Customisation for ubiquitous web applications – a comparison of approaches

Gerti Kappel

Business Informatics Group (BIG), Vienna University of Technology, Vienna, Austria
E-mail: gerti@big.tuwien.ac.at

Birgit Pröll

Institute for Applied Knowledge Processing (FAW), Johannes Kepler University Linz, Linz, Austria
E-mail: bproell@faw.uni-linz.ac.at

Werner Retschitzegger

Department of Information Systems (IFS), Johannes Kepler University Linz, Linz, Austria
E-mail: werner@ifs.uni-linz.ac.at

Wieland Schwinger

Software Competence Center Hagenberg (SCCH), Hagenberg, Austria
E-mail: wieland.schwinger@scch.at

Abstract: Ubiquitous web applications adhering to the anytime/anywhere/anymedia paradigm are required to be customisable meaning the adaptation of their services towards a certain context. Several approaches for customising ubiquitous web applications have been already proposed, each of them having different origins and pursuing different goals for dealing with the unique characteristics of ubiquity. This paper compares some of these proposals, trying to identify their strengths and shortcomings. As a prerequisite, an evaluation framework is suggested which categorises the major characteristics of customisation into different dimensions. On the basis of this framework, customisation approaches are surveyed and compared to each other, pointing the way to next-generation customisation approaches.

Keywords: ubiquity; web applications; customisation; context; adaptation; personalisation.

Reference to this paper should be made as follows: Kappel, G. Pröll, B. Retschitzegger, W. and Schwinger, W. (2003) 'Customisation for ubiquitous web applications – a comparison of approaches', *International Journal of Web Engineering Technology*, Vol. 1, No. 1, pp. 79–111.

Biographical notes: Gerti Kappel is a full Professor of business informatics at the Vienna University of Technology since 2001. Prior to that she was a full Professor of computer science at the Johannes Kepler University Linz. She received the MS and PhD degrees in computer science from the University of Vienna and the Vienna University of Technology in 1984 and 1987, respectively. From 1987 to 1989 she was a visiting researcher at Centre

Universitaire d'Informatique, Geneva, Switzerland. Her current research interests include object-oriented modelling, database/web integration, ubiquitous web technologies, web engineering, and applications to workflow management and electronic commerce. She has been involved in national and international joint projects between university and industry, as well as sponsored by the Federal Ministry for Education, Science and Culture, and the EU. Kappel is a member of ACM, IEEE, GI, and OCG.

Birgit Pröll studied Computer Science at the Johannes Kepler University Linz, Austria. Since 1991 she is employed at the FAW (Institute for applied Knowledge Processing, Head: Prof. Dr. Roland Wagner) at the Johannes Kepler University Linz. She has been engaged in industrial and research projects in the areas of expert systems and CAD, configuration management, relational/object-oriented databases, and information systems and electronic commerce on the World Wide Web. From 1995 to 2000 she managed the development of the web-based tourism information systems TIS@WEB and TIScover at the FAW. Her current research interests and fields of teaching comprise information retrieval, web engineering and electronic commerce.

Werner Retschitzegger studied business informatics at the Johannes Kepler University (JKU) Linz, Austria. He received the MS (1991) and PhD (1996) degrees from the Faculty of Business, Economics and Social Sciences and his Habilitation (Venia Docendi) for applied computer science from the Faculty of Natural Sciences and Engineering of JKU. From 1990 to 1993 he has been working for the Institute for Applied Knowledge Processing in Hagenberg, Austria, being involved in various national and international industrial and research projects. Since 1993, he is affiliated with the Department of Information Systems at JKU. In 2002, he got a temporary full professorship for business informatics at the Vienna University of Technology. He has published more than 60 papers in international refereed journals and conference proceedings. His current research interests comprise object-oriented modelling, the integration of database and web technology and its application to electronic commerce, ubiquitous web applications and web engineering.

Wieland Schwinger is working as a Senior Researcher and Project Manager of strategic research projects at the Software Competence Centre Hagenberg. Prior to that he worked as Teaching and Research Assistant at the department of Information Systems and at the Centre for Computing and Information Technology of the Johannes Kepler University (JKU) Linz, Austria focusing on web applications. He studied computer science at the University of Skövde, Sweden and business informatics at JKU, receiving master degrees from both universities in 1997 and 1998 respectively. For his PhD he specialized in modelling ubiquitous web applications graduating in 2001 from the JKU. His current research interests comprise ubiquitous web applications, conceptual modelling, and mobile computing. He is involved in national and international projects on the development of methods and tools for ubiquitous web applications amongst them in the EU-project "Ubiquitous Web Applications (UWA)" (IST-2000-25131). He is author of more than 20 publications about web application modelling, customisation, and web engineering. He serves as a reviewer for several international journals and conferences and was organizing member of ECOOP96 and co-chair of IIWAS2001, and IIWAS2002.

1 Introduction

Over the next decade, millions of businesses, billions of people, and trillions of devices will be connected [1]. The promise of this envisioned future is access at *anytime*, from *anywhere*, with *anymedia*, to *everyone* and *everything*, thereby boosting productivity and enabling more satisfying ways to get things done. This will radically change the way how people will interact, how business is conducted, and how our every day live will be organized. The internet and the World Wide Web in particular will be the technological bases for that fundamental change.

Looking back in history, the World Wide Web can be characterized by *three generations of applications* differing in both, the *technology* used and the *services* provided [2–4]. At the beginning, the World Wide Web has been employed merely for simple *read-only applications*, presenting information to anonymous users whose number and type is not necessarily predictable. Such applications were realized by some web server offering static web pages for browsing through, only. Soon after, the web was more and more used as a platform for full-fledged, increasingly complex applications, where a huge amount of change-intensive information were (partly) managed by underlying database systems [5]. Web pages are generated out of the database either at the user's request or in advance as soon underlying data changes [6]. This led to the *second generation of web applications*, providing the basis for promising application areas like *e-commerce* [7].

Currently, we are facing the *third generation of web applications* being characterised by the *anytime/anywhere/anymedia paradigm* as mentioned before, thus providing *ubiquitous access* to services, turning e-commerce into *m-commerce* [8]. The traditional 'one-size-fits-all' approach in the development of web applications is not appropriate to serve the fundamental objective of ubiquitous web applications to communicate the *right thing* at the *right time* in the *right way*. The user should be enabled to interact efficiently with the system despite restrictions in the physical environment, thus preserving *semantic equivalence* of services and to take advantage from knowledge about the situation of use which is necessary to achieve *semantic enhancement* of services.

Ubiquitous computing was first stressed by Marc Weiser [9], envisioning a scenario where computational power would be available everywhere embedded in walls, chairs, clothing and the like. Weiser's goal was to achieve the most effective kind of technology, which is available throughout the physical environment, while making them effectively invisible to the user. In the area of web applications, ubiquity is not seen as visionary in this highly pervasive sense, meaning that computing power is embedded everywhere. Rather, *ubiquitous web applications* build more on existing technology, in that web access is no longer primarily a domain of browsers based on desktop PCs but more and more done by various commercially available mobile devices [10,11]. In particular, ubiquity for the web offers new opportunities and challenges for web applications in terms of *time-aware* [12], *location-aware* [13], *device-aware* [14] and *network-aware* [15] services which can be *personalized* for a certain user or user group, too [16].

The pre-requisite for realising such services is awareness of their *context* [17,18]. One must understand what context is to determine its relevancy and how it can be exploited for adaptation purposes [19]. Knowledge about the device used, e.g., its display resolution, would allow to render the graphical representation of a service accordingly, information about location and time of access together with user preferences would allow to provide more accurate services taking into account the current situation of use. In the

following we use the term *customisation* to denote the adaptation of an application's services towards its context.

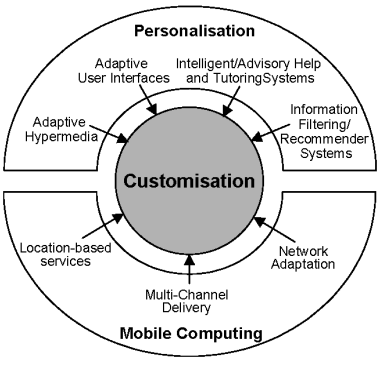
By considering all the different facets of context as mentioned above, customisation is seen more comprehensive than traditional *personalisation*, covering user and usage data only [20,21]. Furthermore, it has to be emphasised that customisation deals with context which can be predicted at design time and may occur at a certain point at run time. Personalisation for example means adapting to different persons, such that they perceive the application differently at the same time. In contrast to that, it is not the primary focus of customisation to accommodate for unanticipated changes over time, caused by, e.g., changing organisational or technological requirements [22,23]. These changes are rather covered by means of system maintenance and re-engineering techniques which are not the focus of this paper [24].

Several approaches for customising web applications have been already proposed, each of them having different origins and pursuing different goals for dealing with the unique characteristics of ubiquitous web applications. This paper compares some representatives of these proposals stemming from different research directions and aims to identify their strengths and shortcomings. According to that overall goal, the remainder of the paper is organised as follows. First of all, Section 2 looks back in history, identifying the roots of customisation within different research directions. Next, Section 3 presents the criteria which are part of the evaluation framework and categorises them along different dimensions. Section 4 gives an overview of 10 customisation approaches and points out their distinguishing characteristics in light of the evaluation framework. Section 5 puts the descriptions of the approaches into perspective by summarising the results and reporting on lessons learned. Section 6 concludes the paper with an outlook to future work.

2 The origins of customisation

Considering the notion of customisation from a historical point of view, one can identify at least two areas of major influence namely personalisation and mobile computing (cf. Figure 1).

Figure 1 Origins of customisation



The notion of *personalisation* represents a major challenge since the end user has been put in the middle of concern when developing interactive applications which dates back to the early 1980s. According to [16], personalisation provides users with an experience most suitable for their background knowledge and objectives. As can be seen in Figure 1, personalisation has been investigated in various different areas of application development. An area dealing with personalisation issues already for a long time is the user interface community, which brought up the notion of *adaptive user interfaces*, cf., e.g., [25]. Adaptive user interfaces aim at tailoring a system's interactive behaviour to skills, tasks and preferences of human users. The broader approach of *intelligent or advisory help and tutoring systems* includes adaptive characteristics as a major source of its intelligent behaviour, cf., e.g., [26,27]. These systems adapt their explanations and teaching strategies to the individual needs of users in terms of their knowledge level and learning progress. Another area dealing with personalisation issues but emphasising more on adapting the content of an application are *information filtering and recommender systems* like found in [28,29]. The goal of these systems is to go through large volumes of dynamically generated textual information and present to the user those which are likely to satisfy his/her information requirements. The emerge of hypertext and hypermedia [30] led to another research direction called *adaptive hypermedia* [31]. Whereas the 'pre-web' generation of adaptive hypermedia systems explored personalised presentation and navigation support (e.g., the need for alternative access paths to information) for a closed group of users [32], nowadays, one faces the formidable task of developing web applications for an unpredictable number of anonymous users while making them work as if they were designed for each individual user.

The area of *mobile computing* can be seen as second major root of customisation. In contrast to personalisation having already a long tradition in application development, the research on mobile computing begun in the early 1990s. At this time, the focus was primarily on *mobility* issues in terms of *location-based services* [33–35]. One of the first projects in this realm reported in literature was the 'Active Badge' project at Olivetti, where information about a user's physical location was sensed at any particular moment to modify the behaviour of programs running on a stationary server [36]. Today, location information which can be made available by mobile network providers or using technologies such as the Global Positioning System (GPS) is used for realising various indoor or outdoor location-based services such as geographically targeted advertising, fleet management, traffic control or emergency services.

Another important research direction in the area of mobile computing is dedicated to *multi-channel delivery*, meaning that services provided by web applications are made available on various mobile devices [37]. This requires to consider the varying capabilities of devices in terms of hardware (e.g., display size and resolution, method of input, memory, disk capacity and computational power) and software (e.g., operating system and web browser) in order to allow a proper adaptation of the application's user interface and interaction behaviour. There are already standardisation efforts in this direction, most notably the device independent activity of the W3C [38–40]. Since mobile devices also imply a wireless connection carrying certain network constraints, *network adaptation* is another influencing research direction for customisation [15]. In this respect, communication autonomy requires that the application properly adapts to sudden disconnections which can be caused either voluntary (e.g., the users wants to avoid disturbance, to reduce cost or power consumption) or happen against the will of the user because the battery becomes empty or network connection is lost. Restrictions and

variations in bandwidth require to change either the content of the transmitted data (e.g., by lossy or lossless compression mechanisms) or the methods used to send that data (e.g., by altering the underlying protocol) [15].

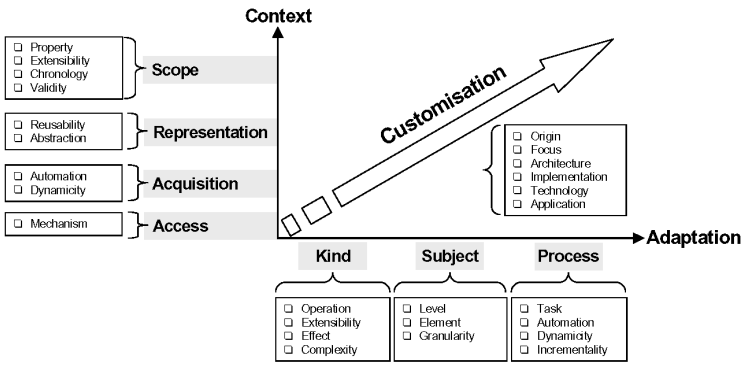
As can be seen from this historical overview, customisation issues have a long tradition in computer science and have been dealt with in a large number of different areas. Existing literature on the state of the art concentrate either on personalisation in certain application domains such as hypermedia systems (cf. [31,16]) or emphasise more on customisation in mobile computing (cf., e.g., [15,41]). In addition, some of these surveys are done without using a proper underlying evaluation framework. This paper, in contrast, tries to cover the whole spectrum of customisation as required by ubiquitous web applications learning from the different research directions discussed above. It proposes a uniform evaluation framework which is suitable for comparing approaches originating from both, personalisation and mobile computing.

3 Evaluation framework

In the following, we want to elaborate on what is necessary when realising ubiquitous web applications by means of customisation, proposing a comprehensive and uniform evaluation framework. The benefit of this evaluation framework is threefold. First, it allows a structured, uniform view to better understand the various aspects of customisation. Second, it can be used as a conceptual framework for evaluating existing approaches on customisation (cf. Sections 4 and 5). Third, it may be employed for developing next generation customisation approaches (cf. Section 6).

Basically, our evaluation framework comprises two orthogonal dimensions *context* and *adaptation* and the mapping in between represented by the notion of *customisation* (cf. Figure 2).

Figure 2 Evaluation framework



In the following, the evaluation framework is explained in more detail. The criteria discussed are partly based on our previous work in this area as described, e.g., in [42,43]. It has to be emphasised that some of these criteria serve as requirements which should be fulfilled by customisation approaches to be appropriate for developing ubiquitous web applications, while others are simply a means to characterise existing approaches along a

common structure. The reference in parentheses besides each criteria will be used as pointer to the criteria in the survey in Section 4.

3.1 Context

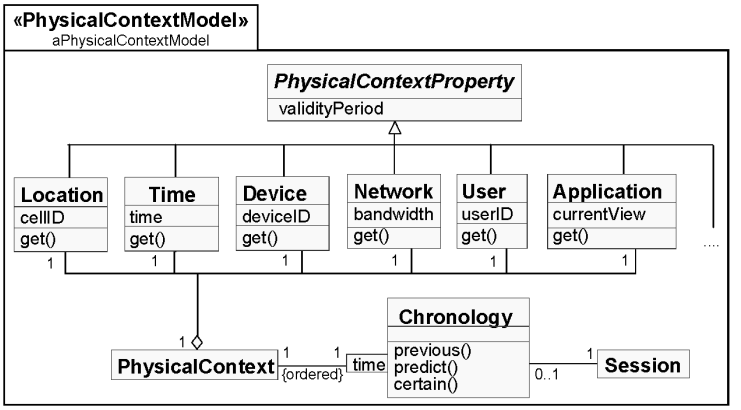
The *context* dimension includes the circumstances of consumption relevant for a ubiquitous web application, mainly dealing with the question 'why to customise and when'. In this respect, we define context as the *reification of certain properties*, describing *the environment of the application* and *some aspects of the application itself*, which are necessary to determine the need for customisation. Context can be further specified by looking at the *scope* of the context which is considered for customisation, its *representation,* its *acquisition* and *access mechanisms* used (for a similar categorisation, cf., e.g., [21]).

3.1.1 Scope of context

The scope of context comprises not only the different context *properties* supported by the system together with the ability to *extend* them, but also the time dimension of context in terms of *chronology* and *validity*.

Property (C.P.). Although the relevant kind of context is dependent on the ubiquitous web application which should be developed, customisation approaches should support a set of built-in context properties. The following context properties (which are illustrated in Figure 3 using UML [44]) are most often found in literature and considered to best support the notion of ubiquity as defined above:

Figure 3 Context properties



- *Location*: Location copes with the need for mobile computing and location-aware services by capturing information about the location from which an application is accessed.
- *Time*: The context property time allows to adapt the application with respect to certain timing constraints such as opening hours of shops or timetables of public transportation.

- *Device*: This context property refers to the demand of ubiquitous web applications for anymedia in terms of multi-channel delivery and provides basic information about the hardware and software capabilities of the device accessing the application.
- *Network*: To allow adaptation on basis of the network it is necessary to provide network context information in terms of, e.g., bandwidth or package losses.
- *User*: Information about the user in terms of, e.g., demographic data, knowledge, skills and capabilities, interests and preferences, goals and plans takes into account the necessity of personalisation. Since the user is regarded being part of the context, we follow an *application-centric perspective* thus monitoring all context properties relative to the application.
- *Application*: In addition to the previously stated context properties, all of them representing information about the environment of the application, it is also required for customisation purposes to get information about the *state of the application* itself, with respect to a certain user [45].

It has to be emphasised that all criteria discussed in the following are applicable to each context property described above.

Extensibility (C.E.). Certain applications may require also additional context properties which are not built-in (e.g., current outside temperature or heartbeat rate). It is not possible to foresee what kind of context properties that might be since the list of context properties is virtually unlimited. Thus, it is required that built-in context properties can be easily extended by additional ones. Note that this criteria deals with system maintenance which is, as disclaimed previously, not the primary focus of this paper.

Chronology (C.C.). Practice has shown that it is useful to broaden the view on context by considering not only the *current context* at a given point in time but also *historical information* (cf. Figure 3). This is necessary to be able to identify changes in context over time. For example since the bandwidth might be constantly changing, it is more important to be able to trace the average bandwidth instead of just having information about the latest one. In contrast, allowing to adapt towards a restricted display size requires information of the current device only. Besides historical context, it might be useful to anticipate possible *future states* of a context, too. For example, concerning video streaming, it is not only relevant how the bandwidth changed in the past, but also how the bandwidth will develop or how stable it can be considered in the future, to be able to tune the resolution of the video accordingly, thus guaranteeing a constant video stream. Finally, since web applications enforce the notion of *sessions*, possibly consisting of a sequence of transactions, context needs to be considered within the boundaries of sessions, i.e., each session has its own context (cf. Figure 3).

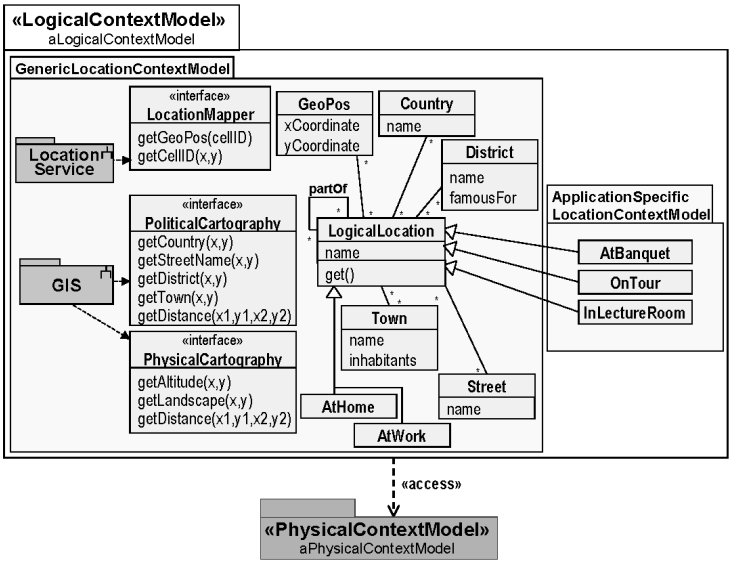
Validity (C.V.). Context properties may not be valid during the complete period until they are updated on basis of the environment. In a mobile scenario an example would be, that, if location information does not change within a certain period, the device might not be online any longer, thus, the location information might no longer be valid. Another example would be the validity of opening hours, which for example, may change during the seasons. Thus, it may be required to specify the period during which a context is valid (cf. Figure 3).

3.1.2 Representation of context

The representation of context comprises two important issues, namely mechanisms for enhancing the *reusability* of the context representation and the level of abstraction at which context is represented.

Reusability (C.R.). A customisation approach should allow to *explicitly represent context* within the system, not just to intermingle context with adaptation or the application itself. An explicit representation would allow for reusability of already defined context across several applications. Another requirement for achieving reusability is that context provided by the customisation approach should be *generic*, that is application-independent. Application independent context may even be reused from external sources, e.g., third-party providers. As illustrated in Figure 4, an example for such an external source would be a Location Service converting a physical cell ID to an *XY*-position in space and vice versa or a Geographic Information System (GIS) providing among others geographical data about cities or streets for a given geographic latitude and longitude.

Figure 4 Reusability and abstraction of context



Abstraction (C.Ab.). According to the level of abstraction where context properties are represented it should be distinguished between *physical context* and *logical context*. Whereas physical context is at a very low level of abstraction which can be directly sensed from the environment (e.g., location in terms of a mobile phone's cell ID), logical context would enable to enrich its semantics thus making it meaningful for customisation purposes (e.g., a street name). Logical context can be provided in terms of profiles (e.g., describing certain hardware characteristics) which is most often the case, but can also be built from existing physical or other logical context by applying different kinds of abstraction or inference mechanisms, (e.g., certain geographical positions relate to the same location in terms of a town). Figure 4 shows an example of a logical context model for the physical context property location. It provides not only a variety of generic

information about the political and physical cartography using both, external sources as described above and abstraction mechanisms in terms of sub-classing, but contains also application specific context information.

Although highly desirable for many applications, the existence of logical context is optional. Note that in the following, we use the term context for depicting both, physical and logical context, not least since all criteria discussed in this section are applicable to both, physical and logical context.

3.1.3 Acquisition of context

The acquisition of context can be characterised by the degree of *automation* considering who is responsible for acquiring the context and the degree of *dynamicity* in terms of when the context is acquired.

Automation (C.Au.). Concerning the acquisition of context, first it has to be defined *who* is in charge for gathering appropriate context information, be it either a human (*manual acquisition*) or the system (*automatic acquisition*) or a combination thereof (*semi-automatic acquisition*). For ubiquitous web applications it is desirable to automatically gather as much context information as possible to reduce user interaction. Physical context properties may be sensed directly from the environment, logical context may be automatically computed on the basis of other context information available. Bandwidth available in the future is an example of context information constructed automatically on bases of the history of bandwidth. Building user categories on the bases of interaction patterns would be another example. Semi-automatic means that automatic acquisition of a physical or logical context is accompanied with information entered manually by, e.g., a user, a designer or the vendor of a device. Although, physical context properties are per definition automatically gathered (cf. above), automatic acquisition is not always possible (cf., e.g., a device not capable of providing location information or a user which can be identified using a login procedure only). Logical context, especially profile information has to be most often entered manually or semi-automatically or even if it could be computed automatically, some necessary information could be missing (cf., e.g., usage data is not available for a user accessing the application for the very first time). The goal should be, of course, to design the application to be robust enough to deal with missing context either by requesting the user to provide the context manually or by providing a default context.

Dynamicity (C.D.). Another important aspect is *when* context acquisition takes place. Considering the frequency of context changes, context can be either *static*, i.e., determined once at application start up (e.g., the device used to select the appropriate interaction style), without considering any further changes or *dynamic*, i.e., determined on every change during runtime (e.g., the bandwidth to adapt the resolution of an image on the fly). In principle, to cope with the dynamic nature of ubiquitous web applications, dynamic context acquisition is required. Although dynamic subsumes the static case, its realisation has completely different implications for the application regarding, e.g., performance or data access. Thus, an additional explicit support of static context acquisition could be desirable.

3.1.4 Access to context

Mechanism (C.M.). Context has not only to be acquired and represented in a proper way, but there must be also appropriate mechanisms in order to make context accessible to the adaptation component of customisation as described below. In this respect one can distinguish between *pull-based* and *push-based approaches*. Whereas the former requires to poll context information as soon as it is required, the latter provides the current context information (to the interested 'clients' in case of a subscription mechanism) as soon as a context change occurred. It has to be emphasised that, for flexibility reasons, a combination of both would be most desirable.

3.2 Adaptation

The second dimension of our evaluation framework is covered by the notion of adaptation, characterised by the *kind of adaptation*, i.e., what changes have to be done, the *subject of adaptation* in terms of what to change and the *process of adaptation* characterising how adaptation is performed.

3.2.1 Kind of adaptation

The kind of adaptation subsumes not only built-in adaptation *operations* and possible *extension* mechanisms but also reasons about their *effect* and *complexity*.

Operation (A.O.). Customisation approaches should support a library of built-in adaptation operations which are appropriate for the built-in context properties (e.g., filter some content, add links, change resolution of an image). For a taxonomy of adaptation operations in the area of personalisation for hypermedia systems it is referred to [32] and [21].

Extensibility (A.Ex.). Similar to built-in context properties, it is required that built-in adaptation operations can be extended by user-defined adaptation operations to be able to offer all necessary adaptations.

Effect (A.Ef.). Regarding the effect of built-in adaptation operations, three categories should be supported. Built-in operations should be able to add certain parts to a web application not present before (e.g., showing a personalised advertisement or executing some additional service), removing certain parts from a web application (e.g., removing all images) or perform some *transformations* (e.g., changing the modality by showing textual descriptions instead of video clips). Considering these effects, it is interesting to reason about the *semantic value*, which should and can be provided by a customised service. The semantic value describes the quality of the output of the customisation for a user relative to a non-customised version of the service. Analogously to the effect of adaptation it may *enhance the semantic value* of a service, *reduce the semantic value*, or it may *preserve the semantic value*, thus achieving *semantic equivalence*. Especially personalisation and location-aware services endow the application with semantic enhancement, in that each particular user is provided with specific added value. On the other hand, the same application customised for the same user may (and certainly does) look different when it is run on different devices and/or in different situations. This is inevitable (for example it is impossible to show that beautiful applet on a PDA with no virtual machine installed), but the service (or the added value) provided to the user should nevertheless be the same. In this case customisation enables to maintain semantic

equivalence, which means that, despite the different context, the value provided to the user should still be the same. The semantic value may be judged on either *objective* or *subjective* bases. Some adaptations may result in an objectively semantically equivalent version (e.g. translating a text from English into Sanskrit) but may be perceived by the user as semantically not equivalent (supposing, e.g., the user is not able to understand Sanskrit). An example where the information is reduced (thus objectively not equivalent) but perceived as semantically equivalent by the user, is the transformation of a digital audio into mp3 format. Though the latter includes only a subset of the information of the digital audio version, the loss is (nearly) not perceivable by the user. Because of this ambiguity, we do not judge the semantic value of an operation but rather the obvious effect of the operation itself.

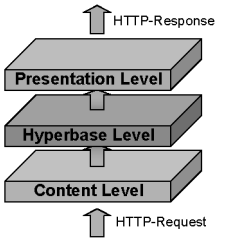
Complexity (A.C.). It should not only be possible to define a *single adaptation operation* which should be performed due to a certain context, but also to define *multiple adaptation operations* performed on the same or on different subjects (cf. below). A transformation of the presentation of a movie from a video sequence into an alternative series of scene pictures is an example of a complex adaptation since it consists of disabling the video and enabling the presentation of the scene pictures [46]. In case that the adaptation is complex, a proper *precedence mechanism* has to be provided to ensure the consistency of the adaptation's effect.

3.2.2 Subject of adaptation

The subject of adaptation can be characterised by looking at the level of the web application which is effected by the adaptation as well as at the concrete *elements* and by distinguishing the number of effected elements denoted by *granularity*.

Level (A.L.). First of all, each level of a web application should be allowed to be subject of adaptation [32,42,47]. These levels, which are illustrated in Figure 5, comprise *content level* (i.e. domain-dependent data), *hyperbase level* (i.e., the navigation structure), and *presentation level* (i.e., the layout of each page together with user interaction facilities). Content adaptation changes the information that is presented to the user by adding or removing content (e.g., a filtering operation), hyperbase adaptation changes the navigation structure (e.g., disabling a link), and presentation adaptation changes the way information is presented to the user preserving semantic equivalence using a transform operation (e.g., changing the modality). Unfortunately it is not always possible to unambiguously categorise each adaptation operation according to these levels. For example concerning compression operations it depends on the compression ratio if the operation works at the content level reducing the semantic value or if it is part of the presentation level preserving semantic equivalence.

Figure 5 Web application levels



Furthermore, a certain adaptation operation may be local to one level only. Changing the display colour of headlines in a page is an adaptation solely performed at the presentation level, none of the other levels need to reflect that adaptation. In other cases it will be required that adaptations done at a certain level are propagated to the other ones. For example, should the content of the web application be reduced from an image rich presentation to a simpler text based version, not only the content needs to be adapted but likewise, the hyperbase as well as the presentation needs to reflect the changes, thus being adapted accordingly. Adaptation should not only effect the web application but it should also be possible to *adapt the customisation itself*, either by updating the context (e.g., actualising the logical user context based on usage data) or by changing pre-defined adaptations (e.g., enabling and disabling a certain adaptation).

Element (A.El.). Each of the levels mentioned above comprises a number of different application elements like *pages*, *links*, *access structures*, *input fields*, *lists*, and *media types*. For each of these application elements, it should be possible to apply different adaptations. For example, video data may be adapted by means of compression methods reducing the resolution or dropping frames whereas access structures may be adapted from a guided tour navigation style to an index navigation style.

Granularity (A.G.). Another important issue concerns the *granularity of adaptation*, indicating the number of application elements effected by a certain adaptation. The granularity which should be supported ranges from *micro adaptation to macro adaptation*. Whereas micro adaptation is concerned with fine-grained adaptations by effecting a single application element only (e.g., disabling a certain link on a certain page), macro adaptation means that rather large parts of an application are adapted, thus effecting multiple application elements (e.g., changing the language effects every textual application element visible to the user). Note that, there is no exact border between micro and macro adaptation. In its most extreme form, macro adaptation simply means that depending on the context, the whole application realising a certain service is substituted by another one, thus better fitting the context.

3.2.3 Process of adaptation

Task (A.T.). The process of adaptation comprises a number of tasks which should be separated in order to allow a fine-grained control about their automation and dynamicity (cf. below). These tasks comprise *initiation*, *proposal*, *selection*, *production, presentation* and *reversion* of the adaptation (cf. [21] for a similar categorisation).

Automation (A.A.). It has to be considered, similar to context, *who* is responsible for performing the tasks necessary for processing the adaptation. For a certain adaptation, all these tasks may be applied either fully *automatically*, so that the human cannot take influence on the adaptation, *manually*, i.e., the user is responsible for the tasks or *semi-automatically* meaning that the user controls if one (or more) of the tasks is (are) performed automatically by the system or not. According to [48], systems supporting fully automatic adaptations are called *adaptive* in contrast to *adaptable* systems, offering more or less control possibilities for the user in terms of semi-automatic adaptation. It would be, for example, reasonable that the system initiates adaptation by proposing an additional index page, lets the user decide which links should be included and finally performs the adaptation. As mentioned in [21], which form of automation to choose has to be carefully weighed for each adaptation, taking convenience of the user, demands of

the user, irritation of the user and the consequences of false adaptation into account. The replacement of images because of device restrictions for example, could be done automatically, whereas the decision whether response time or resolution quality is of higher importance should be made by the user. It is crucial in any case that the user is able to understand the consequences of adaptations which could be facilitated by means of proper tool support [49]. Different to automation of context, where manual acquisition represents a reasonable alternative, manual adaptation is not feasible relating more to re-engineering than to customisation.

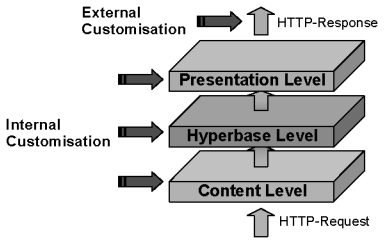
Dynamicity (A.D.). Likewise the context, it is important to consider at which point in time the adaptation tasks are performed which may not be the same. In general, it can be distinguished between *static* and *dynamic* adaptation, meaning the tasks are performed either at design time or at runtime [50]. Concerning dynamic adaptation, three more options can be distinguished. First, the tasks can be done as soon as a context changes (immediate dynamic adaptation). Second, adaptation can be done not before the user requests the page which is subject to adaptation (deferred dynamic adaptation) [51]. Third, adaptation can be done periodically. An example for static adaptation is to predefine two versions of an image, one with a high resolution intended to be presented on desktop computers and a low-resolution black-and-white one intended for handheld devices. The actual version of the image which is presented is then just chosen at runtime either automatically or not. In contrary an image might be dynamically adapted relative to the bandwidth currently available since the different network bandwidth situations can hardly be pre-assumed. Similar to context, the dynamic nature of ubiquitous web applications demands for dynamic adaptation allowing to parameterise the adaptation operation appropriately, although static adaptation production may be useful for some cases, especially due to performance reasons, degrading the adaptation operation to a simple ‘select’. Dynamic presentation of adaptations on a change of context is used in order to realise push-based adaptive services (e.g., pushing a special discount advertisement to the user’s mobile device, if it is lunch time, and the user is nearby a fast food restaurant which matches the user’s preferences). It has to be emphasised that the criteria of automation and dynamicity are orthogonal to each other, meaning that each combination would be reasonable in some sense with respect to the tasks described above.

Incrementality (A.I.). Adaptation may be done from *scratch*, i.e., each time adaptation is required, it is done based on the original (non-adapted) version of the service [38]. For example, transforming a video to audio and to a series of picture scenes need to be conducted from the original video. In contrary, adaptation may also be performed *incrementally*, meaning that adapted versions are made persistent allowing that subsequent adaptations are conducted on bases of the results of previous adaptations. Thus, incremental adaptation could sometimes significantly reduce processing resources. Despite this benefit incremental adaptation might not be applicable in all cases, e.g., if lossy transformation are applied which cannot be reversed in later adaptation steps. Consequently, both forms of adaptation should be supported by a system. Furthermore, it has to be emphasised that incremental adaptation can also be seen as a form of adaptation to change since in this way the system may continuously evolve over time. Although this is not our focus we included this criteria because of the benefit described.

3.3 Overall characteristics of customisation

Concerning customisation itself, the evaluation framework contains some general criteria comprising the *origin* of the approach according to the distinction made in Section 2, its major *focus*, its basic *architecture*, whether it is *implemented* or not, the *technology* used and exemplary *applications*. Concerning the architecture, the crucial issue is if the ubiquitous web application is aware of customisation in terms of knowing about *context* and/or *adaptation* (referred to as *internal customisation*) or not (referred to as *external customisation*) [15] (cf. Figure 6).

Figure 6 Alternative architectures



The benefit of internal customisation is that adaptations can be very powerful, effecting every application element, since the application is designed for adaptation. External customisation in contrast means that the application responds to a request and delivers the result regardless of any customisation. Customisation is rather done in a successive step on basis of the result, delivered by the application. In this case the application is *completely unaware of customisation*, allowing to customise even already existing applications. The potential of customisation, however, is rather limited. External customisation can be realised using a proxy-based architecture where the proxy is responsible for customisation. Because of the benefits and drawbacks of each alternative, a customisation approach should support both options.

4 Comparison of customisation approaches

There are numerous customisation approaches reported in literature (for an overview, cf., e.g., [15,21,31,41]). This section describes 10 of these approaches, using the evaluation framework described in the previous section. The rationale behind choosing these 10 was to assort a representative mix of approaches having different origins, thus supporting different concepts and having different application areas. Another intent was to evaluate not only research approaches but also commercially available systems. Additionally to recently proposed approaches, we included also older ones if their importance and influence on others suggested so.

Each of the selected approaches is described in the following within a separate subsection according to the criteria of the evaluation framework, thus giving an overall understanding of each approach before discussing general findings of the evaluation in Section 5. For readability reasons, the criteria are not always described in the same order. To ensure, however, that every criteria can be traced in the description, the criteria are referenced by means of their abbreviations given in Section 3. The results of the

evaluation are also summarised within three tables, one for describing general characteristics of the customisation approach (cf. Table 1), another one dealing with context issues (cf. Table 2) and finally a table regarding adaptation (cf. Table 3). The criteria are rated using the following symbols:

☑… means that there is evidence that the approach provides explicit concepts to support the criteria, or although not explicitly mentioned, it can be obviously inferred that the criteria is supported

☐… means that the approach does not explicitly support a certain criteria

~… means that the criteria cannot be applied meaningfully

In case that the concepts provided by an approach differ from their implementation, the differences are mentioned within the description of the approach, ratings within the tables are done on bases of the concepts.

Table 1 Overall comparison of approaches

		Origin	*Major focus*	*Architecture*	*Implementation*	*Technology*	*Application*
Approach	Cheverst et al.	location-based services	context-aware generation of tourist information	internal	implemented	HTML meta-tags	location-aware tour guide
	De Bra et al.	adaptive hypermedia	simple user modelling and adaptation tool	internal	implemented	XML, condition/ action rules	e-learning
	Dey et al.	location-based services	context representation and processing	internal	implemented	Java, XML	location-aware - attendance list - display - mailing-list
	Fink et al.	adaptive user interfaces	improve the overall access to a web application considering persons with special needs	internal	implemented	knowledge representation language, HTML meta-tags	personalised information about a metropolitan area
	Fox et al.	network adaptation	on-demand datatype-specific compresson of content	external	implemented	-	image distiller, video stream distiller
	IBM TP	multi-channel delivery	adaptation of existing services towards different client capabilities	internal/ external	implemented	XPath, XSLT	information portals and ebusiness sites
	Nagao et al.	multi-channel adaptation	semantic annotation for supporting versatile and intelligent web content	external	implemented	IBM transcoding publisher, XML, SMIL	applied to some existing web sites
	Oracle Wireless	multi-channel delivery	adaptation of existing services towards different client capabilites	external	implemented	XML, XSLT, Java, DBS	various
	Rossi et al.	adaptive hypermedia	adaptation modelling	internal	implemented	condition/action rules, oo-patterns, UML	conference paper review system, e-commerce
	Schmidt et al.	mobile computing	context representation and processing	internal	implemented	-	light-sensitive display, context-aware access through WAP devices

4.1 The GUIDE system of Cheverst et al.

The GUIDE system of Cheverst *et al.* [52–55] stems from the area of location-based services. The focus is to provide tourists with up-to-date and context-aware information about a city via a PDA. Although the focus of the system is in providing location-based services, a more comprehensive logical context model in terms of profiles is provided (C.Ab.), distinguishing between so-called *personal context* in terms of information about the user (e.g., preferences, current location and a history of already visited attractions

(C.C.)) and so-called *environmental context*, comprising information about attractions (e.g., links between nearby attractions, opening and closing times, relevance to user interests) (C.P.). Thus, logical context is mainly specific to the tourism domain (C.R.). There are no mechanisms to extend the pre-defined context properties (C.E.) nor inference mechanisms to automatically derive higher-level logical context (C.Ab.) or a validity period (C.V.). The current physical location context is gathered automatically (C.Au.) at runtime (C.D.), although the user is able to manually enter the current location in case that cell coverage is temporarily left. Logical context is, in principle, entered manually, certain information about the user, e.g., interests is acquired semi-automatically based on the interaction history (C.Au.). Context information is accessed in a pull-based manner (C.M.).

Table 2 Comparison of context characteristics

		Scope of Context									*Representation of Context*		*Acquisition of Context*					*Access to Context*	
		Property						**Extensibility**	**Chronology**		**Validity**	**Reusability**	**Abstraction**	**Automation**			**Dynamicity**		**Mechanism**	
		location	time	device	network	user	application		history	future				manual	semi-automatic	automatic	static	dynamic	push	pull
Approach	Cheverst et al.	✓	✓			✓			✓				✓	✓	✓			✓		✓
	De Bra et al.					✓		✓				✓	✓		✓			✓		✓
	Dey et al.	✓	✓			✓	✓	✓	✓			✓	✓			✓		✓	✓	✓
	Fink et al.					✓			✓			✓	✓		✓		✓	✓		✓
	Fox et al.			✓	✓									✓		✓		✓		✓
	IBM TP	✓		✓	✓	✓		✓				✓	✓	✓				✓		✓
	Nagao et al.					✓	✓					✓	✓	✓	✓		✓	✓		✓
	Oracle Wireless	✓		✓		✓						✓	✓			✓		✓		✓
	Rossi et al.					✓							✓			✓	✓	✓		✓
	Schmidt et al.	✓	✓	✓	✓	✓		✓			✓	✓	✓			✓		✓	✓	

Legend:	✓	... explicitly supported
		... not explicitly supported
	~	... not applicable

Taking a look at the adaptation features of GUIDE, it is distinguished between *coarse-grained adaptation*, e.g., changing the language of the descriptions and *fine-grained adaptations*, e.g., presenting information about the current context or filtering/sorting information depending on a certain context (A.O., A.Ef.). Thus a certain adaptation cannot only effect a single but rather numerous web pages as is the case when changing the language or when generating a complete guided tour (A.G.). In particular, the subject of adaptation comprises all three levels (A.L.), focusing on text and link adaptations without changing the modality (A.El.). Adaptations can be complex, e.g., location-based filtering can be followed by a sorting operation before presenting the adapted web page to the user (A.C.). Adhering to an integrated architecture, adaptations are realised by web pages intermingling with proprietary HTML meta tags which allow to query the context (e.g., determining the user's interest in that particular attraction which has a certain historical value associated) and to perform the appropriate adaptation (e.g., insert a user's

location or insert nearby attractions). Extensibility of this pre-defined tag set is not foreseen (A.Ex.). In principle, there is no separation between the different tasks of the adaptation process (A.T.), adaptation is done fully automatic (A.A.). The tags are interpreted on the fly, thereby realising dynamic adaptation (A.D.) as soon as the user accesses a context-aware web page. Concerning dynamic adaptation however, there is a separation of tasks with respect to the computation of nearby attractions. This production task is done automatically, immediately after the location context has changed, whereas the presentation itself is done upon a user's request. Finally, incremental adaptation is not supported (A.I.).

Table 3 Comparison of adaptation characteristics

		Kind of Adaptation																		*Subject of Adaptation*														*Process of Adaptation*						
		Operation	Extensibility	Effect			Complexity		Level			Element							Granularity		Tasks	Automation			Dynamicity		Incrementality
				add	remove	transform	simple	complex	content	hyperbase	presentation	text	audio	image	video	link	others	micro	macro		automatic	semi-automatic	manual	static	dynamic	
Approach	Cheverst et al.	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓				✓		✓	✓		✓				✓	
	De Bra et al.	✓		✓	✓		✓	✓	✓	✓		✓				✓		✓			✓				✓	
	Dey et al.	✓	✓	✓		✓	✓		✓			✓	✓				✓	✓			✓			✓	✓	
	Fink et al.	✓		✓	✓	✓	✓		✓	✓	✓	✓		✓		✓		✓		✓		✓		✓		
	Fox et al.	✓			✓	✓	✓				✓	✓		✓	✓			✓		✓		✓			✓	
	IBM TP	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓		✓				✓	
	Nagao et al.	✓			✓	✓	✓	✓	✓		✓	✓	✓	✓	✓			✓		✓	✓	✓	✓		✓	
	Oracle Wireless	✓	✓			✓	✓	✓			✓	✓	✓	✓				✓	✓		✓				✓	
	Rossi et al.	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓				✓	✓	✓	✓		✓			✓	✓	
	Schmidt et al.	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~	✓				✓	

Legend: ✓ ... explicitly supported; ... not explicitly supported; ~ ... not applicable

4.2 The AHA! System of De Bra et al.

The system 'AHA!' proposed by De Bra *et al.* [56] origins from the area of adaptive hypermedia systems and represents a simple user modelling and adaptation tool which is realised external to a web application. In particular, the approach is used for customising online courses in the area of e-learning.

AHA! dynamically (C.D.) considers user as the only explicitly supported physical context property (C.P.), although it is mentioned that other properties could be supported as well (C.E.). Logical context information about the user is represented in terms of XML files in a generic way (C.R.) and includes a knowledge level and an interest level which can be expressed by Boolean, Integer or String values (C.Ab.). These levels can be defined for each part of the web pages which should be adapted. Logical context can be automatically updated by means of adaptation operations (cf. below) (C.Au.), as soon as

the user accesses a web page (C.M.). Neither the chronology of context (C.C.) is considered nor a validity period (C.V.).

The adaptation model which is again represented by means of XML incorporates among others requirements and condition/action rules for arbitrary parts of the web pages which should be adapted. The requirements are depicted by Boolean expressions, indicating the desirability of links or fragments of pages for a user. Depending on the evaluation results of these expressions links are enabled or disabled and fragments are included or excluded (A.O.). These are the only adaptation operations provided by the system, user-defined ones are not supported (A.Ex.). Consequently, subject of adaptation are textual elements at the content level and links at the hyperbase level (A.L.), (A.El.). The effects of these atomic adaptations (A.C.) comprise reduction and enhancement (A.Ef.) of a certain web page (A.G.). Condition/action rules are used in order to perform (possibly cascading) updates to the user model by, e.g., increasing/decreasing the knowledge level (A.C.). Adaptation is done statically since there is no parameterisation (A.D.) and automatically (A.A.) thus providing no separation between the different tasks of adaptation. Adaptation is done non-incrementally (A.I.) and initiated as soon as a page or parts thereof are accessed or the user model is updated.

4.3 The Context Toolkit of Dey et al.

The approach of Dey *et al.* [19,41,57,58] origins from the area of location-aware services. Its main focus is to provide a comprehensive conceptual framework together with a toolkit for representing and processing context information independent of an application, thus establishing an external customisation architecture. The context toolkit which realises only a subset of the frameworks' concepts has been implemented using Java and XML [59]. Various applications have been developed on basis of the toolkit, including, e.g., an in/out board for indicating those persons which are inside a building, a personalised information display which shows the user in front relevant information and a context-aware mailing list sending e-mails only to those subscribed users which are currently in a certain building.

The context properties supported by this approach comprise location, time, user and status denoting the application state (C.P.). There is an explicit separation between physical and logical context (C.Ab.). Physical context is represented by so-called *context widgets*, which hide the details of actual context acquisition (e.g., done by means of sensors) and provide a uniform interface to components using context, thus allowing extensibility (C.E.). In addition, the system maintains a history of contexts (C.C.). Concerning logical context, different abstraction mechanisms are provided to represent both, application-dependent and generic context (C.E., C.R.). These mechanisms comprise *interpreters* to produce 'higher-level' context information by combining existing context properties and *aggregators* to gather and store logically related context information available from different widgets at one place. For accessing context, polling and subscription mechanisms are supported (C.M.). Context is determined dynamically (C.D.) and acquired automatically, focusing on automatic inference of logical context information (C.Au.). Context availability is explicitly mentioned but not further dealt with, a validity period is not supported at all (C.V.).

Considering adaptation, pre-defined *context services* are introduced which are responsible to execute actions on behalf of the application (A.O.). Context services can

be executed synchronously or asynchronously and divided into three categories comprising *recommendations* (i.e., display of context information or proposal of appropriate actions to the user), *triggers* (i.e., automatic execution of services) and *taggers* (i.e., attachment of context information to objects for later retrieval). Currently, the context toolkit provides only a couple of generic context services comprising the display of choices and messages, conversion of text to speech, a beeper and an audio switch. It is, however, possible to implement application-specific ones in terms of Java programs (A.Ex.). The pre-defined context services (both, the suggested ones and the implemented ones) are simple (A.C.), adding or transforming (A.Ef.) single text and audio elements as well as services (A.El., A.G.) at the content level and the presentation level (A.L.). Adaptation tasks are not separated (A.T.), the whole adaptation process is done fully automatically (A.A.) and non-incrementally (A.I.). In principle, adaptation is done dynamically, except concerning the audio switch, where the adapted versions are already pre-defined before runtime (A.D.). Finally, it is interesting to mention that the architecture of the system is reflective, in that the whole customisation capabilities of the framework (i.e., widgets, interpreters, aggregators and services available) are maintained within a registry called *discoverer*, thus providing a lookup mechanism for applications using the context toolkit.

4.4 The AVANTI project of Fink et al.

Fink *et al.* [60] present the AVANTI project, coming from the area of adaptive user interfaces. The goal of this project is to improve the overall access to a web application, particularly considering the needs of persons with special needs. The system is employed to provide hypermedia information about a metropolitan area (e.g., about public services, transportation and buildings) for a variety of users.

Customisation is focused on user context, characteristics of devices and network are mentioned but not further dealt with (C.P.). The issue of extensibility is not considered (C.E.). Logical context information (C.Ab.) about users and user groups in terms of interests, knowledge and abilities is explicitly represented (C.R.) by means of a knowledge representation language similar to KL-ONE [61]. Logical context is acquired semi-automatically (C.Au.). Within an initial phase, it is first of all entered by a human at design time (C.D.), by exploiting the history of a user's interaction with the system (C.C.) it is further on refined at runtime (C.D.) using an inference mechanism called *adaptation rules*. Access to context is performed in a pull-based manner (C.M.). A validity period is not supported (C.V.).

Built-in adaptation operations (A.O.) are non-complex (A.C.) and allow to include, exclude or transform (A.Ef.) parts of a certain web page (A.G.). An example for an optional part would be the supplementary information on wheelchair accessibility, an example for an alternative element would be a general vs. a detailed description or an image vs. its textual description. Since the optional and alternative parts of a web page are already pre-defined at design time, adaptation is done statically (A.D.). Extensibility of adaptation operations is not an issue (A.Ex.). Subject of adaptation are textual elements or images (A.El.) at all three levels of a web application (A.L.). The process of adaptation is performed again by means of adaptation rules, some of its tasks are separated (A.T.), thus allowing semi-automatic adaptation (A.A.). In particular, the

system produces certain adaptations which can be further refined by the user. Finally, adaptation is done non-incrementally (A.I.).

4.5 Fox et al.

The approach of Fox *et al.* [62,63] has its origin in the area of network adaptation. In this respect they propose a proxy-based architecture that enables to perform on-demand datatype-specific compression of data to a variety of client devices and network constraints. The approach has been used for realizing different applications including a modular proxy client, an image distiller, and a video stream distiller.

According to the overall focus the approach supports device and network characteristics including effective processing capabilities, bandwidth, roundtrip latency, and packet error (C.P.). Apart of those no context properties can be considered (C.E.). Context is considered dynamically (C.D.) as soon as the client accesses the web application (C.M.). Three methods of determining the context are discussed: explicitly given context through the user, exploiting network profiles, and automatic monitoring of, e.g., the network context (C.Au.), the context, however, is not explicit represented consequently the criteria of reusability is not supported (C.R.). No inference mechanisms are available to capture more abstract context (C.Ab.).

Dependent on the format of text, images and video streams (A.El.) transformations and reductions (A.Ef.) with as little loss of the semantic value as possible are offered. These non-complex (A.C.) adaptations are pre-defined (A.O.), no mechanism is offered to include new adaptations (A.Ex.). All adaptations aim at preserving the service to the user hence tailoring the presentation level only (A.L.). Since application elements of a single page are customized adaptation takes place at a micro level (A.G.) only. Although the adaptation is initiated automatically the user is able to engage into the adaptation process (A.T.) by explicitly retrieving (A.A.) the original version of an application element if desired thus reversing the adaptation. The adaptation is performed dynamically (A.D.) on bases of the original web application (A.I.) each time a page is requested.

4.6 IBM Websphere Transcoding Publisher

IBM WebSphere [64], is a commercial web application development platform. The *Transcoding Publisher* is part of *IBM WebSphere* supporting multi-channel delivery by adapting services towards different client capabilities based on the 'InfoPyramide' described in [65,66]. Additionally, *IBM WebSphere Personalisation and WebSphere Everyplace Suite* offer a personalisation component and a component enabling to realise location-based services, respectively. For this an architecture realising both internal as well as external customisation is implemented. Various information portals and e-business solutions have been realised on bases of this approach.

The context considered comprises location, device, network, and user (C.P.). The physical context information is dynamically (C.D.) enriched through separated components, called *Request Editors* thus incorporating logical context (C.Ab.) at request (C.M.) (e.g., given the user agent string in the request, browser capability information is added). The logical context information is provided through profiles which are maintained by the developer (C.Au.) through dedicated tools. In this way *Request Editors* represent logical context information in a reusable way (C.R.). Furthermore, application

context can be included by means of page annotation. Through the *Request Editors* additional context information can be included (C.E.) into the system. Explicitly, only the current context of a request is considered (C.C.). Monitoring components are foreseen in the platform framework which would allow the developer to consider also historical information as well as the automatic generation of logical context information but no explicit support is given (C.Au.). No validity constraints are addressed in the approach (C.V.).

A series of so called *Document Editors* provide predefined adaptation operations (A.O.). A text-engine allows to transform the presentation (A.L.) so it fits the constraints of devices (e.g., images are transformed to links). Deck fragmentation allows to trim pages to the demands of devices understanding WML [49] reorganising the hypertextual structure (A.L.) of the application introducing links between smaller fractions of the pages. Text clipping allows to operate on the content (A.L.) of the web application which can not only be reduced but also transformed or even extended (A.Ef.). Furthermore, image transcoding and multi-media transformations are offered. Thus, specific adaptation operations are available to adapt text, images, video and voice data (A.El.) where the effects of the predefined adaptations are partly defined within the logical profiles. Additionally, a stylesheet editor is provided which allows the developer to specify adaptations in terms of XSLT [59]. Furthermore, new adaptations can be introduced (A.Ex.) in terms of servlets. Consequently, the approach offers complex adaptations (A.C.) and allows to adapt the web application both at a micro as well as at a macro level (A.G.). Adaptation is performed automatically (A.A.) without user interaction (A.T.) after the web application has generated the response to the user's request. The adaptations are invoked dynamically (A.D.) on the basis of the original result of the web application (A.I.) according to a given priority.

4.7 The GDA project of Nagao et al.

In the *Global Document Annotation* (*GDA*) project, Nagao *et al.* [67] propose semantic annotation which aims at supporting versatile and intelligent web content particularly focusing on multi-channel delivery. Semantic annotation allows to associate meta-data in terms of XML tags to any web content including text, images, and videos. This meta-data permits to automatically infer the underlying semantic structure of documents. Thus, this approach supports the development of web applications offering transformation, summarisation, and translation aiming at conserving as much semantic value to the user as possible. A personalised summarisation system reflecting the readers' interests and an external transcoding system based on *Web Intermediaries* from *IBM* [64] (an extensible http-proxy-server) have been realised.

The identity of the user (C.P.) is considered dynamically (C.D.) and combined with a user profile providing additional information (C.Ab.). The user profile is represented explicitly (C.R.) and maintained manually (C.Au.), independently of the web application. In case that user preferences are not given, default values are assumed. Additionally, application context comprising the semantic annotations is considered. As language for semantic annotation a set of XML tags is proposed. Semantic annotations are stored separately (C.R.) from the web content and are maintained manually (C.Au.). The following kinds of annotation are distinguished: (i) *linguistic annotation* aiming at making text machine readable, (ii) *commentary annotation* serves for annotating

non-textual content like images, sounds, both kinds of annotations are manually (C.Au.) provided, whereas (iii) *multimedia annotation* describing semi-automatically (C.Au.) the content of digital videos. The user and the application context are the only context properties made available and no new context properties can be included (C.E.). For both, the user context and the application context only the current context is considered (C.C.) as soon as the user accesses the web application (C.M.). The issue of invalid context is not dealt with (C.V.).

Based on this context information the following generic adaptations are considered (A.O.): (i) text transcoding, (ii) image transcoding, (iii) voice transcoding and (iv) video transcoding (A.El.). Those adaptations are performed dynamically (A.D.) based on the original data (A.I.). Text transcoding allows the user to summarise, thus reduce (A.Ef.) the text by three means (A.A.): (i) automatic adaptation based on a learning mechanism relying on a weighted feature vector, (ii) semi-automatic adaptation in terms of automatic summary employing concurrence statistics relative to user specified words of interest and (iii) manually adaptation by which users can specify words and phrases which should be included in the personalized summary. Image transcoding allows to change image size, colour and resolution (A.Ef.). Voice transcoding enables the user to transform (A.Ef.) data to speech by employing a voice synthesis. Video transcoding allows summarisation (A.Ef.) and transformation (A.Ef.) of video to text and video to speech performing a combination of video to text and text to speech transformation (A.C.). It is not foreseen that new adaptations can be introduced (A.Ex.). Individual application elements (A.G.) are adapted when the user requests so (A.T.) thus keeping her in control of the adaptation. Adaptation focuses primarily on the content and presentation level although new links are inserted which allow activation of some adaptations (A.L.).

4.8 Oracle9i Application Server Wireless

Oracle offers a product called 'Oracle9i Application Server Wireless' which is part of the Oracle Application Server [68]. Originating in the area of multi-channel delivery, the focus is to provide a platform which enables not only to develop new web applications independent of any device, thus realising an internal architecture, but also to adapt existing web applications to be used from various devices using a proxy-based architecture, thus supporting external customisation.

Oracle supports location, device and user context (C.P.) at a physical and a logical level in terms of profiles (C.Ab.) and explicitly maintains them within relational tables (C.R.). Depending on the availability of information about physical context properties, their acquisition is done automatically or manually (C.Au.) at runtime (C.D.). Location context can be manually defined by the user (in case that automatic localisation is not possible) using so-called *location marks* which associate logical location information, i.e., an address with coordinates. Device context and user context is identified automatically by examining the request header. In case that the user context is not available, an explicit login procedure is used. Logical user context is supported by allowing to specify very basic user-related data and application-related data (e.g., activated services), logical device context is restricted to some very basic information about devices (e.g., screen-width and -height). It is neither possible to incorporate additional physical or logical context properties into the system nor to change the existing schema for logical context information (C.E.), with the exception of user

context supporting an appropriate API. Chronology of context (C.C.) or a validity period (C.V.) are not supported, context is accessed in a pull-based manner (C.M.).

Adaptation is done by using so-called *transformers* which can be either XSLT-stylesheets [59] or Java programs. For transformations (A.Ef.) there are a number of pre-defined stylesheets for changing the markup language (e.g., WML, Tiny-HTML, VoiceXML) and there are some operations in order to transcode images (e.g., format conversions, rotations) (A.O.). Thus adaptation can be both, simple and complex (A.C.) as well as micro (e.g., a single image) and macro (e.g., all pages of the application) (A.G.), thus effecting text, image and audio elements (A.El.) at the presentation level of a web application (A.L.). For transformers, the subject of adaptation is provided by so-called *adapters*. Adapters are in fact Java programs responsible for gathering data from external sources and converting these data into an intermediate XML format. For this, the so-called *Simple Result DTD* specifies the elements of an abstract user interface consisting of containers, menus, forms and tables. There are a couple of pre-defined adapters for, e.g., HTML pages, XML pages, relational tables and plain text. It has to be emphasised that appropriate APIs allow to implement arbitrary adapters and transformers (A.Ex.). Adapters are associated with transformers by means of so-called '*master services*', which are mainly responsible for propagating user requests appropriately. Since it is possible to parameterise master services, dynamic adaptation is supported which is triggered on request (A.D.) and processed in an atomic step (A.T.) without any user intervention (A.A.). The use of adapters allows to customise existing web applications which are not aware of any customisation also to implement applications which directly generate XML files according to the intermediate XML format. Finally, adaptation is done in a non-incremental way (A.I.).

4.9 The OOHDM approach of Rossi et al.

Rossi *et al.* [69,70] propose a UML-based modelling method for web applications called *Object-Oriented Hypermedia Design Method (OOHDM)* which has been recently extended by personalisation concepts. The basic architecture of this approach adheres to internal customisation and employs object-oriented design patterns [71] and condition/action rules for representing the personalisation concepts. Currently, a visual modelling tool is realized and the personalisation concepts are implemented using Smalltalk. The abilities of OOHDM are demonstrated by modelling a conference paper review system and several examples in the domain of electronic commerce.

Focusing on personalisation, OOHDM supports user context only (C.P.), extensibility is not explicitly considered (C.E.). Chronology of context is captured by a shopping history example which does not allow to reason about time spans of context (C.C). A validity period is not supported (C.V.). Although context is represented explicitly, it is primarily application-specific, thus impeding reusability of context (C.R.). It has to be noted, however, that OOHDM deals with other forms of reusability by providing generic mechanisms to model personalisation and by proposing a specific framework language OOHDM-frame [72] for modelling generic parts of web applications. Concerning abstraction of context, there is a separation between physical and logical context focusing, however, on application-specific user profiles only (C.Ab.). Interestingly, besides context information, the user profile has also knowledge about the personalisation itself. Context acquisition is done automatically (C.Au.), either statically or dynamically

(C.D.) Access to context is performed pull-based not before the personalized application element is accessed (C.M.).

OOHDM provides built-in adaptation operations (A.O.) including filtering, recommendation, and selection thus providing the full range of adaptation effects (A.Ef.). These operations are extensible by application-specific ones using the Strategy pattern (A.Ex.). Complex adaptation operations are supported (A.C.) using the Composite pattern and as precedence mechanism conflict solvers are used. Adaptation can be done at each level of a web application (A.L.) comprising the content level in terms adaptation of the business logic, the hyperbase level comprising nodes and link topologies, and the presentation level including interface objects and interaction styles (A.El.). Macro adaptation is achieved through static adaptation whereas dynamic adaptation (A.D.) enables adaptation at a micro level (A.G.). Adaptation is initiated automatically (A.A.), neither task separation (A.T.) nor incremental adaptation is supported (A.I.).

4.10 Schmidt et al.

Schmidt *et al.* [73–75] propose a layered architecture supporting context recognition as a foundation for mobile computing. Their aim is to make context detection independent of the web application increasing flexibility and robustness. The approach allows to influence all aspects of the web application through internal customisation but does not rely on a specific technology. Prototypical applications are realized providing users with better services utilizing this architecture and overcoming the restrictions of limited input devices.

The proposed architecture comprises four layers. The *sensor layer* is responsible for dynamically gathering information from *physical sensors*, i.e., electronic hardware components that measure physical parameters, and *logical sensors* which gather information from external host services. This also makes possible to include application independent context information and to encapsulate the physical context. The understanding of context is very broad covering beside user, time, location, and device also temperature, touch intensity, accelerations etc. Although not explicitly enumerated but due to the fact of a very broad definition of context comprising a three dimensional space of *environment*, *self* and *activity* also network is part of the considered context (C.P.). Furthermore, the approach is open to include any contextual information which can be detected by a sensor (C.E.). The *cue layer* provide an automatic (C.Au.) abstraction of the physical context (C.Ab.) since one or more cues may dynamically (C.D.) transform values of one sensor. *Cues* are a function which transforms the information reported by the *sensor layer* into a symbolic or sub-symbolic output. The *context layer* represents the current context (C.C.) at an abstract level (C.R.) as a set of two dimensional vectors indicating the situation and the certainty of its detection. This allows also to address the validity of context information (C.V.). Most interestingly also the approximation of context if not available is considered. The context information is actualised independently of the user's access (C.M.).

Three *scripting primitives* are offered at the *application level* to describe when dynamic adaptation (A.D.) in terms of the invocation of an arbitrary function is automatically (A.A.) initiated. The first scripting primitive 'entering a context' allows to specify the activation of an adaptation when the context is identified the first time and the second scripting primitive 'leaving a context' when the context is detected to have

changed. Finally, the scripting primitive 'while in context' allows to repeatedly activate an adaptation as long as the context is valid. Beyond these activation primitives the context information is made available to the application programmer. The realization of the adaptation function is left to the application developer (A.Ex.). Consequently, no statements on the character of the adaptation can be inferred.

5 Summary of results and lessons learned

In this section, we will briefly summarise the results of our comparison by pointing out the major strengths and shortcomings of the approaches surveyed and reporting on lessons learned. For this, the structure of this section follows the major dimensions and criteria of our evaluation framework given in Section 3.

5.1 Context characteristics

Full range of context properties not considered. Interestingly, none of the approaches considers all context properties proposed in Section 3. Depending on their origin, the approaches concentrate either more on personalisation issues or more on mobile computing. There can be, however, some clusters of context properties identified, which are mostly considered together. A number of approaches concurrently take *network and device properties* into account. Those two are often considered together which is reasonable since mobile devices also imply a wireless connection carrying certain network constraints. Besides these technical context properties some approaches consider the physical space in terms of location and time. Personalisation seems to be always an issue, also for those approaches having their origin in the mobile computing area. It obviously has been acknowledged that the social context of a user is also relevant for ubiquitous web applications, thus recognising the long tradition of personalisation. At least simple user profiles are most often supported, tailoring location-based services, multi-channel services or network adapted services to the user's needs.

Potential of combining context properties hardly utilized. Although, as mentioned before, most of the approaches support more than one context property, with respect to a certain adaptation, these context properties are often considered independently from each other (i.e., user for personalisation, location for location-aware services). The context for a certain adaptation is rarely captured as a combination of context properties (e.g., a certain user at a certain location may require other adaptation than at a different location or than another user at this location). Supporting the latter would allow to consider complex real-world situations as a basis for proper adaptation. Only the two approaches focusing on context representation and processing Dey *et al.* [41] and Schmidt *et al.* [75] provide mechanisms for combining context properties in terms of aggregators or cues, respectively.

Chronology is dealt with only in terms of history. Chronology of context which is an important means in order to make adaptations more appropriate, either by considering the context's history or by predicting future values is considered by a few approaches only. In case that a context history is supported, it is mainly used in order to update the user context. Future context is supported by none of the surveyed approaches. This shows that current systems are mainly re-active instead of being pro-active in a sense that future context situations are anticipated. This aspect is dealt with, e.g., in Ref. [76] where

automatic data recharging facilities are realised which pro-actively push content to mobile devices based on anticipated connection failures.

Validity of context is not an issue. Although obsolete information about a context could lead to non-appropriate adaptations of the system (e.g., the list of nearby restaurants is computed on basis of obsolete information about the current location of the user), only one approach (cf. [75]) considers the issue of context validity. In particular, as already mentioned, each context is assigned a value indicating the certainty of the context's occurrence.

Simple context abstraction. Most of the approaches represent logical context in terms of profiles matched to the current context, which is a rather simple form of abstraction mechanism. Inference mechanisms to derive higher-level context for better supporting the adaptation process or the application itself are rarely considered. There are only two exceptions, again Dey *et al.* [41] and Schmidt *et al.* [75], providing various inference mechanisms comprising, e.g., interpretation or computation of standard derivation or quartile distance.

Non-standardised context representation. Recently, also standardisation efforts have been undertaken to collect requirements and provide representation techniques for profiles using XML technologies [59], particularly focusing on device independence and personalisation cf. [38–40]. None of the approaches evaluated in this survey, however, employed these standards although most of them employ XML to represent profiles. They rather rely on proprietary profiles having no common understanding of how the context information is to be interpreted. For example, context information about a screen size is sometimes interpreted either as a minimum screen size required by the application or as a maximum screen size available at the device.

Automatic acquisition of logical context is predominant for user context only. Automatic context acquisition is above all dealt with by approaches in the area of personalisation, in that interaction history is used to update the logical user context in terms of knowledge and interest properties. Other logical context such as device profiles or location profiles is mostly gathered manually. Missing context information is, if at all, dealt with in terms of default values or by requesting user input.

Push-based access to context is not exploited. Most approaches are request-based thus exploiting the context information not before the response is generated. Changes in the context cannot be immediately considered endangering the system to miss some information vital for adaptation. Furthermore, the whole spectrum of possibilities offered by push-technology can not be exploited which would be most relevant for frequently changing context properties.

5.2 Adaptation characteristics

Extensibility of adaptation operations is not commonly recognised. Most of the approaches focus on very special customisation problems (e.g., personalised text summarisation or image compression) consequently offering a limited set of predefined adaptation operations neglecting extensibility. Dey *et al.* [41] and Rossi *et al.* [69] as well as the two commercial approaches surveyed fulfil the requirement of extensibility since they offer a plug-able architecture which allows arbitrary adaptation components to be integrated (e.g., employ user-defined stylesheets or Java programs).

Coarse-grained adaptation underdeveloped. Sometimes certain context situations may require a comprehensive adaptation of the application. Especially to preserve semantic equivalence in case of multi-channel delivery, complex adaptations at a macro level are desirable, allowing to simultaneously customise different parts of a web page. Such coarse-grained adaptations are supported by four approaches only, using either stylesheets in order to adapt the presentation level by, e.g., changing the markup language (cf. [68,64]) or by using object-oriented design patterns to transparently change, e.g., complete navigation structures or the like (cf. [53,69]). It is without a doubt easier to realise only adaptation of single elements of an application rather than allowing for coarse-grained adaptation which requires to handle conflicts between a couple of adaptations as well as to take care of their precedence.

Adaptation of interaction behaviour is not tackled. Adaptation at the presentation level primarily focuses on topics like media transformation or changing the layout and colour style of a page rather than on adapting the interaction behaviour of the application. The reason for this lack could be that web applications are still seen as an information medium thus mainly focusing on content and presentation, not as full-fledged software applications where interaction facilities are a crucial means for achieving quality of access. Mobile scenarios implying restricted input means particularly require the attention of interaction behaviour adaptation. Nevertheless, it has to be emphasised that there are already first steps in this direction. IBM's Transcoding Publisher, e.g., allows to automatically fragment a web application into different decks of a WML application for mobile devices. Rossi *et al.* allow to change interaction behaviour by utilizing object-oriented design patterns.

Adaptation tasks are machine controlled. Although, as already mentioned, automatic adaptation bears the danger of irritating the user or can even lead to false adaptations, most of the approaches adhere to automatism. Only [60,63,67] allow the user to supervise the adaptation either by controlling in which form the adaptation should be done or to revise the adaptation thus requesting the original web page.

Adaptation is done primarily from scratch. None of the approaches support incremental adaptation, rather, each adaptation is done on the original version of the application element. Thus, existing approaches do not take advantage of the performance gains which could be achieved when re-using already adapted application elements.

6 Future work

Based on the evaluation results described in this paper we currently tackle the issue of customisation in ubiquitous web applications from a software engineering point of view [42,47,77–79]. Models of a ubiquitous web application prior to its construction are essential for comprehension in its entirety, for communication among project teams, and to assure architectural soundness and maintainability. There exist, however, only a few methods dedicated to the modelling of traditional web applications neglecting to a great extent ubiquity in terms of customisation.

Therefore, we currently develop a modelling method for ubiquitous web applications focusing on customisation. Customisation is regarded as a *new modelling dimension*, influencing all other tasks of web application modelling including content, hyperbase and presentation design. As a prerequisite for supporting customisation, a set of *generic models* is introduced comprising a *context model* and a *rule model*, together with several

sub models using UML profiles [44] as the basic formalism. Generic means that the models provide, in the sense of an object-oriented framework, not only pre-defined classes and language constructs in order to model customisation but also allow to extend them by means of sub-classing. The context model provides detailed information about the environment of an application and the application itself, thereby triggering the actual customisation as soon as the context changes. Physical context properties possess a history and a validity period, and for each of them, a logical context model exists, containing a generic part and an application-specific part. The rule model employs a rule-based mechanism (cf., e.g., [80]) in terms of *event/condition/action rules* in order to specify the actual customisation. Whereas the event allows for a push-based access to actual context information, the condition allows for a pull-based access by querying the physical and logical context models. For separation of concerns, the application being modelled is divided into a *stable part*, comprising the default, i.e., context-independent structure and behaviour and a *variable*, *context-dependent part*, thus being subject to adaptations. A set of about 80 *generic adaptation operations* is provided for each modelling element at each level of the web application which can be complemented by *application specific* ones. These adaptation operations can be integrated into the ubiquitous web application on the basis of *adaptation hooks*. A *customisation toolkit* in terms of a *customisation rule editor* and *browser* supports an integrated modelling process and facilitates reusability on the basis of a *repository of customisation rules*, *macros* and *patterns.* Finally, a process is introduced, covering the whole task of customisation modelling, with a special focus on reusability, herewith providing a *holistic view* on the development process of ubiquitous web applications.

Acknowledgements

This work was partially funded by *Ubiquitous Web Applications* (UWA) an EU-funded Fifth Framework Programme project (IST-2000-25131) and the strategic research project *Ubiquity Engineering* at the Software Competence Centre Hagenberg (SCCH) which is conducted in the framework of the Kplus Competence Centre Program funded by the Austrian Government, the Province of Upper Austria, and the Chamber of Commerce of Upper Austria.

References and Notes

1 Spohrer, J. and Stein, M. (2000) 'User experience in the pervasive computing age', *IEEE Multimedia*, Vol. 7, No. 1, January–March.

2 Other categorisations of generations of web applications can be found in Ref. [3,4].

3 Powell, T. (1998) *Web Site Engineering*, Prentice Hall.

4 Conallen, J. (1999) 'Modeling web application architectures with UML', *Communications of the ACM* (*CACM*), Vol. 42, No. 10, October.

5 Ehmayer, G., Kappel, G. and Reich, S. (1997) 'Connecting databases to the web – a taxonomy of gateways', *Proc. of the 8th Int. Conference on Database and Expert Systems Applications* (*DEXA 97*), September, France, LNCS 1308, Springer.

6 Pröll, B., Retschitzegger, W., Sighart, H. and Starck, H. (1999) 'Ready for prime time – pre-generation of web-pages in TIScover', *Proc. of the ACM Conf. on Information and Knowledge Management* (*CIKM*), November, Kansas City, Missouri.

7 Kappel, G., Retschitzegger, W. and Schröder, B. (1998) 'Enabling technologies for electronic commerce', *Proc. of the XV. IFIP World Computer Congress*, Vienna/Austria and Budapest/Hungary, August/September.

8 Chakraborty, D. and Chen, H. (2000) 'Service discovery in the future for mobile commerce', *ACM Crossroads*, Winter.

9 Weiser, M. (1991) 'The computer for the 21st century', *Scientific American*, Vol. 265, No. 3, September.

10 Mattern and Sturm [11] uses the term pervasive computing to denote web applications running on different commercially available devices.

11 Mattern, F. and Sturm, P. (2002) 'From distributed systems to ubiquitous computing – the state of the art, trends, and prospects of future networked systems', *Proc. Symposium trends in der Informationstechnologie am Beginn des 21. Jahrhunderts*, May, pp. 109–134.

12 Kleinrock, L. (1996) 'Nomadicity: Anytime, anywhere in a disconnected world', *Mobile Networks and Applications*, Vol. 1, No. 4, January.

13 Großmann, M., Leonhardi, A., Mitschang, B. and Rothermel, K. (2001) 'A world model for location-aware systems', *Informatik*, Vol. 8, No. 5.

14 Rodriguez, J.R. *et al.* (2001) Extending E-business to Pervasive Computing Devices – Using WebSphere Everplace Suite Version 1.1.2, IBM Redbooks, International Technical Support Organisation, SG24-5996-00.

15 Badrinath, B., Fox, A., Kleinrock, L., Popek, G., Reiher, P. and Satyanarayanan, M. (2000) 'A conceptual framework for network and client adaptation', *IEEE Mobile Networks and Applications* (*MONET*), Vol. 5, No. 4, pp. 221–231.

16 Kobsa A. (2001) 'Generic user modeling systems', *User Modeling and User-Adapted Interaction*, *Ten Year Anniversary Issue*, Vol. 11, No. 1–2, pp. 49–63.

17 The notion of context can be found in various different fields of computer science, for an overview, cf., e.g., [18].

18 Brézillon, P. and Pomerol, J.-Ch. (2001) 'Modeling and using context for system development: Lessons from experiences', in Humphreys, P. and Brézillon, P. (Eds.): *Journal of Decision Systems*, Vol. 10, No. 2, pp. 265–288.

19 Abowd, G.D. (1999) 'Software engineering issues for ubiquitous computing', *Int. Conf. on Software Engineering*, Los Angeles.

20 Note that there are already a few personalisation approaches, considering more facets of context than just user and usage data [21].

21 Kobsa, A., Koenemann, J. and Pohl, W. (2001) 'Personalized hypermedia presentation techniques for improving online customer relationships', *The Knowledge Engineering Review*, Vol. 16, No. 2, pp. 111–155.

22 In Ref. [23], this distinction is referred to as adaptation to difference, which is not necessarily time-dependent and adaptation to change which is seen over time only.

23 Alatalo, T. and Siponen, M.T. (2001) 'Towards the OWLA methodology for development of open, Web/Wireless and adaptive hypermedia information systems', Poster at *ACM HyperText*.

24 Kniesel, G., Noppen, J., Mens, T. and Buckley, J. (2002) 'Report on the first workshop on unanticipated software evolution (USE)', held in conjunction with *The 16th European*

Conference on Object-Oriented Programming, ECOOP2002 Workshop Reader LNCS 2548, Springer.

25 Good, M.D., Whiteside, J.A., Wixon, D.R. and Jones, S.J. (1984) 'Building a user-derived interface', *Communications of the ACM* (*CACM*), Vol. 27, No. 10, October.

26 Carroll, J.M. and Aaronson, A.P. (1988) 'Learning by doing with simulated intelligent help', *Communications of the ACM* (*CACM*), Vol. 31, No. 9, September.

27 Sleeman, D. and Brown, J.S. (Eds.) (1982) *Intelligent Tutoring Systems*, Academic Press, New York, pp. 227–282.

28 Avery, C. and Zeckhauser, R. (1997) 'Recommender systems for evaluating computer messages', *Communications of the ACM* (*CACM*), Vol. 40, No. 3, March.

29 Loeb, S. and Terry, D. (1992) 'Information filtering', *Communications of the ACM* (*CACM*), December, Vol. 35, No. 12.

30 Conklin, E.J. (1987) 'Hypertext: An introduction and survey', *IEEE Computer*, Vol. 2, No. 9, September.

31 Brusilovsky, P. (2001) 'Adaptive hypermedia in Kobsa, A. (Ed.): *User modeling and User Adapted Interaction*', Vol. 11, No. 1/2, pp. 87–110.

32 Brusilovsky, P. and Maybury, M.T. (2002) 'From adaptive hypermedia to adaptive web', Brusilovsky, P. and Maybury M.T. (Eds.): Special Issue on the Adaptive Web, *Communications of the ACM* (*CACM*), Vol. 45, No. 5, pp. 31–33.

33 Want, R. and Schilit, B.N. (2001) 'Expanding the horizons of location-aware computing (Guest Editor's Introduction)', *Computer*, Vol. 34, No. 8, August, pp. 31–34.

34 In this respect, Makimoto and Manners [35] coined the term nomadic information systems being accessible for users on the move using mobile devices.

35 Makimoto, T. and Manners, D. (1997) *Digital Nomad*, John Wiley & Sons.

36 Want, R., Hopper, A., Falco, V. and Gibbons, J. (1992) 'The active badge location system', *ACM Transactions on Information Systems*, Vol. 10, No. 1, pp. 91–102.

37 Eisenstein, J., Vanderdonckt, J. and Puerta, A. (2001) 'Applying model-based techniques to the development of UIs for mobile computers', *5th International Conference on Intelligent User Interfaces* (*IUI*), ACM Press, pp. 69–76.

38 W3C (1999) *PIDL – Personalized Information Description Language*, W3C Note, http://avocado.w3.mag.keio.ac.jp/TR/NOTE-PIDL.

39 W3C (2000) *Composite Capabilities/Preference*, Profiles, http://www.ccpp.org/.

40 W3C (2001) *Device Independence Principles*, W3C Working Draft, http://www.w3.org/TR/di-princ/, September.

41 Dey, A.K., Salber, D. and Abowd, G.D. (2001) 'A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications', Anchor Article of a Special Issue on Context-aware Computing, *Human–Computer Interaction (HCI) Journal*, Vol. 16, No. 2–4, pp. 97–166.

42 Kappel, G., Retschitzegger, W. and Schwinger, W. (2000) 'Modeling customizable web applications – a requirement's perspective', *Int. Conf. on Digital Libraries: Research and Practice*, November, Koyoto, Japan.

43 Kappel, G., Pröll, B., Retschitzegger, W., Schwinger, W. and Hofer, T. (2001) 'Modeling ubiquitous web applications – a comparison of approaches', *Proc. of the Int. Conference on Information Integration and Web-based Applications and Services* (*iiWAS*), September, Linz, Austria.

44 *Unified Modelling Language 1.4 Specification* (2002) Object Management Group (OMG), http://www.omg.org/technology/documents/formal/uml.htm.

45 Note that, personalisation approaches often use the term "usage data" to refer to the application states passed through by a certain user [16].

46 Note that in Ref. [21] the term 'complexity of adaptation' is used to denote if there is a direct relationship between a context and a certain adaptation (e.g., selecting the text only mode vs. selecting a proper adaptation on basis of different user interests and usage data). In our framework, this criteria is covered by the abstraction criteria of the context and the complexity of the adaptation.

47 Kappel, G., Retschitzegger, W. and Schwinger, W. (2001) 'A holistic view on web application development – the WUML approach', *Tutorial notes at 1st Int. Workshop on Web-oriented Software Technology*, June, Valencia, Spain.

48 Oppermann, R. and Specht, M. (1999) 'A nomadic information system for adaptive exhibition guidance', *Int. Conf. on Hypermedia and Interactivity in Museums*, September, Washington.

49 McIlhagga, M., Light, A. and Wakeman, I. (1998) 'Towards a design methodology for adaptive applications', *4th annual ACM/IEEE International Conference on Mobile Computing and Networking*, October, Dallas, TX, USA.

50 Kobsa *et al.* [21] use the term volatility to denote the dynamicity of adaptation but do not distinguish between production of adaptation and presentation.

51 It has to be noted that this category is already included in the first one since the access of a user results in a change of the application state. Because of its importance, however, there is an own category for this scenario.

52 Davies, N., Cheverst, K., Mitchell, K. and Efrat, A. (2001) 'Using and determining location in a context-sensitive tour guide', *IEEE Computer*, Vol. 34, No. 8, August.

53 Cheverst, K., Davies, N., Mitchell, K. and Friday, A. (2000) 'Experiences of developing and deploying a context-aware tourist guide: The GUIDE project', *Proc. of the 6th Int. Conference on Mobile Computing and Networking* (*MOBICOM*), Boston, MA, USA, ACM Press, pp. 20–31.

54 Davis, N., Cheverst, K., Mitchel, K. and Efrat, A. (2001) 'Using and determining location in a context-sensitive tour guide', *IEEE Computer*, August.

55 Cheverst, K., Mitchel, K. and Davis N. (2002) 'The role of adaptive hypermedia in a context-aware tourist guide', *Communications of the ACM* (*CACM*), Vol. 45, No. 5, May.

56 De Bra, P., Aerts, A., Smits, D. and Stash, N. (2002) 'AHA! the next generation', *ACM Conference on Hypertext and Hypermedia*, May.

57 Dey, A.K. and Abowd, G.D. (2000) 'The context toolkit: Aiding the development of context-aware applications', *Workshop on Software Engineering for Wearable and Pervasive Computing*, June, Limerick, Ireland.

58 Dey, A.K., Kortuem, G., Morse, D.R. and Schmidt, A. (2001) 'Situated interaction and context-aware computing', Editorial, *Personal Ubi Comp*, Vol. 5, No. 1, pp. 1–3.

59 Means, W.S. and Harold, E.R. (2001) *XML in a Nutshell, a Desktop Quick Reference*, O'Reilly.

60 Fink, J., Kobsa, A. and Nill, A. (1998) 'Adaptable and adaptive information provision for all users including disabled and elderly people', *New Review of Hypermedia and Multimedia*, Vol. 4, pp. 163–188.

61 Brachman, R.J. and Schmolze, J.G. (1985) 'An overview of the KL-ONE knowledge representation system', *Cognitive Science*, Vol. 9, No. 2, pp. 171–216.

62 Fox, A., Brewer, E., Gribble, S. and Amir, E. (1996) 'Adapting to network and client variability via on-demand dynamic transcoding', *ACM 7th International Conference on Architectural Support for Programming Languages and Operating Systems*.

63 Fox, A., Gribble, S.D., Chawathe, Y. and Brewer, E.A. (1998) 'Adapting to network and client variation using active proxies: Lessons and perspectives', *IEEE Personal Communications*, September.

64 IBM WebSphere Transcoding Publisher 4.0 Specification (2001) *http://www-4.ibm.com/software/webservers/transcoding/*.

65 Mohan, R., Smith, J.R. and Li, C.-S. (1999) 'Adapting multimedia internet content for universal access', *IEEE Trans. on Multimedia*, Vol. 1, No. 1, March, pp. 104–114.

66 Smith, J.R., Mohan, R. and Li, C.-S. (1998) 'Content-based transcoding of images in the internet', *Int. Conf. on Image Processing*, October.

67 Nagao, K., Shirai, Y. and Squire, K. (2001) 'Semantic annotation and transcoding: Making web content more accessible', *IEEE MultiMedia*, April–June, pp. 69–81.

68 Waddington, P. and Gill, P.J. (2002) 'Oracle9i application, server wireless edition in action', *Oracle Magazine*, January–February.

69 Rossi, G., Cappi, J., Fortier, A. and Schwabe, D. (2001) 'Seamless Personalization of E-commerce Applications', *Proc. of the ER-Workshop on Conceptual Modeling Approaches for e-Business* (*eCOMO2001*), November, pp. 457–470.

70 Schwabe, D., Guimarães, R.M. and Rossi, G. (2002) 'Cohesive design of personalized web applications', *IEEE Internet Computing*, Vol. 6, No. 2, pp. 34–43.

71 Gamma, E., Helm, R., Johnson, R. and Vlissides, J. (1995), *Design Patterns – Elements of Reusable Object-Oriented Software*, Addison-Wesley Pub Co.

72 Schwabe, D., Rossi, G., Esmeraldo, L. and Lyardet F. (2001) 'Engineering Web Applications for Reuse' *IEEE Multimedia*, Vol. 8, No. 1, pp. 2–12.

73 Schmidt, A., Aidoo, K.A., Takaluoma, A., Tuomela, U., Van Laerhoven, K. and Van de Velde, W. (1999) 'Advanced interaction in context', *1st Int. Symposium on Handheld and Ubiquitous Computing*, Karlsruhe, Germany, LNCS, Vol. 1707, Springer.

74 Schmidt, A., Beigl, M. and Gellersen, H.W. (1999) 'There is more to context than location', *Computer & Graphics*,Vol. 23, No. 6, December, pp. 893–901.

75 Schmidt, A. and Van Laerhoven, K. (2001) 'How to build smart appliances?', *IEEE Personal Communications*, Vol. 8, No. 4, August, pp. 66–71.

76 Cherniak, M., Franklin, M. and Zdonik, S. (2001) 'Expressing user profiles for data recharging', *IEEE Personal Communications*, July.

77 Finkelstein, A., Savigni, A., Kappel, G., Retschitzegger, W., Kimmerstorfer, E., Schwinger, W., Hofer, Th., Pröll, B. and Feichtner, Ch. (2002) 'Ubiquitous web application development – a framework for understanding', *6th World Multiconference on Systemics, Cybernetics and Informatics*, July, Orlando, Florida, pp. 431–438.

78 Kappel, G., Retschitzegger, W. and Schwinger, W. (2001) 'Modeling ubiquitous web applications – the WUML approach', *Int. Workshop on Data Semantics in Web Information Systems*, November, Yokohama, Japan.

79 Kappel, G., Retschitzegger, W., Kimmerstorfer, E., Schwinger, W., Hofer, Th. and Pröll, B. (2002) 'Towards a generic customisation model for ubiquitous web applications', *Proc. of the 2nd Int. Workshop on Web Oriented Software Technology*, June, Malaga, Spain, pp. 79–104.

80 Kappel, G. *et al*. (2001) 'Bottom-up design of active object-oriented databases', *Communications of the ACM* (*CACM*), Vol. 44, No. 4.