Supporting Systematic Usage of Context in Web Applications

Joachim Wolfgang Kaltz and **Jürgen Ziegler**
IIIS, University of Duisburg-Essen (Campus Duisburg)
Lotharstrasse 65, D-47057 Duisburg, Germany
{joachim.kaltz,juergen.ziegler}@uni-due.de

Abstract

Context can be seen as a paradigm aiming to improve user interaction with software. For Web applications in particular, the issues of content explosion and technological constraints need to be addressed better. This paper discusses an approach for engineering context-aware, adaptive Web applications. This approach integrates context knowledge with domain ontologies, and allows adaptation to be specified for each aspect of application generation. We further describe CATWALK, our implemented framework, which makes use of the context models to enable adaptivity in applications.

Introduction

The amount of information provided to users of Web-based systems is increasing rapidly, often resulting in confusion as to which information is relevant. In addition, though Web-based systems have traditionally focused on information delivery, they are increasingly used as a platform for providing further types of services, often of an interactive nature. This raises the question of which services to provide in which situation, and how to integrate them as seamlessly as possible. The amount of effort a user expends to find information and to use services must be reduced: software should "require human attention for only critical aspects of task execution that require their input" (Thayer & Steenkiste 2003).

The concept of *context* is now used in a variety of fields, see, e.g., (Brézillon *et al.* 2004), (Beigl *et al.* 2003) or (Hernadvölgyi *et al.* 2004), often to improve user interaction with a system. Web applications (and thus Web Engineering), in our view, require a specific focus. Web-based systems have several particular characteristics, of which the "multiplicity of user profiles and the varied operational environments" (Deshpande & Murugesan 2001) are particularly relevant regarding context usage. The fashion in which the system will be used is less predictable than in classic software scenarios: though Web applications are often targeted at certain groups of users, the actual type of user and the usage scenarios can not be known precisely.

Using context systematically in Web Engineering is a challenging but worthwhile effort, as the goal is to positively affect quality of Web-based systems, specifically usability and efficiency aspects as outlined in ISO 9126. The goal of adapting and optimizing the application to context is to make the application more operable for the user. This is of particular interest in scenarios where the user has less time for an interaction and/or less powerful devices. The application hopefully becomes easier to understand and to learn. Optimizing the application in this regard would also positively influence Web performance. The tailored offering decreases Web traffic, as less irrelevant information is exchanged. The user achieves her goals more quickly, so less interaction with the system is required.

In previous work (Kaltz, Ziegler, & Lohmann 2005), we describe a conceptual model where context knowledge is integrated with domain knowledge, and represented in ontologies. Additional challenges, which we now address in this paper, are to describe how context is then used within a Web application and how such a usage can be systematically supported in a run-time environment. We discuss our CATWALK[1] system, which is based on an extensible architecture utilizing model artefacts. The system can be used as a framework to implement adaptive prototypes for Web software. A corresponding prototype, notably demonstrating service incorporation, is shown as illustration.

Scenario

Consider a technician responsible for maintaining industrial installations. The technician uses her company's Web-based system in different situations. When in the office, she accesses an Intranet portal via her desktop browser to gather information regarding the company's products, to update the accountancy of her work, etc. When away from the office, she can use her PDA to check and update her appointment list. When visiting an installation, she may scan the installed product with an integrated RFID-reader in her PDA which reads the product identification. The device then retrieves all relevant details about this product from the company's database, via a Web Service, assuming the technician has network access at this site, e.g., via WLAN. If furthermore the system knows (or assumes) the user's current task is to perform a maintenance inspection, it can provide in addition

[1] Context-aware Adaptation through Transformations for Web Applications Leveraging Knowledge

a dynamically generated graphical user interface for regulating the inspected product via a network interface. Once the technician has returned to the office, if the PDA is used, the system might propose a synchronization service of any work steps performed while offline.

To adapt its offering according to the situation, the system needs to know the context, such as the role of the user (technician) or the hardware device used (desktop, mobile device). The location is also important, although coarse-grained in this scenario: in the office, or away from the office. The current task (make a maintenance inspection, check for appointments, etc.) can likewise be used as input to adapt the system, and may thus also be considered context. Time information can be used to further fine-tune the offering: if the current time is within usual maintenance hours, it is more likely the user is making an inspection call, so this option might be proposed first. If not, another option, e.g., "update appointment schedule", might be presented first.

Generally speaking, the system can attempt to provide the most likely features in a given situation; however it should not disable other possible features, as the precise context of usage can not be known for certain.

Context in Web Engineering

Context-awareness can be used for adaptation in different aspects of a Web application; we summarize potential effects in Table 1.

Aspect	*Adaptation*
navigation	(i) show or hide parts of the navigation structure, (ii) dynamically (re-)construct the navigation structure, (iii) offer additional links (e.g., recommended links).
content	(i) select relevant documents and services, rank them, (ii) adapt the resources themselves, by selecting document parts and integrating them into a final document, (iii) adapt the parameters of integrated services according to context.
presentation	(i) adapt the layout and other user interface characteristics, (ii) adapt to the (potentially changing) constraints of the user's physical device.

Table 1: Adaptation effects in a Web application

By *service* we understand a mechanism through which the application provides the user with a dynamic offering: e.g., a reservation service through which the user can book a flight. Typically, the Web application presents such a service to the user, and to implement the service relies on an actual, back-end service, which is either locally available or is a third-party offering. In our approach, we implement such offerings via Web Services (in the W3C sense).

Previous work (Kaltz, Ziegler, & Lohmann 2005) provides details on our approach to modeling context knowledge. A possible representation of knowledge for the example scenario is shown in Fig. 1. In the underlying model, the

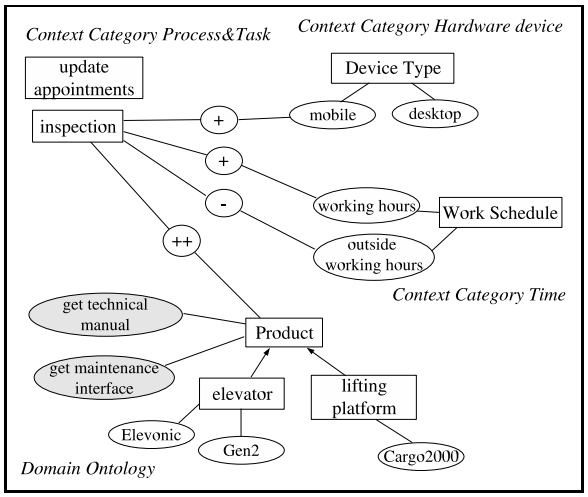

Figure 1: Context knowledge for the example scenario

types of product (e.g., "elevator") are domain knowledge, as are instances of products ("Elevonic"). Services such as "get technical manual" are associated. Context knowledge is modeled as a structuring of context factors and relevance relationships, e.g., for the task "inspection", the concept "Product" is deemed highly relevant; the opposite is likewise true. If the hardware device is "mobile", the task "inspection" is somewhat relevant. Likewise, if time is within working hours, the same task is relevant.

In the following, we discuss an approach to modeling Web applications which integrates the specification of adaptation to context. Our work is situated within the larger scope of the WISE research project[2], which aims to provide a methodology for the systematic development of complex and dynamic Web-based applications. Our contribution to this project is to account for context and enable its usage in run-time Web applications, by augmenting the methodology as shown in Fig. 2.

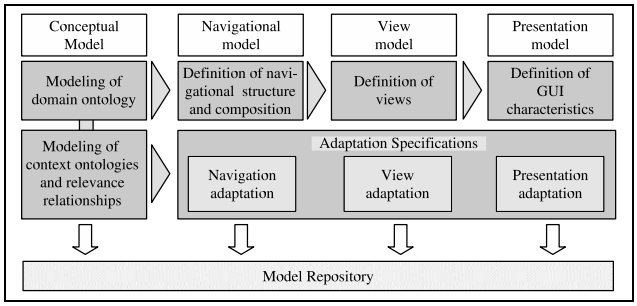

Figure 2: Context modeling in the WISE methodology

[2]Web Information and Service Engineering, BMBF funded project (Nr. 01ISC30F), see http://www.wise-projekt.de

Application Models and Context

The *domain ontology* contains the conceptual knowledge and references to pieces of content: text elements, documents, images and service descriptions. The ontology is represented in OWL. Context factors are categorized and also represented in ontologies. Furthermore, relevance relationships are modeled. Such a relationship is a weighted association between ontology entries, and may involve two or more entries, of arbitrary types.

The *navigation model* describes navigation and composition structures, and is built upon entries of the domain ontology. The model consists of relations describing targets for navigation nodes. The targets may refer to known navigation or content elements of the application. The target of a relation may also refer to an adaptation specification (see below), in which case the target is determined by adaptation.

The *view model* allows to specify which elements of the navigation model are to be viewable in which context. The *presentation model* in turn describes the look & feel of the Web application. The model associates style information with elements used in the navigation model. This association may refer to an adaptation specification, in which case the style information is determined by adaptation. This is possible because style entries can be determined to be in a contextual relationship in the same manner as content entries. Style entries refer to CSS classes, allowing to delegate the actual design to CSS files.

Adaptation Specifications

An adaptation specification determines, on the one hand, which sort of information is to be retrieved or generated for a particular usage in the application, and, on the other hand, how the modeled relevance relationships are to be used in combination with the ontological knowledge to determine contextual relations in a particular situation. An adaptation specification has the following properties (see also Fig. 3):

- the types of items to be computed;
- the maximum number of items;
- the context relevance threshold;
- a list of ontologies. If specified, all retrieved items must belong to one of these ontologies;
- a list of context categories. If specified, only these categories will be considered for determining context-relevant information;
- a list of "pre-adaptation" and /or "post-adaptation" items, which are fixed domain items to be retrieved, but only when context-relevant items are found;
- a list of domain exploration properties, defining ontology exploration strategies for computing contextually relevant items; i.e., to find, in addition to items involved in a (modeled) relevance relationship, items in specific ontological relations with these. Each property defines:
  - a type of relation in the ontology that should be followed. The type name may be arbitrary, in which case the name can be used for (semantically free) lookup in the ontology; or a name with predefined meaning (e.g., "superclass") can be used, allowing a more intelligent adaptation mechanism;
  - the depth of exploration for this relation;
  - the percentage the contextual relationship remains relevant upon each level of relation following.

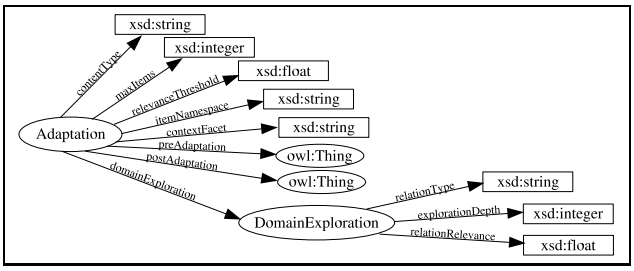

Figure 3: Adaptation specification definition

System Design & Implementation

We now describe CATWALK, our framework for context. It supports the systematic usage of adaptation in Web applications, by providing run-time components which make use of the artefacts resulting from application modeling as described above.

Apache Cocoon as Foundation

We view Apache Cocoon as a particularly appropriate foundation for generating context-aware Web applications. Adaptation can be achieved by sequencing XML transformations through controller logic, as in (Fiala *et al.* 2004): "According to the user/platform profile [document generation] is subdued to a series of XSLT transformations, each considering a certain adaptation aspect".

Our work generalizes this approach: each step of the application generation process is an XML transformation, whereas this transformation can be specified by XSLT instructions or by a custom (Cocoon-based) "transformer" component. If the functionality a transformation step must provide is well-determined, it can be specified as an XSLT stylesheet, XSLT being a general purpose XML transformation language. If instead the functionality required in a processing step is complex (e.g., to integrate arbitrary adaptation results), a custom transformer component is used. The generation process is specified as Cocoon pipelines.

Components for Context

CATWALK consists of components for context, XSLT stylesheets, and pipeline fragments which define processing steps integrating these components. The components in effect augment Cocoon; i.e., they are extensions of existing Cocoon component types, and thus may be used within regular Cocoon processing. Fig. 4 describes the CATWALK architecture and a typical processing flow in this architecture. The client request is matched in a Cocoon pipeline and processing flows through custom components, ultimately resulting in a response to the client (e.g., a page of a Web site).

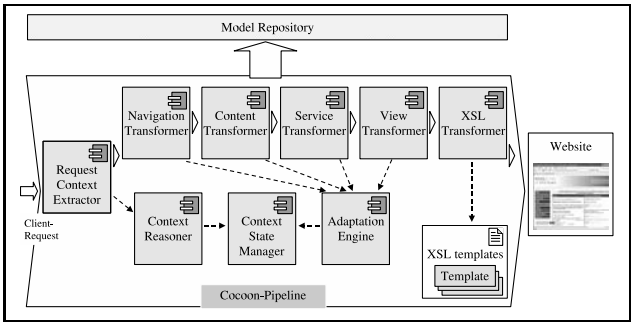

Figure 4: CATWALK architecture

Each component implements a specific concern, in the sense of the separation of concerns architectural design principle (Fielding & Taylor 2002), and is realized by one or more Java classes and possibly additional artefacts (such as XSLT stylesheets).

The *ContextStateManager* handles access to the context, that is, it allows other components to obtain information about context, and stores any (present and past) context which has been recognized. The framework provides a default implementation which handles storage via the current Web session of the user.

The *RequestContextExtractor* is responsible for determining the context of the user's current request; that is, for translating any parameters of the request (present in the URL and in the protocol) into the terminology and granularity used in the application models. Cocoon configuration mechanisms are used to specify which implementation accomplishes the translation for which category of context.

The *RequestContextExtractor* transmits the current context information to the *ContextReasoner*. This component in turn is responsible for determining the degree of activity of the domain and context ontology elements; for this determination, varying implementations may use different techniques, such as spreading activation, and/or heuristics based on analyzing past context activation degrees in addition to currently recognized factors. The default implementation uses present and past user navigation to compute the domain context, and sets the other sorts of context according to current request parameters. The *ContextReasoner* then updates context state via the *ContextStateManager*.

The *AdaptationEngine* computes items which are appropriate to the current context. A further functionality is to support context-sensitive service integration, by computing default values for service parameters according to context. CATWALK provides a default implementation which implements the strategy defined by an adaptation specification such as described above. The JENA toolkit (McBride 2001) is used for reading the ontologies, to traverse them and to search for specific statements.

The actual application generation is achieved by several transformer components, each of which may call upon the *AdaptationEngine*. The *NavigationTransformer* generates navigational information for the current request, meaning a structured representation of the nodes that the user can reach from the current position; some of these nodes may be generated according to context. The role of the *ContentTransformer* is to generate content appropriate to the current node. Some content items associated to the node may be placeholders for adaptation, in which case the component delegates to the *AdaptationEngine*. Any pieces of content which refer to services are now processed by the *ServiceTransformer*, which has two roles:

- if the service requires user interaction, the transformer determines the required parameters for this service, computes default values according to context (by using the *AdaptationEngine*) and generates an XML representation such that an ulterior GUI generation step will have sufficient information about the service;
- if the service is to be called, the transformer calls the Web Service and converts its response into a form which, again, may be used by a GUI generation step.

The *ViewTransformer* is the last custom transformation component in the CATWALK processing sequence; it is responsible for determining view and presentation information for the generated pieces of content. As discussed above, presentation determination can be guided by context to determine an appropriate style. The XML representation output by this component is then usable by XSLT stylesheets for GUI generation.

Prototype Web Application

As proof of concept of the framework for context, a prototype Web application representing a fictitious e-commerce site was modeled. Navigation adaptation is illustrated in Fig. 5. In this demonstration, the context is "summer"; i.e., in the

Figure 5: Navigation adaptation demonstration

"time" context ontology the instance "summer" of "Season" is active. The navigation model contains, for the navigation node labeled as "Clothing tips", an adaptation specification designed to retrieve domain elements which can be offered as links. As a consequence, CATWALK adds within this navigation node additional navigation possibilities in the order of context-relevance. The node labeled as "Outdoor running shoes" is considered most relevant, as it is in a direct relevance relationship with "summer" in the model. The adaptation specification in this scenario instructs the system

to explore super-class relations up to a depth of two, therefore two levels of super-classes of "Outdoor running shoes" are also contextually relevant, however to a lesser degree, as this specification states a loss of relevance for such an ontological relationship.

Content and presentation adaptation are achieved similarly, as is service selection. Regarding service integration, additional adaptation is possible. A Web Service representing the functionality of finding auctions of clothing items is used; it provides a "search" operation with 3 parameters: category, season and size. In the application model, knowledge of the service's existence is specified, including contextual hints for the parameters: the parameter "Category" is associated with a concept in the domain ontology ("wear"); the "Season" with the time ontology (as a category of context); "Size" refers to an arbitrary attribute with this name.

When the user navigates to "Stores", the system generates an interface to the auction service, as this service is associated to this node in the navigation model. The *AdaptationEngine* can provide default parameters as shown in Fig. 6. The values are computed according to context; this com-

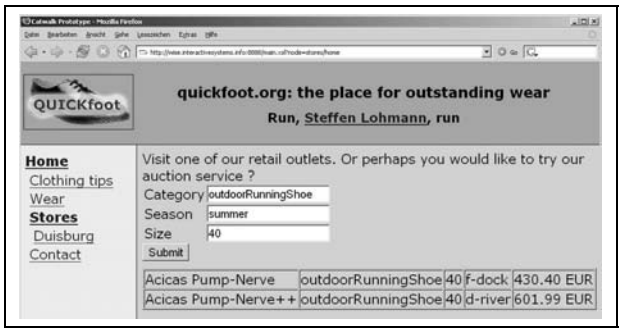

Figure 6: Service adaptation demonstration

putation is guided by the contextual hints. In the scenario shown here, the user has recently viewed product information on "Outdoor running shoes", either by menu navigation or by following a recommended link such as generated by navigation adaptation. Furthermore, the user has identified herself to the system. The "Category" field thus contains the most relevant domain context (resulting from user navigation); in this case the "Outdoor running shoes", being a sub-category of wear. The "Season" is read from the category "time" within the context state; the "Size" is found in the user context, which includes the user's known attributes. In the screenshot, the user has furthermore called the service; results are rendered according to a "table" user interface template.

Context in the Engineering Process

The people involved in the Web application development process can extend conventional features with adaptation to context by using the methods described in this work, and by basing the run-time application on the CATWALK framework. This process can involve several roles:

1. context-modelers define factors within context categories and their relations with the domain model;
2. adaptation modelers specify where adaptation is to occur in the application, what sort of adaptation is desired and what strategies are to be used;
3. context-sensing implementors configure, complete or extend the components responsible for sensing;
4. adaptation implementors extend or replace the adaptation component to modify ontology inference mechanisms;
5. application testers and quality experts use the generated application as a prototype to verify the effects of context on the application.

Roles 3 and 4 are optional: the framework contains a default implementation and interpretation for sensing and adaptation. This means that adaptive Web applications can be achieved by means of context and navigation description only, without requiring additional software programming. This in turn allows for rapid prototyping of such applications; the prototyping is further supported by a context simulation feature. In particular, this provides Web engineers with a platform in which to observe the effects of context modeling and adaptation strategies on the actual adaptation occurring in Web applications, thus allowing to evaluate and refine these, or to study new strategies.

Related Work

General-purpose methodologies for Web applications such as WSDM (De Troyer & Leune 1998) provide systematic means for specifying an application. However, they have limited support for integrating dynamic resources and for steering the system according to context. eW3DT (Scharl 2000) or WebUML (Conallen 2002) do explicitly allow to model dynamic resources; however these models are used only for design, and not within a run-time system. For achieving context-awareness in a system however, this is problematic: a way of integrating context information with dynamic resources must be available, and the run-time system must have dynamic access to the modeled knowledge.

Research projects in adaptivity often explore a specific scenario, and thus, according to (Kappel *et al.* 2003), architectural considerations in research often apply only to the described scenario. Work which aims at a general-purpose architecture includes (Winograd 2001) and (Cannataro, Cuzzocrea, & Pugliese 2002). The former is an example of a distributed architecture, well-suited to periodically gather environment data, e.g., location sensing. The latter is an example of an integrated architecture, in this case for adaptive hypermedia. A particular contribution of our work is the systematic separation of concerns regarding context into a component architecture, with the goal of simplifying the task of replacing or extending mechanisms related to context processing. In addition, our approach permits service incorporation; i.e., the integration of interactive offerings within the application.

Regarding the integration of services in particular, (Keidl & Kemper 2004) introduce a context processing architecture for Web Services. Our work in turn focuses on integrating services within a general Web Engineering process accounting for context.

Conclusions & Discussion

The main purpose of this work is to systemize and facilitate usage of context within Web applications. Such applications may then offer functionality in an adaptive manner, the goal of which is to enhance the interaction of users with the system's offerings.

For this purpose, knowledge about context is structured and, along with domain knowledge and its relation to context, represented in ontologies. Application models are built upon these ontologies and include adaptation specifications. The model as a whole is uniform, permitting the usage of common adaptation mechanisms for the various stages of application generation. A system architecture is provided to make use of the modeled knowledge, wherein each functional concern is handled separately. The resulting system is in effect a framework which generates a dynamic, adaptive Web application.

The system is designed to be extensible. On the one hand, the component-based approach to generation and adaptation allows to replace or extend the way each concern is implemented as a software component. On the other hand, application knowledge is specified in ontologies; i.e., with formal semantics. This is a prerequisite for reasoning support; that is, for allowing for complex inferences to be mechanically achieved. Existing reasoners such as RACER (Haarslev, Möller, & Wessel 2004) can be plugged in via JENA. Furthermore, the chosen XML content-based architecture addresses the information-centered nature of Web applications, while at the same time enabling dynamic, interactive offering by way of service descriptions, which are considered in a similar manner as pieces of content.

A limitation in the approach presented is the support for arbitrary sensing mechanisms and data structures, in particular the matching of sensed parameters onto context factors as known in the application model. Supporting arbitrary sensing in a general-purpose, unifying approach is problematic. In particular, there are no existing Web standards specifying how environment information may be defined and how a Web client would communicate these to the Web server. Current approaches must thus define specific mechanisms for transmitting client context information in an application-relevant way, and require custom programming for context sensing when the required information does not conform to generally valid patterns. Such a task in turn can be facilitated by a context-sensing framework such as described in (Dey, Abowd, & Salber 2001).

References

Beigl, M.; Krohn, A.; Zimmer, T.; Decker, C.; and Robinson, P. 2003. AwareCon: Situation Aware Context Communication. In *Proceedings of UbiComp 2003: Ubiquitous Computing*, 132–139. Berlin: Springer.

Brézillon, P.; Borges, M.; Pino, J.; and Pomerol, J.-C. 2004. Context-awareness in group work: three case studies. In *IFIP International Conference on Decision Support Systems (DSS-2004)*, 115–124. Monash University, Australia.

Cannataro, M.; Cuzzocrea, A.; and Pugliese, A. 2002. XAHM: an adaptive hypermedia model based on XML. In *Proceedings of the 14th international conference on Software engineering and knowledge engineering*, 627–634. New York, NY, USA: ACM Press.

Conallen, J. 2002. *Building Web applications with UML (2nd Edition)*. Reading, Mass.: Addison-Wesley.

De Troyer, O. M. F., and Leune, C. J. 1998. WSDM: a user centered design method for Web sites. *Comput. Netw. ISDN Syst.* 30(1-7):85–94.

Deshpande, Y., and Murugesan, S. 2001. Summary of the second ICSE workshop on web engineering. *SIGSOFT Softw. Eng. Notes* 26(1):76–77.

Dey, A. K.; Abowd, G. D.; and Salber, D. 2001. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. *Human Computer Interaction* 16(2-4).

Fiala, Z.; Hinz, M.; Houben, G.-J.; and Frasincar, F. 2004. Design and implementation of component-based adaptive Web presentations. In *Proceedings of the 2004 ACM symposium on Applied computing*, 1698–1704. New York, NY, USA: ACM Press.

Fielding, R. T., and Taylor, R. N. 2002. Principled design of the modern Web architecture. *ACM Trans. Inter. Tech.* 2(2):115–150.

Haarslev, V.; Möller, R.; and Wessel, M. 2004. Querying the Semantic Web with Racer + nRQL. In *Proceedings of the KI-2004 International Workshop on Applications of Description Logics*. CEUR.

Hernadvölgyi, I.; Ucelli, G.; Symonova, O.; Delpero, L.; and de Amicis, R. 2004. Shape Semantics from Shape Context. In *Proceedings of the KI-2004 International Workshop on Modelling and Retrieval of Context, Vol-114*. CEUR.

Kaltz, J. W.; Ziegler, J.; and Lohmann, S. 2005. Context-aware Web Engineering: Modeling and Applications. *RIA — Revue d'Intelligence Artificielle, Special Issue on Applying Context Management* 19(3):439–458.

Kappel, G.; Pröll, B.; Retschitzegger, W.; and Schwinger, W. 2003. Customisation for Ubiquitous Web Applications - A Comparison of Approaches. *Int. J. Web Eng. Technol.* 1(1):79–111.

Keidl, M., and Kemper, A. 2004. Towards context-aware adaptable web services. In *Proceedings of the 13th international World Wide Web conference - Alternate Track Papers & Posters*, 55–65. New York, NY, USA: ACM Press.

McBride, B. 2001. Jena: Implementing the RDF Model and Syntax Specification. In *Proceedings of the WWW2001, Semantic Web Workshop*.

Scharl, A. 2000. *Evolutionary Web Development*. Secaucus, NJ, USA: Springer-Verlag.

Thayer, S. M., and Steenkiste, P. 2003. An Architecture for the Integration of Physical and Informational Spaces. *Personal Ubiquitous Comput.* 7(2):82–90.

Winograd, T. 2001. Architectures for Context. *Human-Computer Interaction* 16:401–419.