A FRAMEWORK FOR CONTEXT MODELING IN ADAPTIVE WEB APPLICATIONS

Michael Hinz, Stefan Pietschmann, Zoltán Fiala
Dresden University of Technology
Department of Computer Science, Chair for Multimedia Technology
D-01062 Dresden, Germany
{michael.hinz, stefan.pietschmann, zoltan.fiala}@inf.tu-dresden.de

ABSTRACT

Modern Web systems that provide content to heterogeneous user groups using different devices have to deal with varying context information to support context-awareness. Accomplishing this requirement necessitates sensing, processing, and representing that information. Still, up to now there is a lack of reusable solutions for efficient context management. Therefore, this paper presents a component-based extensible framework for modeling context information that can be effectively used for adapting Web applications. The framework allows for faster development and deployment of context-aware adaptive Web applications and provides a set of modeling components supporting important application domains such as location-based services, device independence, and personalization. Furthermore, in order to support the addition of further context modeling techniques, it provides generic extension mechanisms.

KEYWORDS

Context-aware, Adaptive, Device Independence, Personalization

1. INTRODUCTION

Today's Web turns from a conventional presentation medium to a personalized ubiquitous environment that supports mobile applications. In order to avoid the development of monolithic solutions being dedicated to specific Web applications, there is a need for generic and extensible mechanisms for sensing, processing, and representing the context information used in those scenarios (Bardram, 2005). Current research in context-aware computing mainly focuses on sensor technologies, smart environments, and the underlying technical infrastructure (WWRF, 2001). Still, there is a lack of comprehensive development support in the area of context-aware Web applications. An integrated approach which takes the physical context of the user, his behavior, his needs and preferences, but also the characteristics of the networks and devices he uses into account, is still missing (WWRF, 2001). Thus, frameworks are needed that can help programmers to develop and deploy context-aware applications more efficiently.

To meet this requirement, this paper introduces a context modeling framework that offers a significant benefit to developers of context-aware Web applications, such as location-aware, device-aware, and personalized Web applications. The framework supports ubiquitous Web scenarios by providing a generic and extensible basis for handling context information. Aided by this solution, providers only have to maintain their Web content and its adaptation according to the available context information. The sensing, processing, and representing of context information is provided by the framework.

The rest of this paper is structured as follows. After a discussion of related work (Section 1.1) Section 2 gives an overview of the framework and its context model that acts as an interface for adaptation modules in a Web system. Section 3 introduces components for modeling different types of context information. It is shown how application domains such as location-awareness, device independence, and personalization can be supported. Section 4 describes the extensibility of the framework by introducing abstract interfaces on different framework layers. Finally, Section 5 shows the upgrading of a Web system to a context-aware system using the presented framework.

1.1 Related Work

Recently, significant research has been done in the area of context-aware computing. Besides *formalizations* of the term context (e.g. (Dey, 2001a), (Chen, 2000)) as well as surveys of context modeling solutions (e.g. (Strang, 2004)), there are a lot of approaches initiated by the pervasive computing community for *sensing context information* (Schmidt, 2002). Furthermore, there is already a number of standards available for *representing context information* ((Klyne, 2004), (Strang, 2003), (WAG, 2001)). Still, (WWRF, 2001) criticizes the missing support for context-awareness by an integrated approach that manages context information. To close this gap, many context-aware systems use context or user *modeling servers* or other infrastructure components (Fink, 2000), (Hohl, 2002). Those servers offer an abstraction layer that provides support for acquiring and processing context data. However, in (Bardram, 2005) several restrictions of these approaches are mentioned, such as single point of failure, lacking scalability, and extensibility. Furthermore, we see restrictions in the cases when context information has to be sensed on the client device that is not in direct contact with the modeling server.

As an alternative, *context frameworks* can speed up the development and deployment of context-aware applications. The Context Toolkit (Dey, 2001b) is a framework that supports the rapid development of a wide range of context-aware applications by using context widgets that sense context data and refine low-level data into higher-level knowledge about the context. Another middleware is the Context-Aware Toolkit that provides tools for the quick development and deployment of context-aware applications for mobile or even non-mobile devices (CAT, 2006). JCAF provides a generic, extensible, and expressive Java programming model for context-aware applications and context models (Bardram, 2005). However, all these approaches primarily focus on native code applications dedicated to the area of context-awareness. None of them takes the specifics of Web applications into account (e.g. heterogeneous client devices, limited device capabilities, client/server and distributed system architectures, network communication between client device and server). Therefore, this paper takes the lessons learned from the pervasive and context-aware community and introduces an extendable framework for modeling context in adaptive Web applications.

2. OVERVIEW OF THE CONTEXT MODELING FRAMEWORK

Providing an integrated approach for context-aware Web applications necessitates generic mechanisms for gathering, processing, and representing context information, so that it can be effectively used for adaptation. Therefore, the architecture of our framework is divided into three different layers: the *sensor layer*, the *context modeling layer*, and the *context model layer* (see Figure 1). The sensor layer encapsulates *sensor components* that gather and monitor different properties of the user's context (e.g. user location, device capabilities, user interactions). This sensor information is processed by the *context modeling components* of the second layer in order to discover also implicit information and represent it in a sophisticated way. Depending on the type of context this can be a complex task that takes external data sources or even the execution of complex learning or other modeling algorithms into account. For instance, the location modeling component described in this paper senses the user's location via a GPS sensor and produces additional semantic information characterizing the acquired location and surrounding objects. Section 3 presents several example context modeling components that are already implemented and integrated into the framework. Since these components only cover selected application areas and not all domain specific types of context data, the context modeling framework also provides extension mechanisms.

In order to be accessible by various Web applications, the resulting context data has to be represented in a sophisticated way. Therefore, an extensible *context model* stores all gathered and processed data in different context profiles (see Figure 1) that are often associated with various application scenarios. The context model is structured according to CC/PP (Composite Capability / Preference Profiles), an RDF grammar for describing device capabilities and user preferences in a standardized way (Klyne, 2004). Depending on the requirements of a Web application predefined profiles can be used. The concrete vocabulary which can be used in a profile is declared by a profile scheme (RDF or XML schema). By adding new profile schemes the context model can be extended arbitrarily (see Section 4). Figure 1 shows some of the predefined context profiles. Technical properties and capabilities of users' client devices are stored in the *device profile* on the basis of an extended version of the WAP User Agent Profile UAProf (WAG, 2001) providing a common

vocabulary for WAP devices and other mobile devices (e.g. PDAs) as well (Hinz and Fiala, 2005). The *location profile* represents the location of the user based on the Mobile Location Protocol MLP (OMA, 2004). The location is stored in the form of a coordinate system-based position enriched with additional semantic information that is associated to the position or its environment. Based on developed XML schema definitions the *user profile* of the context model is divided into two parts. The first part (*identification profile*) contains information to identify users. Besides a set of general properties (name, email etc.), arbitrary extensions are allowed. The second part (*preference profile*) contains user preferences, for instance (see Section 3.3) in a rule based representation based on the CDL4 algorithm (Shen, 1996). Furthermore, there exists a number of other profiles for storing low-level context information that can be processed in order to establish enhanced context information. For instance, the *session profile* integrates user interactions by grouping them to page requests and sessions. Based on this interaction history the user modeling process generates new knowledge about the user in terms of rules that are stored in the *preference profile*. Similarly, the history information can be used to generate a *long term profile* that combines information about all users of the system. E.g. an application specific stereotype membership of a user can be represented by this profile in order to reduce server load by handling groups of users together.

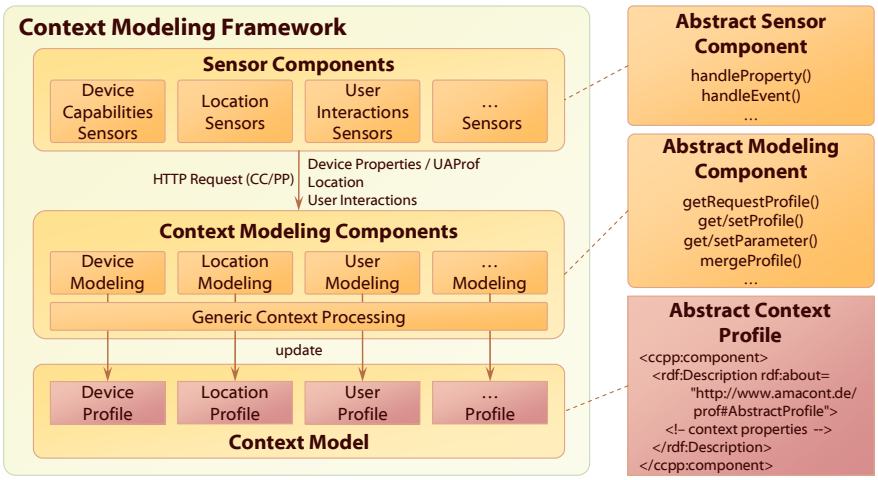

Figure 1. Overview of the context modeling framework

3. FRAMEWORK COMPONENTS FOR CONTEXT MODELING

This section outlines example components provided by the framework that aid the modeling of different types of context information. They support important application domains such as location-based services, device independence, and personalization.

3.1 Location Modeling Components

The trend of offering wireless networks and mobile internet access for a wide variety of mobile devices establishes new areas of Web applications that take the location of the user into account. Since this location changes whenever the user moves, reliable mechanisms for location-tracking are needed. To fulfill this task the presented context modeling framework provides location modeling mechanisms that can be effectively used to develop and deploy location-based services.

For obtaining the user's location different sensor components exist (Figure 2) or can be added using the framework's extension mechanisms (section 4). Depending on the used position determination technique the sensors work either client- or server-sided. Firstly, the framework provides a *GPS Adapter* (Java Applet) that accesses a GPS receiver via Bluetooth, permanently obtains the position provided by the GPS receiver, and communicates it through the client-server communication manager (section 4) to the server. Secondly, a *JSR-179 Adapter* is in implementation that uses the optional J2ME Location API (JSR-179) to obtain the user's location directly through a JSR-179 ready device. As a third possibility, an implemented *MLP Adapter* (can

work both on the client or the server) provides a sensor component that abstracts from the concrete position determination technique (Hinz and Fiala, 2005). Note that by using the MLP standard (OMA, 2004) any third party MLP location server (e.g. Openwave Location Manager) can be applied to obtain the user's location.

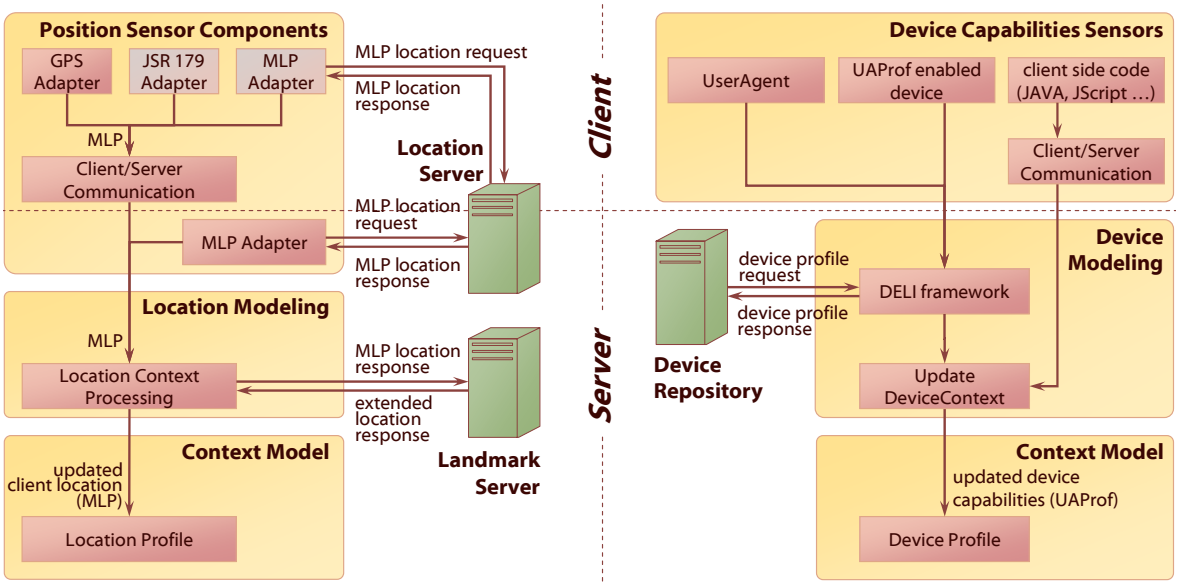

Figure 2. The location (left) and the device (right) modeling mechanism

Since the location sensed by the location sensors of the framework is provided in the form of coordinate system-based position information, it is still difficult to use it for adaptation processes in the authoring process of location-aware Web applications. To enrich the coordinate system-based position by semantic information a Location Context Processing Component acting as a "reverse geocoder" associates landmarks to the user's actual position. The resulting *semantic location* is stored in extension to the already available position in the location profile and can be effectively used for adaptation purposes. Note that the granularity of the landmarks depends on the actual Web application, i.e. the processing of the semantic location is based on different application specific landmark stores. For instance, while an outdoor route planner application has the granularity of addresses (towns, streets, and house numbers), an indoor application like a museum guide has a finer granularity. Taking the granularity of landmarks into account the framework provides a markup scheme that enables the hierarchical definition of landmarks. The following code snippet shows an instance of our markup scheme that represents countries, towns, and important places. Furthermore Figure 3 (left) shows a screenshot from a prototypically implemented online video store that relies on that code. Since the actual user is located in the area of the Dresden University of Technology, the application offers a list of documentation movies about this university (shown under the satellite map).

```
<LandmarkStore>
    <semLoc name="Germany"> <desc>country</desc>
        <semLoc name="Dresden">
            <semLoc name="TUD"> <desc>Dresden University of Technology</desc>
                <circle x="51.032" y="13.858" radius="1000"/>
            </semLoc>
            <semLoc name="Frauenkirche"> <desc>church</desc>
                <rect x1="51.052" y1="13.74" x2="51.058" y2="13.75"/>
    ... ... ... </semLoc>
    <semLoc name="Spain">  ...  </semLoc>  ...
</LandmarkStore>
```

3.2 Device Modeling Components

In today's mobile Web, delivery of content that is tailored to users' platform capabilities and dynamically changing device properties is essential. Typically, the in- and output interfaces of restricted handheld devices are limited, i.e. they are far away from suitable interfaces to the Internet. Therefore, the context framework

offers *device capabilities sensors* and *device modeling components* for processing the *device context* and representing it as the *device profile* in the context model (Figure 2).

The device capabilities sensors implement several strategies for determining platform properties (Hinz and Fiala, 2005). Firstly, there exists a sensor component for acquiring (mostly static) device properties by analyzing the *HTTP user-agent parameter* (delivered with the HTTP request). Secondly, a *UAProf-based sensor* was developed, allowing to gather device properties by analyzing the requests of UAProf enabled clients (User Agent Profile (WAG, 2001)). Finally, a third mechanism is based on *client side code fragments* written in JavaScript, Jscript and Java. Such code fragments can be included into Web documents during their generation on the server in order to directly gather even dynamically changing device properties on the client. The gathered information is encoded in a UAProf like representation and is integrated into the HTTP request by the same client/server communication component that is used by the location sensors.

Besides the sensor components the framework offers device modeling components that process the gathered capabilities and store them in the device context that is based on an extended UAProf specification (compare Figure 1 and Figure 2). These components use the DELI framework (Butler, 2002) which provides an API for Java servlets to determine client capabilities using CC/PP and UAProf. Furthermore, a mechanism for mapping user-agent parameters to an appropriate device profile in a device repository was developed, too. A more detailed description of the device modeling mechanism can be found in (Hinz and Fiala, 2005).

Figure 3 shows a movie store prototype (created within the scope of the AMACONT project (Fiala, 2003)) that takes different device capabilities into account that is. The left screenshot shows the application on a high resolution display that allows to utilize a layout with three columns. In the middle figure the layout is adjusted to the available presentation space by using a two-column layout. Parts of the third column are integrated into the second one. Finally, the right figure shows the presentation on a PDA. Due to the technical restrictions only one column is available. Furthermore, optional parts of the presentation are hidden or replaced by simpler interaction elements (e.g. the navigation menu is exchanged by a text-based link list).

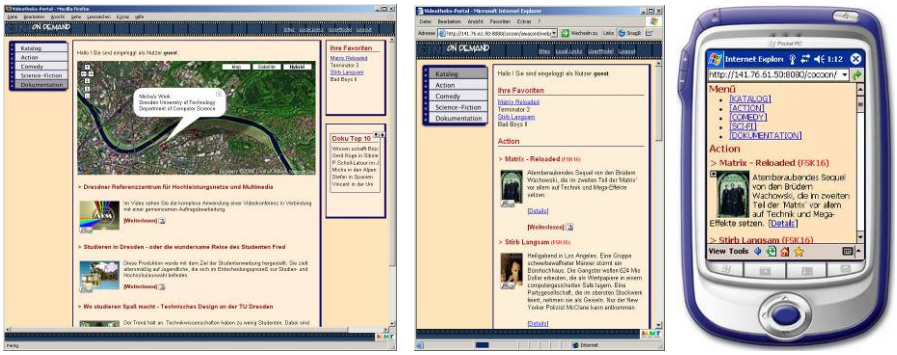

Figure 3. Prototype of an online movie store basing on the device and location modeling mechanisms of the framework

3.3 User Modeling Components

In this section two components for user modeling are introduced. While the first one allows for the explicit updating of context model parameters by using so called Request Processing Rules, the second one provides an implicit user modeling mechanism based on the incremental learning algorithm CDL4 (Shen, 1996).

Generic Request Processing Component: The context parameters maintained by the context modeling components shown above can be mostly acquired and updated automatically. However, if an adaptive Web application is expected to take into account different characteristics (e.g. knowledge, preferences etc.) of its users, as well, then the maintenance of this context information is typically very domain specific, i.e. it should be explicitly specified by the author(s) of the Web application. In order to support their processing in a generic way, our framework provides a generic request processing component. This component was developed as part of the Self-Adapting Generic Adaptation Component (SAG (Houben, 2005)), a tool for adding adaptation to Web applications. Consequently, it can be flexibly configured to explicitly update context model parameters according to user requests. The component is configured by so called *Request Processing Rules*. These rules are triggered at each user request and allow for creating new context parameters or modifying existing ones. The core of a Request Processing Rule is the assignment of a new

value to a context model parameter. This new value can be simply a constant but it can be also an arbitrary complex arithmetical term based on request parameters, existing context model parameters, constants and arithmetical operators. Optionally, a Request Processing Rule can be bound to a condition; in this case it is performed only if that condition is valid.

User Preference Modeling Component: Adapting content to dynamic user preferences can be effectively used for optimizing Web pages on mobile devices (Hinz, 2004). However, capturing such preferences in order to support personalization is a serious problem if we do not want to explicitly ask the user for his/her preferences. For that purpose the framework provides an implicit user modeling component. It allows observing users' browsing behavior by client-side sensor components that track interactions being performed on media assets (e.g. video, audio, image, text). The captured user events are stored in the session profile of the context model. Based on that profile and the semantics behind the media assets the user modeling component performs a complementary discrimination learning algorithm (CDL4 (Shen, 1996)) that analyses the interactions and calculates a preference profile of the user. A more detailed description of the user modeling mechanism can be found in (Hinz, 2004). As an example Figure 4 shows a possible sequence of automatically generated XHTML documents in an online movie store. In the left picture the user enters the default version of the page containing no detailed information. When the user is interested in getting enhanced information about a movie, she/he can enlarge the title picture or uncollapse a toggle-text providing more detailed text description (Figure 4, middle). As mentioned above, this interaction is captured by client-sided sensor components and sent to the server. Based on this interaction the CDL4 algorithm calculates a preference profile as part of the context model. This context information can be used for adapting the presentation if the user enters the same or any other page containing the movie list again (Figure 4, right).

Figure 4. Sequence of a browsing session powered by the implicit working user modeling component

4. EXTENDING THE FRAMEWORK

Developing and deploying context-aware applications can be significantly facilitated by reusing the already implemented context modeling mechanisms. As described in the previous section, our framework provides a set of generally applicable context modeling components that serve different adaptation purposes (e.g. device independency, location-awareness and personalization). However, since each adaptive Web application might have its domain specific requirements towards the representation and maintenance of its context model, it isn't possible to provide predefined components for all conceivable adaptation scenarios. Therefore, extension mechanisms on each layer of the framework allow to create dedicated context modeling solutions.

Abstract Context Profile: Chapter 2 already described the context profiles provided by the framework that can be used for a wide range of adaptive Web applications. These profiles together set up a CC/PP based context model (Klyne, 2004). Extending the context model necessitates to add a new CC/PP profile that is able to represent the required new context information. This can be done by providing a grammar (e.g. RDF schema or XML schema) that defines the vocabulary of the new type of context.

Abstract Modeling Component: For the development of new modeling components the framework provides an abstract Java class from which they can be derived. The class interface offers generic methods and fields for receiving sensor data and for updating the existing context model by e.g. simply overwriting parameters (or profiles) or by merging existing profiles together.

Abstract Sensor Component: For adding new sensor components to the framework two strategies can be used. Sensors working on the server side can be derived from an abstract sensor component. This abstract

Java component offers an interface that allows to access the sensed context data via getter methods. As an example, this concept is used by the server-side MLP adapter of the location modeling (see Figure 2). However, most types of context information can not be gathered on the server side. For instance, observing the user's position via a GPS receiver or capturing his/her interactions that do not cause a server request is only possible on the client. For supporting such scenarios the presented framework provides a client-server communication manager that enables the registration of various client side sensor components. Once a sensor component is registered to the manager it can use an interface for communicating properties (`handleProperty()`) or events (`handleEvent()`) acquired on the client. The manager automatically collects all sensed data and communicates it to the server within the next server request without affecting any other communication. The client-server communication manager was implemented in JavaScript and Jscript (for PDAs). Since those technologies do not support inheritance strategies the framework provides a prototype script that can be used as a starting point for developing new sensor components. Furthermore, an adapter is provided, too, that supports the binding of sensor components based on Java applets. This way developer can use a more complex programming language for providing new sensor components.

5. INTEGRATION INTO ADAPTIVE WEB SYSTEMS

The presented framework was realized based on the Apache Cocoon XML Publishing Framework but can be easily ported to other solution using JSP technology. The Abstract Modeling and Sensor Component (and therefore all other modeling and sensing components) are derived from a Cocoon Action. On the client side there is support for JavaScript, Jscript and JavaApplet implementations. In the near future there will be also support for client side sensors based on Java Midlets (J2ME). The context model (that contains all known context information of a user) is integrated in the session context and can be used by an adaptive Web system in order to adjust Web content according to that information. As a proof of concept, the context modeling framework was integrated into the modular system architecture of the AMACONT project and was successfully applied in different application scenarios.

First, Figure 5 shows how it is used in the pipeline-based adaptation process of AMACONT's component-based adaptive Web documents (Fiala, 2003). Secondly, it is also successfully utilized by the generic transcoding tool (Fiala, 2005), for adding adaptation and interaction processing functionality to existing Web applications. Furthermore, based on this context framework, adaptation of component-based 3D applications is supported (Dachselt, 2006). As an example Figure 5 (see the two screenshots on the right) shows a device adaptation which replaces a complex ring menu for selecting chairs on the notebook by a floating menu on the PDA because of resolution restrictions.

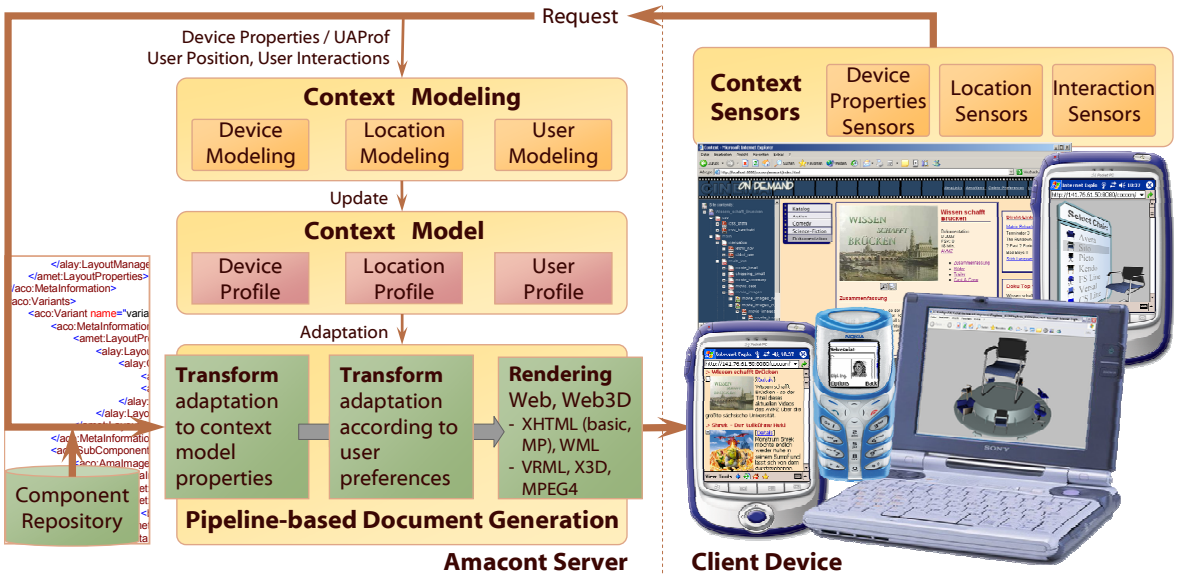

Figure 5: Integration of the context framework into an adaptive Web system

6. CONCLUSION AND FUTURE WORK

In this paper a component-based extensible framework for context modeling was introduced. Being an integrated solution for sensing, modeling and representing context information, it provides a sound basis for the development of adaptive and context-aware Web applications. We explained an implemented set of reusable modeling components supporting common adaptation scenarios, such as location-awareness, device independency, and personalization. Furthermore, generic extension mechanisms supporting the integration of additional context sensing and modeling techniques to the framework were described. Our experiences with the framework proved its support for the development of context-aware systems.

Future work will concentrate on providing a broader set of sensors and modeling mechanisms and on the management support of context information in the authoring process of adaptive Web applications. Currently we work on the integration into Web architectures to use the presented context framework and additional adaptation mechanisms for adapting 3D Web and rich media applications represented by various single or compound media formats. Furthermore, another main activity will focus on the distribution of context models (or profiles) between different servers and clients. Thus, users will be able to decide which context information they are willing to share with the server architecture.

REFERENCES

BARDRAM, J. E. (2005) The Java Context Awareness Framework (JCAF) - A Service Infrastructure and Programming Framework for Context-Aware Applications. Pervasive 2005. Munich, Germany.

BUTLER, M. (2002) DELI: A DElivery context LIbrary for CC/PP and UAProf. HP, TR HPL-2001-260 (02/08/2002).

CAT (2006) CAT: Context Aware Toolkit (University of Oregon): http://www.cs.uoregon.edu/research/wearables/CAT/.

CHEN, G., KOTZ, D. (2000) A survey of context-aware mobile computing research. TR2000-381. Dartmouth College.

DACHSELT, R., HINZ, M., PIETSCHMANN, S. (2006) Using the Amacont Architecture for Flexible Adaptation of 3D Web Applications. Proceeding of the international conference on 3D Web technology (Web3D). Columbia, USA.

DEY, A. K. (2001a) Understanding and using context. Personal and Ubiquitous Computing, Vol. 5 (1).

DEY, A. K., SALBER, D., ABOWD, G. D. (2001b) A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. Human-Computer Interaction (HCI) Journal, Vol. 16 (2-4).

FIALA, Z., HINZ, M., MEIßNER, K., WEHNER, F. (2003) A Component-based Approach for Adaptive, Dynamic Web Documents. Journal of Web Engineering, Volume 2 (1-2), 58-73.

FIALA, Z., HOUBEN, G. J. (2005) A Generic Transcoding Tool for Making Web Applications Adaptive. Conference on Advanced Information Systems Engineering (CAiSE'05). Porto, Portugal.

FINK, J., KOBSA, A. (2000) A Review and Analysis of Commercial User Modeling Servers for Personalization on the World Wide Web. User Modeling and User-Adapted Interaction, Volume 10 (2-3).

HINZ, M. & FIALA, Z. (2005) Context Modeling for Device- and Location-Aware Mobile Web Applications. 3rd International Conference on Pervasive Computing (Pervasive 2005), PERMID 2005. Munich, Germany.

HINZ, M., FIALA, Z., WEHNER, F. (2004) Personalization-based Optimization of Web Interfaces for Mobile Devices. Human Computer Interaction with Mobile Devices and Services. Glasgow, Scotland.

HOHL, F., MEHRMANN, L., HAMDAN, A. (2002) A context system for a mobile service platform. Architecture of Computing Systems: Trends in Network and Pervasive Computing. Karlsruhe, Germany.

HOUBEN, G. J., FIALA, Z., SLUIJS, K., HINZ, M. (2005) Building Self-Managing Web Information Systems from Generic Components. Workshop on Adaptive and Self-Managing Enterprise Applications. Porto, Portugal.

KLYNE, G., REYNOLDS, F., WOODROW, C., OHTO, H., HJELM, J., BUTLER, M., TRAN, L. (2004) Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0. W3C Recommendation 15 January (2004).

OMA (2004) Open Mobile Alliance: Mobile Location Protocol (MLP). Candidate Version 3.1.

SCHMIDT, A. (2002) Ubiquitous Computing - Computing in Context. Ph.D. thesis at the Lancaster University.

SHEN, W. M. (1996) An efficient Algorithm for Incremental Learning of Decision Lists. TR USC-ISI-96-012, University of Southern California.

STRANG, T., LINNHOFF-POPIEN, C. (2004) A Context Modeling Survey. Workshop on Advanced Context Modelling, Reasoning and Management as part of UbiComp 2004. Nottingham, England.

STRANG, T., LINNHOFF-POPIEN, C., FRANK, K. (2003) CoOL: A Context Ontology Language to enable Contextual Interoperability. Distributed Applications and Interoperable Systems (DAIS2003). Paris, France.

WAG (2001) Wireless Application Group: User Agent Profile Specification. Open Mobile Alliance WAP Forum.

WWRF (2001) The Book of Visions 2001 - Visions of the Wireless World (V1.0). Wireless World Research Forum.