A versatile context management middleware for mobile Web-based information systems

Martin Hitz
Stefan Plattner

University of Klagenfurt, Austria
martin.hitz@uni-klu.ac.at, stefan.plattner@edu.uni-klu.ac.at

Abstract

A versatile middleware for the development of context sensitive Web applications is presented which is supposed to free developers from the great effort of managing several context dimensions such as location and time. The expected benefits are shorter time to market and reduced development costs. The middleware is currently employed in the course of the development of a prototypical campus information system at the University of Klagenfurt.

Keywords: mobile Web information technology; location based services; personalized services

1 Introduction

Due to the forthcoming introduction of the third generation mobile networks such as UMTS and the growing success of wireless LANs, the domain of mobile and personalized software applications is receiving more and more attention. Experts and business managers consider them to become the so badly needed “killer applications” which are expected to return the heavy investments taken so far (cf. Zipf et al. 2001).

To be able to build smart personalized applications or deliver personalized content in general, detailed and accurate data about the context in which the target user employs her application is needed. There are virtually infinite areas for meaningful application of personalized services, and the field of travel and tourism is among the most promising, where many practical uses can be envisaged within different application categories: mobile personalized services, location based services, geographic information systems, billing services, mobile e-commerce, or tracking services.

The gathering, management and especially the provision of context information presents an additional burden to the development of such applications. This clearly indicates that there is a strong need of a common, versatile software middleware which takes care of the details of the basic technical and administrative tasks inherent to the field of personalized and context based applications, thus significantly reducing the effort required for application development.

In this paper, the main concepts and building blocks of such a middleware (XdC – conte**X**t **d**educed **C**ontent) are presented which is currently being employed for a context-sensitive, Web-based campus information system for the University of Klagenfurt.

The most significant features of XdC can be summarized as follows:

- Tracking of a system user's *context* considering detailed information regarding four orthogonal context dimensions (cf. Section 2).
- Personalization and filtering of *content* by matching a user's context with *context pattern* augmented content (cf. Section 3).
- Selection of context dependent content: XdC provides an interface for the user to navigate and select from the content that is associated to the user's current context (whenever this association is ambiguous).
- Explicit navigation in the context space: By default, the user is presented a selection of the content that is available for the current context. But if the user wants to know what content is available at adjacent points in the context space (i.e., nearby locations, upcoming dates or other roles the user may take), the user is able to change his current context at will (cf. Section 4).
- Maintenance of user preferences with respect to context and content categories. This gives a user the possibility to restrict context dependent content to preferred categories, subjects or fields of interest only. This feature can be used as the basis to manage and select personalized tours through museums or similar sites.
- Explicit consideration of hardware independence regarding context tracking technologies and the client devices (cf. Section 5).
- Modular, distributed, Web-based design.

2 Context Information

To be able to deliver personalized content, the context information on which the customisation process of personalization is essentially based needs to be administered.

In particular, XdC distinguishes between four different context dimensions:

1. The spatial dimension ("Where are you?")
2. The temporal dimension ("When did you ask for information?")
3. The social dimension ("Who are you?" i.e., the user's current role)
4. The technical dimension ("Which device are you using?")

The combination of instances of all four dimension classes establishes a *fully qualified context* (cf. Figure 1) and characterizes an user's current situation. The following subsections deal with the individual context dimensions of XdC.

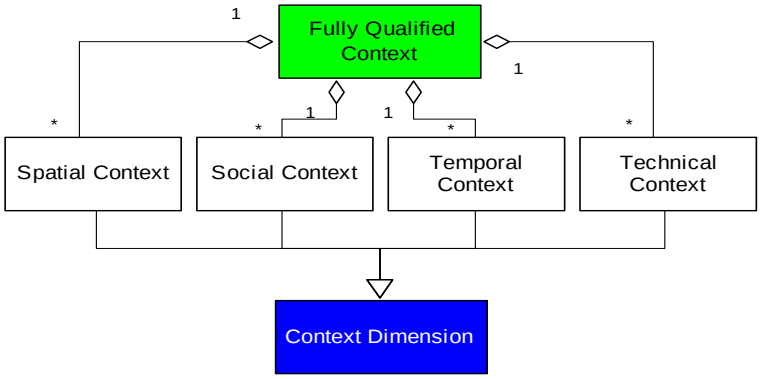

Fig 1: Context Dimensions in XdC

2.1 Spatial Dimension

The perhaps most significant context dimension is the spatial dimension. The knowledge about a user's location opens manifold opportunities and represents the base for a whole class of applications already well known as "location based services" - a class of services to which very high expectations are linked.

Due to the importance of the spatial dimension, much effort and research has already been invested into the manifold problems of location tracking. However, to date no ultimate solution and no technical standard has emerged yet – thus, a context management middleware must be flexible enough to deal with several competing technologies.

For outdoor applications with moderate requirements of spatial resolution, the Global Positioning System (GPS) has become the de facto standard to gather positional information. If there is a clear view to the sky and signals from at least 3 satellites can be received, a GPS-receiver's position can be determined within 10 meters in all three dimensions and – besides the rather cheap client hardware – at no additional cost. Unfortunately, GPS does not function within buildings or other scenarios in which no free sight to the sky is possible. For such situations (e.g. indoors) several other technical approaches already exist, but vary widely in cost, scalability and accuracy. To make this clearer, an exemplary selection of already well known location tracking

systems (cf. Hightower and Borriello 2001) and their major aspects is summarized in Table 1.

Technology		**Accuracy**	**Scale**	**Cost**	**Limitations**
GPS	Radio time of flight lateration	1-10m (95-99%)	24 satellites worldwide	€100 / receiver	Not indoors
Active Badges	Diffuse infrared cellular proximity	Room size (100 %)	1 base per room	Administration costs, cheap tags and bases	Sunlight and fluorescent light interferences
Active Bat	Ultrasound time of flight	9 cm (95 %)	1 base per 10 m²		Ceiling sensor grid required
Cricket	Proximity lateration	2 x 2m (100 %)	1 beacon per 4m²	€10 beacons and receivers	No central management of position information
MSR Radar	802.11 RF (WLAN) scene analysis and triangulation	3-5m (50 %)	3 bases (access points) / floor	802.11 WLAN installation, ~€100 / NIC	802.11 infrastructure required

Table 1: Location tracking systems (exemplary selection)

Thus, a context management middleware which is to be employed in different application domains must support not only one but many different tracking technologies.

The data managed in the spatial dimension of XdC is independent of the location tracking approach used and represents information about *semantic locations* (e.g., "Office E.2.80", "in front of painting "Mona Lisa", "inside St. Steven's Cathedral"). The association of semantic locations to tracking system dependent *physical positions* is maintained by a so-called "spatial driver" module. It is also possible to use multiple such modules to integrate different location tracking technologies within one application. This could be used to overcome the often rigid limitations of specific technologies (e.g., GPS only outdoors).

2.2 Temporal Dimension

From a technical point of view, the temporal dimension is relatively easy to handle. The problem of tracking time has been solved since ages and there are multiple effective solutions with nearly arbitrary grade of accuracy.

However, similar to the spatial dimension, XdC distinguishes semantic times, as for example "work day" and "free time". It is the mapping between semantic times and

absolute time information which represents the main problem in this case and may be defined in manifold ways. This mapping is handled by a so-called “temporal driver” module which in the most trivial case uses the system time of the application server for reference.

2.3 Social Dimension

Since information systems are used by people, XdC of course also manages information about the users of the system. The framework in the current state considers users with all their common properties (e.g. name, nationality, age, gender etc.) and also the different roles (e.g. tourist, guest, technician, employee etc.) a user may play.

XdC in its core however does not determine the way of user-authentication. Techniques like simple login with a password, smartcard or even biometric authentication can be used by integrating a corresponding “social driver” module.

2.4 Technical Dimension

The nowadays strong growing variety of heterogeneous mobile clients (e.g. organizers, mobile-phones, notebooks etc.) and their various networking capabilities cannot be ignored by serious personalized information services. It makes no sense at all to send a video-stream in CIF resolution and 5.1 surround sound to a mobile phone with a postage stamp like display and only very limited bandwidth. Personalized applications should only deliver content that is compatible to a user’s technical context and in essence the technical context determines the way information content is to be rendered or if it – or parts of it - is to be rendered at all.

To give context-sensitive information services based on XdC the chance to suit their content to a user’s mobile device, the middleware has to keep track about a user’s technical capabilities. In particular, XdC manages information about a user’s hardware (device type, screen size, audio capabilities etc.), the software (operating system, browser type and capabilities etc.) and since it is targeted at Web-based information systems, there is also information about a client’s networking capabilities (bandwidth, latency etc.).

3 Context and Content

The central idea behind XdC is to associate a user’s – in this case four dimensional – context with an information system’s content. As XdC is Web based, content can be virtually everything which is presentable to the user by the client specific rendering software. This can range from a static text document to a full featured multimedia

enhanced Web application which may contain complex logic and also exploits the context information gathered, managed and provided by XdC (cf. Figure 2).

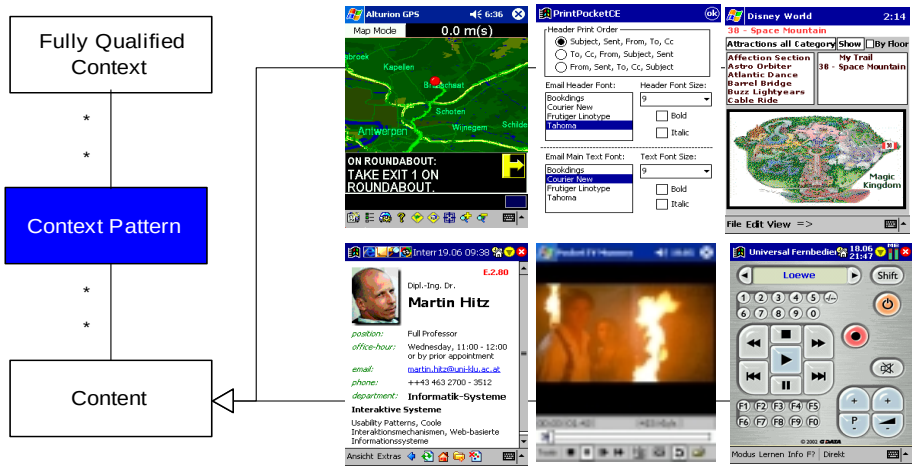

Fig 2: Association of Context and Content

3.1 Association of Context and Content

As the content is seen as external to XdC, content is only loosely coupled to its context information. In essence, XdC only keeps the information needed to access the content, i.e., the content's **U**niform **R**esource **I**dentifier (URI, aka URL). The major advantage of this approach is that beside the requirement that the content be Web-addressable, no further restrictions apply (cf. Pradhan et al. 2001).

However, having only information about the content's Web address, it would be impossible to decide whether the content is suitable to the user's current context or not. To this end, XdC features a flexible way to associate context and content. While it would be possible to directly associate a user's *fully qualified context* with content that should be available in this very specific situation, this approach in practice is very cumbersome. The reason is that a fully qualified context (cf. Section 2) describes the context state of a user in the very detail and this in most cases is really too restrictive. In the normal case, there is the need to control the availability of content only with respect to instances of some of the four context dimensions. For example, there could arise the demand to restrict the availability of a video to a specific location and only to client's that are technically capable to display it, without specifying further requirements regarding the social or the temporal dimension. To be able to do this in a versatile and flexible way, the concept of *context patterns* was introduced into XdC. Context patterns allow to describe such partial context states in a way that it is

possible to decide whether a context pattern matches a user's current fully qualified context or not. So, when integrating new content into XdC's information base, it is crucial – and one important step of personalization – to augment the information about the content with reasonable context patterns.

3.2 Provision of Context Information

The main task of the XdC middleware is to determine which content out of a large content-pool is suitable to the current context of a particular user. This is done by trying to match the user's fully qualified context with the context patterns associated to the content. After this filtering process, the resulting selection – the so-called *conteXt* d*educed* C*ontent* (hence the name) – is presented to the user who then decides which content he likes to see or consume. So a user selected application is started by the user from within XdC. To give the content – in this case active Web applications – the opportunity to access and exploit context information managed by XdC, a handle identifying the user's current session is passed to it at start-up. With this session handle the content is capable of accessing the context information over a well defined API.

Beside information about a user's current context, it is also possible to access a user's context history which directly provides information about the way a user got into his actual state. As the middleware especially features the possibility to also change the current context manually – e.g., shift from room A to room B without actually moving – (cf. Section 4), the system also explicitly maintains information about states which are reached by this convenience function of the system. At least for security sensitive applications it may be important to know if a user really is in the context currently claimed or not.

The eventual degree of personalization depends on both, the amount of context patterns augmenting the XdC information base and the content's application logic which is external to the system but may exploit the context information of XdC.

4 Context Graphs

One of the main features of XdC is the user managed shift of context. This feature allows a user to choose from content available not only for his current context, but also for "adjacent" context. In the spatial context dimension this may be content associated with a nearby location or object. Similar effects can be achieved in other dimensions, although this feature of the XdC user interface has only limited usage with respect to the technical dimension, but at least it is very useful for testing different technical configurations in the phase of content development.

Whether two context instances are considered adjacent to each other is a non-trivial issue. Considering the spatial context dimension it might on the first glance be possible to determine nearby locations by just examining their physical coordinates, but unfortunately there are multiple problems with this approach. For instance, some location tracking systems do not provide world coordinates on which a neighbour search can be based. But even if full GPS-coordinates (latitude, longitude, altitude) are used for every location in the system, a topologically nearest neighbour can in practice be far away if there is such a nasty thing like a concrete wall separating the two. Such semantic problems are even more obvious if one is thinking about neighbourhood relationships in the remaining context dimensions where in general no such information like physical coordinates is available. To tackle these problems and to be able to deal with the notion of adjacent context, the conceptual model of context graphs has been introduced into XdC.

Context graphs are undirected, hierarchical graphs which not only model the neighbourhood but also a hierarchical parent-child or contained-in relationship. Thus, it is possible to express hierarchies of context instances which are used by the framework for organization and smart navigation in these graphs. Furthermore, the edges of the a context graph may be weighted which is useful to order the relation and a nice way to further influence the graph based navigation.

5 Dealing with Heterogeneity

Two major requirements which influenced the design of XdC are the following:

1. XdC should not be bound to a specific application domain.
2. XdC should not be bound to specific tracking technology or client hardware.

To converge to these difficult requirements, several particular techniques are used. A brief overview regarding these techniques is given in the following subsections.

5.1 Tracking Technology

As context tracking is an inherently hardware dependent task, XdC has to feature a way to access the technology dependent data without introducing static dependence to the tracking hardware (cf. Section 2). To be able to do this, the hardware dependent parts are integrated into XdC as interchangeable “driver modules”. There is at least one driver module for each of the four context dimensions whereby in essence each driver fulfils two major functions. Obviously one function is to encapsulate the hardware dependent code which is needed to gather the technology specific data. After data acquisition, the second function of each driver is to abstract the raw (or physical) data by translating it to semantic context instances recognized by the XdC

core. For example, a driver for the temporal dimension could gather precise data about the current time by querying a nearby ntp-server and then translating this raw data to corresponding semantic time instances which may be "Wednesday afternoon" or "holiday".

In addition, to support several complementary technologies, it is also possible to have several driver modules for one single context dimension.

5.2 Client Support

Adapting to different client types is based on XLS transformations. To be able to support multiple client types, there has to be at least one XSL style sheet for each device type. If multiple style sheets for one device type are available, the user may choose a specific style sheet according to her preferences.

As the market of mobile devices is constantly booming, XdC also features the possibility to serve heterogeneous clients within one single application domain. In principle, the only hard requirement to a client device is the availability of a Web-

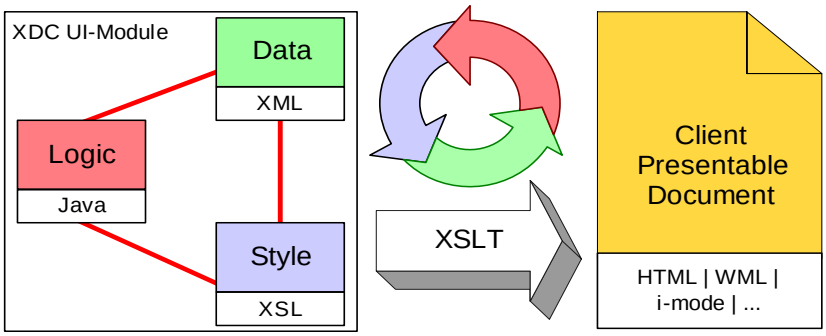

Fig 3: Separation of Concerns

browser. However, the main focus regarding client support in XdC lies in the domain of the "pocketable" devices. For devices of this class, typically some sort of Web browser is available, but these browsers vary widely regarding supported protocols and the corresponding markup languages. Beside software compatibility to existing Web-standards, of course also the physical characteristics like for example the size of the display have to be considered.

To be able to offer smart support for these devices, XdC strictly separates data and the data processing logic from the representation specific to client device. In this connection, representation not only means the look of a document, it also includes the underlying markup language which may be some subset of HTML, WML or similar.

The expected advantage of this approach is that whilst data and logic are retained unchanged, only the representation part has to be exchanged to support various device types. To achieve this functionality, the actual implementation of XdC heavily depends on XML based technology. Thereby, the eXtensible Markup Language serves the purpose of structuring the data independently to its eventual representation. The translation of the XML coded data to client presentable documents is done by using *XSL style sheets* (cf. Figure 3) which are based on XSLT, a transformation language for XML documents (http://www.w3.org/TR/xslt [September 20, 2002]).

6 Conclusion

Building personalized Web applications, i.e., applications that are responsive to the individual needs of each user, is a challenging and time consuming task. The goal of the versatile middleware presented is to free developers of context sensitive applications from the great effort of context management. The expected benefits are significantly reduced development times and thus also noticeably reduced development costs. At the time of writing, the design of the framework is fine-tuned on the basis of the experiences gained during the development of a prototypical campus information system. The fact that the conceptual model of the middleware is closely related to models already used in similar projects (cf. Hitz et al. 2002), together with its modular and versatile approach of context management qualifies XdC to be a stable base for rapid service development.

References

Hightower, J. & G. Borriello (2001). Location Systems for Ubiquitous Computing. *IEEE Computer*, 34 (8), 57-66.

Hitz, M., G. Kappel, W. Retschitzegger & W. Schwinger (2002). Ein UML basiertes Framework zur Modellierung ubiquitärer Web-Anwendungen. *Wirtschaftsinformatik 44 (3), 225-235.*

Huber, A.. & J. Huber (2002). UMTS and mobile computing. Artech House, Boston – London.

Poslad, S., H. Laamanen, R. Malaka, A. Nick, P. Buckle & A. Zipf (2001). CRUMPET: Creation of User-friendly Mobile Services Personalized for Tourism. *Proceedings of 3G 2001 - Second International Conference on 3G Mobile Communication Technologies. 26-29 March 2001. London, UK. http://conferences.iee.org.uk/3G2001/.*

Pradhan, S., C. Brignone, J. Cui, A. McReynolds & M. Smith (2001). Websigns – Linking Physical Locations to the Web. IEEE Computer, August 2001, 42-48.

Zipf, A. & R. Malaka (2001). Developing Location Based Services (LBS) for tourism - The service providers view. *ENTER 2001, 8th. International Congress on Tourism and Communications Technologies in Tourism. Montreal, Canada. 24-27 April, 2001*.