and the necessary or desired adaptations related to them, the next task of the application developer should be to correlate the proper context information to the specific adaptations.
The decision of which adaptors to use — and when — is made by the proxy at runtime, but must be based on a set of predefined rules that indicate the trigger conditions related to context values and the proper adaptations to be executed. In fact, the decision to apply a given adaptation is directly related to the values held by the attributes of a profile, that is, the particular states defined by a subset of context variables. Since these variables are fully represented in a profile description, we extended the ontology to be able also to describe the adaptations that are associated with a given service or application, so that the conditions under which they must be applied are defined using a common vocabulary.

Figure 3 shows this new structure. We created the new basic classes Adaptation, Action and AttributeBinding. A given profile may contain any number of Adaptation instances that represent the adaptations that are available to be used for a certain application. Each Adaptation instance can take one on more actions (Action instances), which represent the responses an adaptation implementation may give when certain conditions hold. To describe conditions, which are modeled as logical expressions involving one or more context variables, we used the class QuotedExpression, imported from OWL-S. This class allows the description of expressions containing logical and comparison operators using the Semantic Web Rule Language (SWRL) [Horrocks et al. 2004]. The class AttributeBinding is used to associate each variable that appears in such expressions with the Attribute instances that represent the same context variables.

5. Profile Processing and Inference

A proxy capable of processing a profile described according to the proposed ontology must have a specific module responsible for this operation. When a mobile device running
a client application wants to interact with a server application, this module must be able to get all pieces of profiles that describe this interaction from the specific URIs and build a complete profile.

These pieces of profile may be related to the device or network capabilities, user preferences, client and server application capabilities, adaptations available to a specific service, etc, and may comprise default component descriptions and dynamic attributes managed by a context service. Therefore, the processing of the profile must be able to retrieve all descriptions, including defaults, and to periodically query the MoCA’s context service to get the updated values of dynamic context variables. Figure 4a shows the example of a hypothetical profile retrieved by a proxy involving a mobile device and a specific slideshow client running on that device. This description states that the device does not have a color display and that the client application supports .JPG and .GIF formats, but not the .TIF format. Fig. 4b shows the adaptations related to the slide show service that may be executed by the correspondent proxy. By interpreting this description, the processing of the profile would allow to decide that the ImageConversion adapter must be instantiated to compose the specific proxy. In addition, based on both descriptions, an inference engine is be able to indicate that ConvertColorToBW and ConvertTifToJpg adaptations should be applied in this case.

![Figure 4. An example of profile containing only static attributes](image)

The previous example shows a static situation though, since all adaptations are triggered by the static capabilities of a mobile device. On the other hand, our second example involves attributes whose values change dynamically. In Figure 5a, we see the profile describing the characteristics of the mobile device, in Fig. 5b, we see a profile describing the adaptation available to a hypothetical video streaming service, and in Fig. 5c, we see the adaptation available at a hypothetical WAP messaging service.
As in the first case, in both applications, the processing of the profile can determine the adaptors that must be instantiated, and also the context variables of interest to be monitored. As both involve dynamic attributes, there are two ways in which the proxy can decide when specific adaptations must be applied. Through synchronous communication (request-reply), the proxy can query the CIS server periodically for context variables updates, change the stored profile and then infer the applicable adaptations at a given point. The second option is simpler and uses the inference services provided by MoCA. Based on the description of the attribute values or range of values that trigger an adaptation, the proxy can subscribe an expression based on several context variables at the CIS and be asynchronously notified when a predefined condition holds.

6. Conclusions

The CC/PP framework was first designed to support content negotiation and adaptation between Web browsers running on mobile devices and servers, so that servers and proxies could customize content to target devices with limited resources. It was supposed to describe just the static characteristics of the devices themselves and of the operating systems and browsers involved. Previous works [Indulska et al. 2003] have extended this standard to embrace more complex context information, involving, for instance, network QoS and geographic location, but these extensions only represent static aspects of attributes.

We extended the initial CC/PP framework to propose an ontology more adequate to guide content adaptation in context-aware systems whose general conditions change dynamically. We detach two main contributions of this work. First, by introducing a new class and new properties we provided a way of representing the dynamic behaviour of some attributes. Second, we defined a set of classes and properties to describe the

![Diagram](image-url)

**Figure 5. An example of profile containing dynamic attributes**
adaptations available to a given service, and how these adaptations are related to the observable context variables. Then, with the resulting ontology, we provided a generic way of describing adaptations that may be applicable in scenarios in the scope of the MoCA architecture, involving specific devices, client applications and services. Proxies that may interpret and reason about profiles described using the proposed ontology — with the support of the MoCA context services — are thus able to dynamically adapt the content provided by a server to clients executing on different sorts of mobile devices.

References


