Context Modeling for Device- and Location-Aware Mobile Web Applications

Michael Hinz Zoltán Fiala
Dresden University of Technology, Department of Computer Science
Heinz-Nixdorf Endowed Chair for Multimedia Technology
D-01062, Germany, +49-351-463-38516

{michael.hinz, zoltan.fiala}@inf.tu-dresden.de

ABSTRACT

Ubiquitous Web systems have to deal with varying context information in order to support context awareness. Accomplishing this requirement necessitates gathering, processing and representing that information, so that it can be used for adapting Web applications. In this paper we propose a mechanism for modeling dynamically changing context information like user's device capabilities and location. Furthermore we show how this modeling mechanism can be integrated into context-aware system architectures. With the aid of this approach we are able to perform a device- and location-aware online document generation of Web documents for mobile appliances.

Categories and Subject Descriptors

H.5.4 [**Hypertext/Hypermedia**]: Architectures. H.3.4 [**Systems and Software**]: User profiles and alert services.

General Terms

Management, Measurement, Design.

Keywords

Context modeling, context-awareness, device independence, ubiquitous computing, location-based services

1. INTRODUCTION

Today's WWW emerges to a medium of communication and cooperation. A multiplicity of users with different requirements uses the Web as a ubiquitous information store. In this way several trends such as personalization [1] and device independence [2] have raised that require the adjustment of Web applications to the user and its client device. Furthermore, with the emergence of location-based services, the client location becomes important context information, too. Still, today's location-based services (e.g. navigation system, location-aware museum guides) are fast-paced but usually stand alone applications not suited for the common Web. Furthermore their development is primarily driven by the mobile telecommunication industry and is therefore limited to applications on mobile phones. We claim that the main reason for this shortcoming is the still lacking support for automatically gathering and representing device and location context data. By improving the access to such context information the richness of communication in human-computer interaction increases, and makes it possible to produce more useful Web services [6]. To meet this challenge, this paper introduces context modeling mechanisms which enable the development of personalized device- and location-aware Web applications at the same time.

The paper is structured as follows. After addressing general context modeling aspects (section 2), modeling mechanisms for device capabilities (section 2.1) and the user's location (section 2.2) are illustrated. Beyond that, methods and techniques for sensing and representing that context information are presented. The described mechanisms were successfully integrated into a Web system performing a dynamic generation of context-aware Web pages (section 3).

2. CONTEXT MODELING

In the literature several definitions of the terms context and context-awareness can be found (e.g. [3], [4]). However, adding context to Web applications requires on the one hand the acquiring and modeling of context information such as location [5] as well as device context [2]. On the other hand that information has to be stored in a standardized representation. Without this application developers are left to develop ad hoc and limited schemes for storing and updating context information. However, the evolution of more sophisticated representations enabling a wider range of capabilities and a true separation of sensing context from the programmable reaction to that context is still an unsolved challenge [6]. To ensure this, the proposed modeling mechanisms represent the gathered information in a profile-based extensible context model ([14]) which relies on the W3C standard CC/PP [7]. Modeling components (e.g. for device, location and user modeling) guarantee a permanent monitoring of context and updating of the corresponding context profiles.

2.1 Device Modeling

Because of the requirements of mobile phones and resource restricted handheld computers their in- and output interfaces are limited e.g. in display size or number of buttons. Therefore they are far away from suitable interfaces to the Internet. Overcoming this burden requires the creation and publication of content that is tailored to users' platform capabilities and dynamically changing device properties. Therefore device capabilities have to be acquired, represented and provided to the document generation engine in the context aware system. In the next chapters those aspects will be explained more detailed.

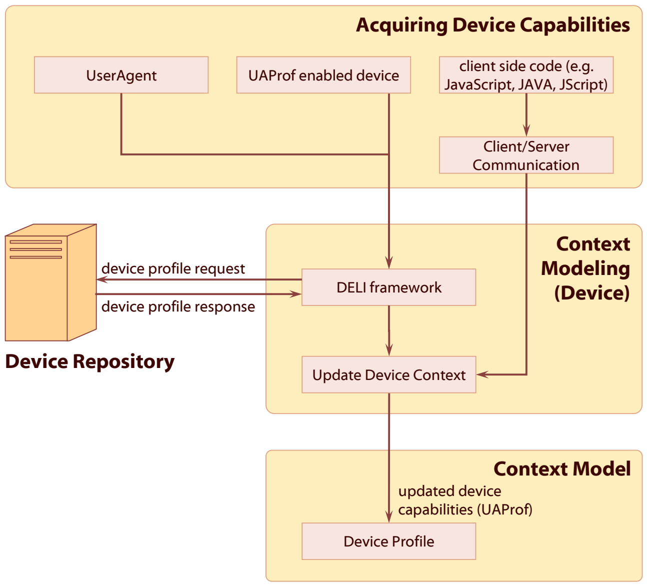
Acquiring Device Capabilities
UserAgent
UAProf enabled device
client side code (e.g. JavaScript, JAVA, JScript)
Client/Server Communication
Context Modeling (Device)
device profile request
device profile response
DELI framework
Update Device Context
Device Repository
Context Model
updated device capabilities (UAProf)
Device Profile

Figure 1: Modeling device capabilities

2.1.1 *Acquiring Device Capabilities*

Several strategies exist to acquire device capabilities for device sensitive Web applications. The most popular method is to analyze the HTTP user-agent parameter that comes with the HTTP request and map this parameter to a device or browser repository on the server side. However, this works only for nearly static device properties. Using the User Agent Profile specification (UAProf [8]) basing on the CC/PP framework [7] establishes a more effective mechanism for gathering dynamically changing device properties on the server by analyzing the UAProf enabled request. However, this specification only provides a common vocabulary for WAP devices. But most of the vocabulary can be adopted for other non WAP devices like normal Web browsers on desktop computers, notebooks or PDAs. In our work we extended the vocabulary in order to support those device classes. Further on for those device classes we provide a mechanism to transfer the gathered device capabilities within the HTTP request to the server.

In this way our device modeling mechanism illustrated in Figure 1 distinguishes between UAProf enabled devices, devices providing the user agent and devices giving support for client side code fragments like JavaScript, Jscript and Java (combinations are possible). Those client side code fragments are included during the Web document generation on the server and directly gather device properties on the client. The gathered information is encoded in a UAProf like representation and integrated in the HTTP request by a client/server communication component for processing that information on the server.

2.1.2 *Processing and Representing Device Context*

According to the gathered capabilities from the client, the server processes the corresponding device context. The processed context is represented as the device profile in the context model (see Figure 1 and Figure 3). The representation is based on the above mentioned extended UAProf format. The processing of the device context on the server depends on the obtained request.

1. If the request only includes the user-agent parameter this parameter is mapped to the according device profile in a device repository. Note that by using only this mechanism dynamically changing device properties (e.g. bandwidth or size of the browsers window) can not be taken into account.

2. If a UAProf enabled device sends a user-agent profile or a difference profile within the request, that information is handled by DELI [9] on the server side which provides an API for Java servlets to determine client capabilities using CC/PP and UAProf. The output of the DELI component makes a profile representing UAProf properties available.

3. Whereas today nearly only WAP 2.0 devices support UAProf, our system is also able to autonomously collect the devices properties of other end devices (e.g. Notebook, PDA) via the above mentioned client side code. The on the client gathered properties are sent within the HTTP request. Our server processes that information and merges it with an existing or by DELI generated device profile.

Together these mechanisms enable to acquire permanently changing device properties. The result is an always up-to-date device profile of the context model on which the document generation is based on.

2.2 Location Modeling

The trends of offering wireless networks and mobile internet access for a wide variety of mobile devices establish new areas of Web applications taking the location of the user into account. Since the location is important context information that changes whenever the user moves, reliable mechanisms for location-tracking are needed. Since normally all contents of a hypermedia Web application are delivered by Web servers, the location information is needed by the server which adjusts the contents according to the user's location. Generally there are two different strategies for obtaining the user location. It can be tracked by processing location data on a server (e.g. third party location server like Openwave Location Manager [13]). Another strategy is to obtain location information from a device-based application and transfer it to the server. Our concept for modeling the location of a client device supports both strategies. Figure 2 shows both scenarios using a location server abstracting from complex locating hardware for obtaining the client location.

2.2.1 *Locating Methods*

Locating methods can be categorized according to many aspects. Today many mobile devices (PDAs and Notebooks) use *satellite supported technologies* like GPS modules, which are either built in or available as extensions. Other devices like cellular phones (GSM and UMTS) or devices powered by radio networks (WLAN) are always connected to a base station handling a cell for mobile devices. So another technology for sensing the location is the use of *network supported cell information*. Generally it strongly depends on the application category which type of location processing should be used. E.g. because of the low signal intensity the conventional GPS does not work *indoors* (e.g. for museum guides or exhibition navigators) and is therefore restricted to *outdoor applications* [6]. Other aspects for a location method categorization are the supported dimension, accuracy, reliability and the positioning model of the method.

This shows that supporting location-based services in Web applications that are not limited to only one location method strongly requires the use of standard location protocols. Therefore in our concept for modeling the user location we use the Mobile Location Protocol (MLP [10]) standardized by the Open Mobile Alliance (OMA). This protocol enables a communication between location-based services and location infrastructure by means of XML interfaces. Figure 2 shows our architecture concept for modeling location information and storing that in the environment profile of our context model.

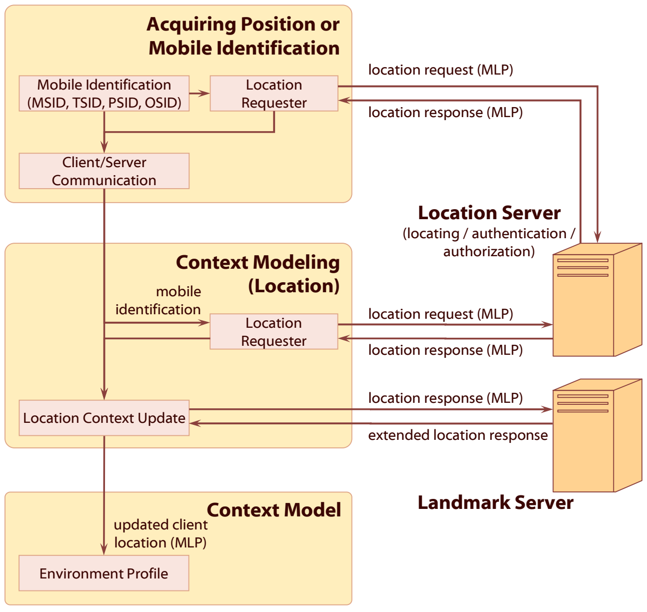

Figure 2: Modeling location and location context

2.2.2 *Acquiring location information*

As one possibility the architecture supports the acquisition of user location on client devices supporting the J2ME location API (JSR 179 [11]). This location API defines an optional package (javax.microedition.location) that enables developers to write wireless location-based applications and services for resource limited devices such as mobile phones, and can be implemented for any common location method [12]. Because the API implementation works on the client device the obtained location information has to transfer to the server for a server based generation of location-aware Web pages. This can be done by integrating that information in the HTTP request through the client/server communication component.

Another possibility to gather location information on the client is to receive the position of the user from a location server which analyzes network supported cell information according to the client mobile identification. In our proposed architecture (Figure 2) a location requester generates an MLP-request containing the mobile identification either in form of a North American Mobile Identification Number (MIN) or a GSM Mobile Subscriber ISDN (MSISDN) which are simply the mobile telephone numbers of the mobile subscriber. The following code snippet shows an example of a possible MLP-request generated by the location requester:

```
<svc_init ver="3.0.0">
  <hdr ver="3.0.0">
    <client><id>…</id><pwd>… </pwd></client>
  </hdr>
  <slir ver="3.0.0">
    <msids><msid type="MSISDN">1234567</msid></msids>
    …
  </slir>
</svc_init>
```

The MLP-response of the location server (e.g. OpenWave [13]) to the client could look like the following XML representation:

```
<svc_result ver="3.0.0">
  <slia ver="3.0.0">
    <pos>
      <msid type="MIN">3035551003</msid>
      <pd>
        <time utc_off="+0000">20050108014000</time>
        <shape>
          <CircularArea>
            <coord>
              <X>39 51 14.399N</X>
              <Y>105 02 53.858W</Y>
              <Z>5280</Z>
            </coord>
            <radius>1000</radius>
          </CircularArea>
        </shape>
        <speed>42.42640687119285</speed>
        <direction>45.0</direction>
      </pd>
    </pos>
  </slia>
</svc_result>
```

The determined position information is included in the next request to our server, which can use this information for generating location-aware Web documents. Again, this is done with the same mechanism of the client/server communication component that was already mentioned in Section 2.1.1.

Moreover the designed architecture supports the location acquisition on the server with a similar mechanism. In this case the client transmits its mobile identification directly to the server, which is responsible for the communication with the location server itself.

2.2.3 *Processing and Representing Location*

The location representations gathered either from the client or the server are stored in the environment profile of our context model. However, the location is represented by coordinate system-based position information which is difficult to use for adaptation processes because the author of a Web page who has to specify the adaptation rules is normally not familiar with it.

The solution is a landmark server storing landmarks which are locations associated with names and descriptions. The granularity of the landmarks depends on the Web application type. An outdoor route planner application for instance has the granularity of addresses (streets, house numbers) whereas an indoor application like a museum guide has a finer granularity.

The landmark server maps the location contained in the MLP-response to associated names and descriptions of that location. By grouping of locations different location granularities are taken into account. After the mapping the extended response is used to update the environment profile of the context model (Figure 2).

For that purpose the MLP extension mechanism is used. The following example shows how the response is extended:

```
<svc_result ver="3.0.0">
  <slia ver="3.0.0">
    <pos>…</pos>
    <LocationExtension>
      <LocationMark>
        <Group>MMTChair</Group>
        <Name>Room 202</Name>
        <Description>Projekt Amacont</Description>
      </LocationMark>
    </LocationExtension>
  </slia>
</svc_result>
```

The extended MLP-data is represented in the context model and can be used for performing adaptation processes.

3. INTEGRATION INTO A CONTEXT AWARE SYSTEM ARCHITECTURE

The proposed context modeling mechanism assures an always up-to-date context model containing information about the user's device and location. Furthermore the context model includes additional information about the user which can be used for personalization aspects. For a detailed introduction in those aspects and user modeling process the reader is referred to [14].

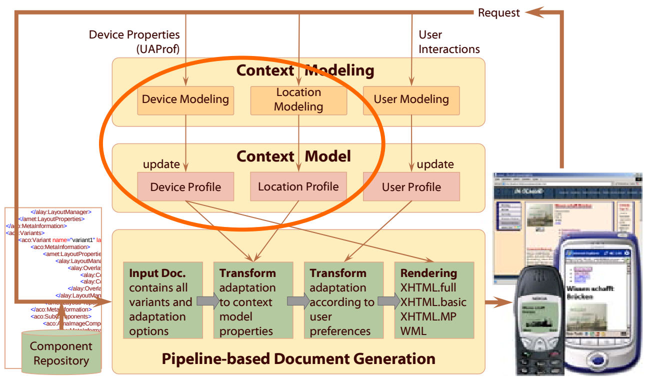
Request
Device Properties (UAProf)
User Interactions
Context Modeling
Device Modeling
Location Modeling
User Modeling
Context Model
update
update
Device Profile
Location Profile
User Profile
Input Doc. contains all variants and adaptation options
Transform adaptation to context model properties
Transform adaptation according to user preferences
Rendering XHTML.full XHTML.basic XHTML.MP WML
Component Repository
Pipeline-based Document Generation

Figure 3: Context aware system architecture

Figure 3 shows how the proposed context modeling mechanisms can be integrated in an architecture for generation of context-aware Web pages. Note that the device and location modeling mechanisms (see Section 2) are located in the highlighted architecture components (compare Figure 1 and Figure 2). For each user request, a complex document is retrieved from a component repository. According to the context model, this document is adapted to user, device and location properties and rendered to a specific output format (XHTML, cHTML, WML etc.). Proving feasibility that overall architecture was implemented on the basis of the XML publishing framework Cocoon.

4. CONCLUSION AND FUTURE WORK

The work we proposed enables Web system architectures to generate context-aware Web pages adapted to users' device capabilities and location information. Therefore we presented a mechanism for modeling that dynamically changing context information and show how it can be integrated in a context-aware system architecture. Future work will concentrate on distributing the context model to enable the user to decide which of the gathered information she/he is willing to share with the server architecture. Another aim is the performance enhancement of the proposed architecture since generating context-aware Web documents makes high demands on online Web publishing systems.

5. REFERENCES

[1] Fink, J., Kobsa, A. *A Review and Analysis of Commercial User Modeling Servers for Personalization on the World Wide Web*. User Modeling and User-Adapted Interaction 10 (2–3), 2000, pp. 209-249.

[2] Butler, M., Giannetti, F., Gimson, R., Wiley, T. *Device Independence and the Web*. IEEE Internet Computing, Vol. 6, No. 5, Sep.-Oct. 2002, pp. 81-86.

[3] Dey, K., Abowd, G. D. *Towards a better understanding of context and context-awareness*. In Workshop on "The What, Who, Where, When, and How of Context-Awareness", Conference on Human Factors in Computer Systems (CHI2000), 2000.

[4] Dey, A. K. *Understanding and using context*. Personal and Ubiquitous Computing, Special issue on Situated Interaction and Ubiquitous Computing 5, 1, 2001.

[5] Abowd, G., Mynatt, E. *Charting past, present and future research in ubiquitous computing*. ACM Transactions on HCI, Special issue on HCI in the new Millenium, 7(1), March 2000, pp. 29-58.

[6] Abowd, G., Mynatt, E., Rodden, T. *The Human Experience*. IEEE Pervasive Computing Magazine, 1(1), January-March 2002.

[7] Klyne, G., Reynolds, F., Woodrow, C., Ohto, H., Hjelm, J., Butler, M., Tran, L. *Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0*. W3C Recommendation, January 2004.

[8] Wireless Application Group. *User Agent Profile Specification*. Open Mobile Alliance WAP Forum, 2001.

[9] Butler, M. *DELI: A DElivery context LIbrary for CC/PP and UAProf*. HP, External Technical Report HPL-2001-260 (revised version 02/08/2002), 2002.

[10] Open Mobile Alliance. *Mobile Location Protocol (MLP)*. Enabler Release Definition for Mobile Location Protocol (MLP) Candidate Version 3.1, March 2004.

[11] *Location API for J2ME (JSR 179)*. Java Specification Request, September 2003.

[12] Mahmoud, Q. *J2ME and Location-Based Services*. Technical Articles of the Sun Developer Network, March 2004.

[13] OpenWave. Location Studio SDK http://developer.openwave.com/dvl/tools_and_sdk/openwave_mobile_sdk/location_studio/

[14] Hinz, M., Fiala, Z., Wehner, F. *Personalization-based Optimization of Web Interfaces for Mobile Devices*. In Proceedings of the MobileHCI 2004, Glasgow, Scotland, September 2004.