Ersch. in: Web Information Systems Engineering – WISE 2007 : 8th International Conference on Web Information Systems Engineering, Nancy, France, December 3 - 7, 2007; proceedings / Boualem Benatallah ... (eds.). - Berlin [u.a.] : Springer, 2007. - S. 398-409. - (Lecture notes in computer science ; Vol. 4831). - ISBN 978-3-540-76992-7

# An Object-Oriented Version Model for Context-Aware Data Management

Michael Grossniklaus and Moira C. Norrie

Institute for Information Systems, ETH Zurich
8092 Zurich, Switzerland
{grossniklaus,norrie}@inf.ethz.ch

**Abstract.** Context-aware computing is a major trend in mobile computing, pervasive computing and web engineering. Several models, frameworks and infrastructures have been developed to represent, process and manage context. While most of these approaches support the adaptation of application logic based on context, the requirements of context-aware systems in terms of data management have received little attention. This is most apparent in the field of web engineering as many web sites are data-intensive and require context-dependent content adaptation to support internationalisation, personalisation and multiple channels. We present a version model featuring alternative versions for context-aware data management and query processing that has been integrated in an object-oriented database system. Finally, we also describe the implementation of a mobile tourist information system based on this system.

## 1 Introduction

Context-aware computing has become an important issue in several application domains. Although the requirements that motivate the need for context-awareness vary from one field of computing to the next, context is widely regarded as a valuable resource that can help to improve applications. In mobile computing context information is used to address the challenges that arise from the reduced interaction bandwidth that characterises most portable devices. Working with a device that has a small screen and limited input facilities can be made less awkward by adapting the application to the current environment of the users. In pervasive computing, the role of context is even more prominent as applications are often not built for standard devices with traditional user interfaces. Rather these applications are completely embedded in the user's environment and have to rely on sensed context information for interaction.

In this paper, we focus on context-awareness in web engineering. As the Web was originally developed to exchange information, the management and delivery of information has always been a primary concern. Also, due to requirements resulting from globalisation, personalisation and multi-channel access, many different forms of context information, both technical and cultural in nature, can be witnessed in web engineering. Over time, the Web has evolved into an omnipresent platform that nowadays also supports complex applications. As a consequence, well-established web standards are no longer only used to build web

sites, but have also been adopted in the realisation of both mobile and pervasive systems. In a sense, web engineering has come full circle as its technologies are now being used to build context-aware systems that were originally motivated by ubiquitous computing.

Efforts towards context-awareness in web engineering are well documented by several model-based approaches that have been extended or designed to provide such features. The personalisation of web sites constitutes one form of adaptation to the user's context and is supported by several methodologies such as OOHDM [1], OO-H [2] and SiteLang [3]. In contrast to these systems that adapt to a single user, WSDM [4] proposes audience-driven personalisation to adapt to a group of users. The Hera [5] methodology combines concepts from adaptive hypermedia with semantic web technologies and therefore supported adaptivity from the start. In Hera, adaptability denotes the ability to adapt a web site to users or device capabilities at design-time, while adaptivity is the adaptation of an application at run-time. Finally, WebML [6] which maybe marks the most comprehensive approach to model-driven web site development has been extended to model context-aware and adaptive web applications [7].

While all of these model-based approaches provide possibilities to specify context-aware adaptation, general platforms that support the implementation of this functionality have not yet emerged. Some design methodologies such as Hera and WebML have introduced their own run-time systems. However, these platforms are often tailored to proprietary features of a methodology or do not even support all capabilities of the model. To adapt the application logic of a web application, the use of aspect-oriented programming has been proposed [8] in the setting of UML-based Web Engineering (UWE) [9]. Unfortunately, similar concepts that would allow the adaptation of the application content at the level of database systems have not yet emerged. In this paper, we present a solution for context-aware data management that consists of a version model supporting context-dependent alternatives and a matching algorithm as the basis for context-aware query processing.

We begin in Sect. 2 with a discussion of related work and background information documenting the origins of our approach. The version model for context-aware data management is presented in Sect. 3 together with the definition of the notion of context used in our work. Section 4 discusses context-aware query processing based on this version model. In Sect. 5, we present a mobile information system that was implemented using a database system extended with the presented version model. Finally, in Sect. 6 we give concluding remarks.

## 2 Related Work and Background

An early approach to deliver content according to the device context of the client is generally known as distilling or transcoding [10,11,12] on a proxy server. Common forms of adaptation include splitting large documents into smaller fragments that can be handled by the client as well as changing the format, colour depth or size of an image. Usually, these transformations are effected

online and thus result in reduced performance. Several solutions to this problem have been proposed, among them the caching of previous results [13] of the distillation process and the pre-computation of content variants [14].

Another solution is proposed by the web authoring language Intensional HTML (IHTML) [15]. Based on version control mechanisms, IHTML supports web pages that have different variants for different user-defined contexts. The concepts proposed by IHTML were later generalised to form the basis for Multidimensional XML [16] which in turn provided the foundation for Multidimensional Semistructured Data (MSSD) [17]. MSSD is represented using a graph model with multidimensional nodes and context edges. Multidimensional nodes capture entities that have multiple variants by grouping the nodes representing the facets. These variants are connected to the multidimensional node using context edges. The label of a context edge specifies in which context the variant pointed to is appropriate. Based on this graph representation, a Multidimensional Query Language [18] has been defined that allows the specification of context conditions at the level of the language. Thus, it can be used to formulate queries that process data across different contexts.

As mentioned previously, some model-based approaches provide a proprietary implementation platform designed to support some or all context-aware features of the methodology. For example, the WebML run-time which is part of the WebRatio Site Development Studio [19] has been extended to allow the execution of context-aware and adaptive web applications. WebRatio is implemented using industry standards such as relational database management systems, Java-based application servers, XML and XSLT. Both the data and navigation models are rendered in XML and passed together with an XSLT representation of the presentation design to an automatic code generator. Apart from this data, the code generator also uses metadata describing data mappings and WebML-specific tag libraries to produce Java Server Pages that are deployed to an application server.

Similar to the approach taken by WebML, web sites designed with Hera can be implemented using the Hera Presentation Generator [20]. It combines the data stored using the Resource Description Framework (RDF) with the models represented in RDF Schema to generate an adapted presentation. This presentation is rendered as a set of static documents that contain the mark-up and content for one particular class of clients. A more dynamic implementation platform for Hera has been proposed based on the AMACONT project [21]. Based on a layered component-based XML document format, reusable elements of a web site can be defined at different levels of granularity. Adaptation is realised by allowing components of all granularities to have variants. A variant of a component specifies an arbitrarily complex selection condition as part of the metadata in its header. AMACONT's publishing process is based on a pipeline that iteratively applies transformations to a set of input documents to obtain the fully rendered output documents.

A more general and extensible architecture to support context-aware data access has been proposed based on profiles and configurations. Context is represented as a collection of profiles that each specify one aspect of context such

as the user or device. Profiles are expressed according to the General Profile Model [22] that provides a graphical notation and is general enough to capture a wide variety of formats currently in use to transmit context information as well as transforming from one format to another. Configurations have three parts matching the general architecture of web information systems in terms of content, structure and presentation. Configurations are stored in a repository on the server side and matched to the profiles submitted by the client as part of its request. The matching is based on adaptation rules consisting of a parametrised profile, a condition and a parametrised configuration [23]. During the matching process, the client profile is compared to adaptation rules. If the client profile matches the parametrised profile of the rule and the specified values fulfil the condition, the parametrised configuration is instantiated and applied.

Although all of these approaches provide some functionality for creating context-aware systems, none of them offers support for the management of context-aware data. As a consequence, this neglect requires that context information about application data is modelled explicitly in the data model. For example, to implement a simple multi-lingual web site otherwise elegant and intuitive data models have to be extended with additional concepts to capture the language context and thus become overcrowded and cumbersome. Another drawback of such stratum approaches is the fact that they scale very poorly when multiple context dimensions are required. Therefore, instead of building frameworks for context-aware data access on top of existing database management systems, we believe that this issue needs to be addressed at the level of data storage and querying inside the database itself. Inspired by version models proposed in the domain of engineering databases [24] and software configuration systems [25], we propose a two-dimensional version model that, in addition to revisional versions, supports alternative versions to represent a data object in different contexts. While the basic organisation of our version model is closely related to previous approaches, the process of evaluating context-aware queries is very different to that used in existing version models to compute configurations. In engineering databases and software configuration systems, "context" information supplied with the query constitutes an integral part of the specification of the query results. Context as seen in mobile and ubiquitous computing as well as web engineering is instead information that can be used to refine the result of a query that otherwise has well-defined default behaviour. As a consequence, our query processor uses an algorithm based on a best match rather than an exact match to select the variant that is most appropriate in a given context. In the next sections, we will present our version model and give further detail how this notion of context has affected its design.

## 3 Version Model

In the setting of our context-aware data management system, context information is regarded as optional information used by the system to augment the result of a query rather than specifying it. As a consequence, such a system also

needs a well-defined default behaviour that can serve as a fall-back in the absence of context information. In our approach, context information is gathered outside the database system by the client application. Therefore, it is necessary that client applications can influence the context information used during query processing. To support this, a common context representation that is shared by both components is required. Since several frameworks for context gathering, management and augmentation already exist, our intention was to provide a representation that is as general as possible. The following definitions specify the notion and representation of context as used by our system. We will use the given sets NAMES and VALUES to denote the sets of legal context dimension names and context values, respectively.

**Definition 1.** *A* context space *represented by $S$ denotes which context dimensions are relevant to an application of the version model for context-aware data management. It is defined as $S = \{name_1, name_2, \ldots name_n\}$ such that $\forall\, i : 1 \leq i \leq n \Rightarrow name_i \in \mathsf{NAMES}$ and therefore $S \subseteq \mathsf{NAMES}$.*

**Definition 2.** *A* context value *$c$ is defined as a tuple $c = \langle name, value \rangle$ where $name \in \mathsf{NAMES}$ and $value \in \mathsf{VALUES}$.*

**Definition 3.** *$C(S)$ denotes a* context *for a context space $S$ and is represented as an unordered set of context values*

$$\begin{aligned} C(S) &= \{\langle name_1, value_1\rangle, \langle name_2, value_2\rangle, \ldots, \langle name_m, value_m\rangle\} \\ &= \{c_1, c_2, \ldots, c_m\} \end{aligned}$$

*such that $\forall\, i : 1 \leq i \leq m \Rightarrow name_i \in S$ and $\forall\, c_i, c_j \in C : i \neq j \Rightarrow name_i \neq name_j$.*

**Definition 4.** *A* context state *denoted by $C_\star(S)$ is a special context, where $\forall\, name \in S : \exists\, \langle name, value\rangle \in C_\star(S)$.*

Our version model for context-aware data management has been specified within the framework of an object-oriented database management system developed at our institute. As this database management system is built on the concepts defined by the OM data model [26], we defined our model as an extension of OM. OM is a rich and flexible object-oriented data model that features multiple instantiation, multiple inheritance and a bidirectional association concept. This model was chosen as, due to its generality, it can be used to represent other conceptual models such as the Entity-Relationship model or RDF.

As the OM model supports multiple instantiation, an object is represented by a number of instances—one for every type of which the object is an instance. All instances of an object share the same object identifier but are distinguishable based on their instance type. For the purpose of our version model, we have broken this relationship between the object and its instances and introduced the additional concepts of alternative and revisional versions. As shown in Fig. 1, in the extended OM model, an object is associated with a number of alternative

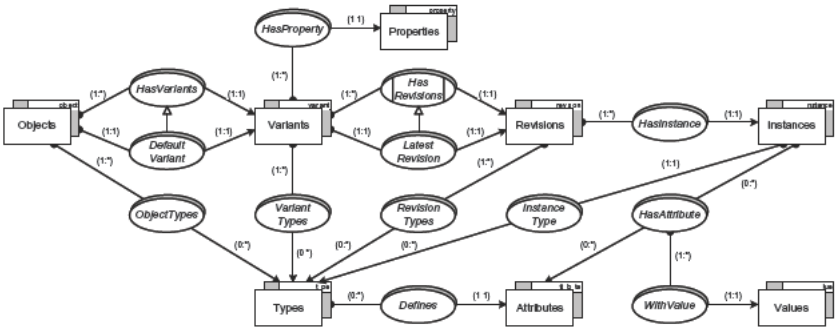


**Fig. 1.** Conceptual data model of an object

versions (Variants) which in turn are each linked to a set of revisional versions (Revisions). Finally, each revision is connected to the set of instances containing the actual data. As can be seen from the figure, our model supports two versioning dimensions. Variants are intended to enable context-aware query processing while revisions support the tracking of the development process. However, for the scope of this paper we will focus on variants exclusively and neglect the presence of revisional versions in the model. Note that all versions of an object still share the same object identifier tying them together as a single conceptual unit. A variant of an object is identified by a *variant context* $C_{var}(S)$ that is stored as a set of properties represented as $\langle name, value \rangle$ context values.

## 4 Context-Aware Query Processing

Context-aware queries over multi-variant objects are evaluated using the matching algorithm shown in Fig. 2. Whenever an object is accessed by the query processor, the algorithm selects the object variant that best matches the current context state of the system $C_\star(S)$. First it retrieves all variants of the input object $o$ linked to it through the *HasVariants* association. After building the context state of each variant from the properties associated to it through *HasProperty*, the algorithm applies a scoring function $f_s$ to this variant context state that returns a value measuring how appropriate the variant is in the current context. It then returns the variant of $o$ with the highest score $s_{max}$. If the highest score is below a certain threshold $s_{min}$ or if there are multiple variants with that score, the default variant is returned. This fall-back mechanism has been introduced into the algorithm to guarantee a well-defined default behaviour.

Similar to context dimensions, the concrete definition of the scoring function depends on the requirements of a context-aware application. Our system therefore allows the default scoring function to be substituted with an application-specific function. In order to give an indication of what is involved in designing such a scoring function, we will describe the *general scoring function*, as specified in Def. 5, that is used in our system as a default. To allow greater flexibility in specifying

```
MATCH(o, C⋆(S))
1  V0 ← rng(HasVariants dr({o}))
2  V1 ← V0 ∝ (x → (x × rng(HasProperty dr({x}))))
3  V2 ← V1 ∝ (x → (dom(x) × fs(C⋆(S), rng(x))))
4  smax ← max(rng(V2))
5  V3 ← V2 % (x → rng(x) = smax)
6  if |V3| = 1 ∧ smax ≥ smin
7     then v ← V3 nth 1
8     else  v ← rng(DefaultVariant dr({o})) nth 1
9  return v
```

**Fig. 2.** Matching algorithm

**Table 1.** Value syntax

| Set | Syntax | Description | Examples |
|---|---|---|---|
| ATOM | $x$ | Atomic value | `en`, `27` |
| SET | $x_1\{: x_i\}$ | Set of atomic values $S = \{x_1, \ldots, x_n\}$ | `at:ch:de`, `red:blue` |
| RANGE | $x_{min}..x_{max}$ | Range of atomic values $I = [x_{min}, x_{max}]$ | `5.5..7.0`, `a..f` |
| STAR | $*$ | Wildcard | `*` |

the current system context as well as to describe object variants, we have partitioned the given set VALUES into four subsets as shown in Tab. 1. Based on these sets, the *value* field of a context value $c$ can be used to specify more than one value. The behaviour resulting from using identity to define equality can still be obtained by using values of set ATOM. In addition, collections of atomic values can be expressed using a value of SET. Values defined by RANGE have been introduced to allow intervals to be defined based on a lower and upper bound. Finally, if all possible values of a context dimension are permissible, the wildcard value from set STAR can be used. Providing a wildcard value in this setting might seem paradoxical at first as one might argue that the corresponding context value could simply be omitted. Its right to exist will become clear when we present how the scoring function computes the score value based on the matching values.

A matching condition ($\cong$) for context values has been defined. As context values are used in our approach to describe both the current context state of the system and the context of a variant, any combination of these four given sets has to be considered. As the discussion of all of the resulting sixteen cases is clearly out of the scope of this paper, we refrain from going into more details here. Intuitively, two context values match if they are either the same, contained in one another or if a wildcard value is involved. Our general scoring function also supports two mechanisms that have been created to give more control over the matching process to the client of our system. On the one hand, an application can assign weights to each of its context dimensions to be used when computing

the score of an object variant. Required and illegal matches can also be specified by prefixing the corresponding context value with + or −, respectively. If, for example, a client requires that all data is delivered in English, it would specify the context value $\langle lang, +en \rangle$ in the current context state $C_\star(S)$. Analogously, an object variant that is not intended for a novice user could be described by the context value $\langle level, -novice \rangle$. To compare such prefixed context values, a second matching condition ($\cong_\pm$) has been introduced that also takes required and illegal values into consideration.

**Definition 5.** *The* general scoring function $f_s$ *takes two contexts* $C_1$ *and* $C_2$ *as arguments and returns a scoring value representing the number of matching context dimensions of the two contexts normalised by* $|N|$*. It is defined as*

$$f_s(C_1, C_2) = \frac{1}{|N|} \sum_{n \in N} (w(n) \times f_i(n, C_1, C_2)) \times \prod_{n \in N} f_\pm(n, C_1, C_2)$$

*where* $N$ *denotes the union* $N_1 \cup N_2$ *of the sets of all names of context values specified either by* $C_1$ *or* $C_2$*, respectively. The indicator function* $f_i$ *is defined as*

$$f_i(n, C_1, C_2) = \begin{cases} 1 & \exists\, c_1 \in C_1, c_2 \in C_2 : \\ & \quad name_1 = name_2 = n \wedge value_1 \cong value_2 \\ 0 & otherwise. \end{cases}$$

*The context dimension weight function* $w$ *returns a weight* $w(n) \in \mathbb{R}^+$ *for every name* $n \in N$*, where* $N$ *represents the set of the names of all context dimensions. Finally, the matching function for prefixed values* $f_\pm$ *is defined as*

$$f_\pm(n, C_1, C_2) = \begin{cases} 1 & \exists\, c_1 \in C_1, c_2 \in C_2 : \\ & \quad name_1 = name_2 = n \wedge value_1 \cong_\pm value_2 \\ 0 & otherwise. \end{cases}$$

## 5 Mobile Information System

The version model and query processing presented has been implemented in the OMS database system which provides native support for concepts of the OM data model. Using this extended OMS system, we have also built the Extensible Content Management System (XCM) [27] based on the separation of the four concepts of content, structure, view and layout [28]. To corroborate our claim that it is vital to place context-awareness also at the level of the database system, we will describe how XCM served as a content server in a multi-channel mobile tourist information system designed to assist visitors to the Edinburgh festivals held each year during the month of August. An overview of the architecture of the so-called EdFest system [29] is shown in Fig. 3.

The clients supported by EdFest are shown on the left of the figure. Apart from traditional clients such as desktop PCs and PDAs, EdFest introduced a novel interaction channel based on interactive paper [30]. Our context-aware

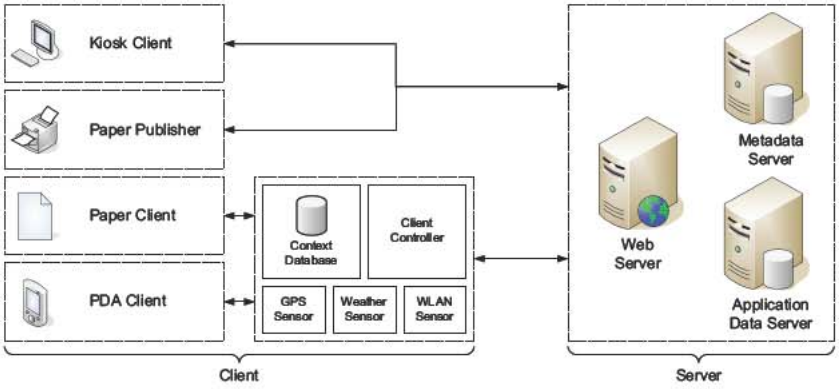


**Fig. 3.** Overview of the EdFest architecture

content management system is shown on the right of the figure. It consists of a web server handling the communication with the clients, a server managing publishing metadata and an application database that stores the content of the EdFest system. While the kiosk and PDA clients are implemented using standard HTML, the paper client actually consists of two publishing channels. The paper publisher [31] channel is used at design time to author and print the interactive documents from the content stored in the application database. The paper client channel is then active at run-time when the system is used and is responsible for delivering additional information about festival venues and events through voice feedback when the tourists interact with the paper documents.

As the paper and PDA client are mobile, they have been integrated with the platform shown at the centre of the figure that manages various aspects of context. Sensors gather location, weather and network availability information that is then managed in a dedicated context database. Context information is sent from the client to the server by encoding it in the requests sent to the content management server. This is done by the client controller component that acts as a transparent proxy that enriches requests with the current context state stored in the context database. For instance, to access the `getEventInfo` information service, the paper client would send the following request to the server.

```
http://.../xcm?anchor=getEventInfo&event=o2451
```

After intercepting and augmenting the request, the extended URI given below is issued to the XCM server by the client controller proxy.

```
http://.../xcm?anchor=getEventInfo&event=o2451&format=+vxml&
              lang=en&user=guest
```

By inserting these parameters into the request, the client provides information about the format it requires, the preferred language and the current system user.

To discuss how context-aware data management is used in XCM, it is necessary to understand its four-step publishing process shown in Fig. 4. As XCM uses

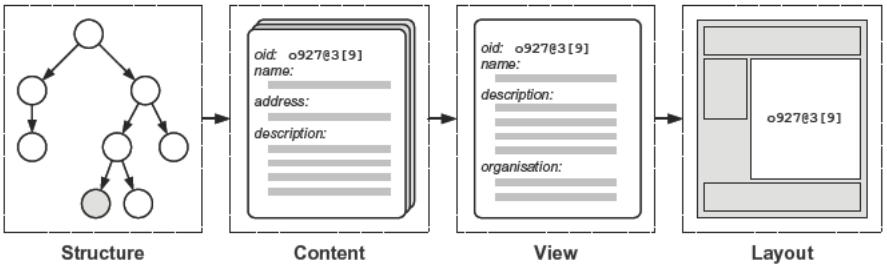


**Fig. 4.** Publishing process

a composite pattern to capture structure, the first step depends on whether the object to publish is a container or content object. If the client requests a container object, the publishing process has to be applied recursively to all components within the container. If a content object is requested, the publishing process is executed only once. Assume that the grey circle in the figure represents the component object requested by the client. Since it is one of the leaf nodes of the structure tree, it represents a content object and can be published directly. As shown in the figure, the content object could represent a person object with attributes *name*, *address* and *description*. The content publisher then retrieves the view object associated with the content object and applies it. In our example, the application of the view object results in the hiding of attribute *address* and the appearance of aggregated content about the *organisation* this person works for. Finally, the publishing component retrieves the presentation objects for the component object. By applying the template to the object, the final presentation is generated.

XCM uses dedicated databases to manage application data and publishing metadata. Both of these databases have been extended with the presented version model. Thus, in each step of the publishing process, a context-aware query is issued to access both data and metadata. As a consequence, XCM supports context-dependent adaptation for content, structure, view and layout. In the EdFest system, this built-in capability of the database system was used to provide location-based services as well as multi-channel and multi-modal access to the system in a uniform way. Further, since in OM everything—even concepts of its metamodel—is represented as objects, it has been possible to address the need for context-aware interactions [32] for different publishing channels with the concept of object variants for method objects.

## 6 Conclusions

We have motivated the need for implementation platforms that support context-aware data management by collecting requirements from mobile and pervasive computing as well as from model-based design methods for web engineering. We believe the issues related to context-aware data management can only be addressed by providing adequate concepts at the level of the data management

system itself. We have shown how version models from engineering databases and software configuration systems could be adapted for context-dependent data management. In contrast to these systems, context is considered as information that refines rather than specifies a query result and therefore requires a different form of query processing that uses a best match function to select object variants.

The presentation of a content management system and an application developed using our version model has illustrated how we were able to address several concerns of context-aware systems in a uniform way through the concept of object variants. In the future, we plan to investigate the potential of this version model by integrating our platform with an existing model-based web design methodology.

## References


1. Rossi, G., Schwabe, D., Guimarães, R.: Designing Personalized Web Applications. In: Proc. Intl. Conf. on World Wide Web, Hong Kong, China (2001)
2. Garrigós, I., Casteleyn, S., Gómez, J.: A Structured Approach to Personalize Websites Using the OO-H Personalization Framework. In: Proc. Asia Pacific Web Conf., Shanghai, China (2005)
3. Schewe, K.D., Thalheim, B.: Reasoning About Web Information Systems Using Story Algebras. In: Benczúr, A.A., Demetrovics, J., Gottlob, G. (eds.) ADBIS 2004. LNCS, vol. 3255, Springer, Heidelberg (2004)
4. Casteleyn, S., De Troyer, O., Brockmans, S.: Design Time Support for Adaptive Behavior in Web Sites. In: Proc. ACM Symp. on Applied Computing, Melbourne, FL, USA (2003)
5. Frăsincar, F., Barna, P., Houben, G.J., Fiala, Z.: Adaptation and Reuse in Designing Web Information Systems. In: Proc. Intl. Conf. on Information Technology: Coding and Computing, Las Vegas, NV, USA (2004)
6. Ceri, S., Fraternali, P., Bongio, A., Brambilla, M., Comai, S., Matera, M.: Designing Data-Intensive Web Applications. The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann Publishers Inc., San Francisco (2002)
7. Ceri, S., Daniel, F., Matera, M., Facca, F.M.: Model-driven Development of Context-Aware Web Applications. ACM Trans. on Internet Technology 7(2) (2007)
8. Baumeister, H., Knapp, A., Koch, N., Zhang, G.: Modelling Adaptivity with Aspects. In: Lowe, D.G., Gaedke, M. (eds.) ICWE 2005. LNCS, vol. 3579, Springer, Heidelberg (2005)
9. Koch, N.: Software Engineering for Adaptive Hypermedia System. PhD thesis, Ludwig-Maximilians-University Munich, Munich, Germany (2000)
10. Fox, A., Brewer, E.A.: Reducing WWW Latency and Bandwidth Requirements by Real-time Distillation. Computer Networks and ISDN Systems 28(7-11) (1996)
11. Fox, A., Gribbe, S.D., Brewer, E.A., Amir, E.: Adapting to Network and Client Variability via On-Demand Dynamic Distillation. Computer Architecture News 24(Special Issue) (1996)
12. Smith, J.R., Mohan, R., Li, C.S.: Transcoding Internet Content for Heterogeneous Client Devices. In: Proc. Intl. Symp. on Circuits and Systems, Monterey, CA, USA (1998)
13. Singh, A., Trivedi, A., Ramamritham, K., Shenoy, P.: PTC: Proxies that Transcode and Cache in Heterogeneous Web Client Environments. World Wide Web 7(1) (2004)

14. Mohan, R., Smith, J.R., Li, C.S.: Adapting Multimedia Internet Content for Universal Access. IEEE Transactions on Multimedia 1(1) (1999)
15. Wadge, W.W., Brown, G., Schraefel, M.C., Yildirim, T.: Intensional HTML. In: Munson, E.V., Nicholas, C., Wood, D. (eds.) PODDP 1998 and PODP 1998. LNCS, vol. 1481, Springer, Heidelberg (1998)
16. Stavrakas, Y., Gergatsoulis, M., Rondogiannis, P.: Multidimensional XML. In: Proc. Intl. Workshop on Distributed Communities on the Web, Quebec City, Canada (2000)
17. Stavrakas, Y., Gergatsoulis, M.: Multidimensional Semistructured Data: Representing Context-Dependent Information on the Web. In: Pidduck, A.B., Mylopoulos, J., Woo, C.C., Ozsu, M.T. (eds.) CAiSE 2002. LNCS, vol. 2348, Springer, Heidelberg (2002)
18. Stavrakas, Y., Pristouris, K., Efandis, A., Sellis, T.: Implementing a Query Language for Context-Dependent Semistructured Data. In: Benczúr, A.A., Demetrovics, J., Gottlob, G. (eds.) ADBIS 2004. LNCS, vol. 3255, Springer, Heidelberg (2004)
19. WebRatio: Site Development Studio (2001), http://www.webratio.com
20. Frăsincar, F.: Hypermedia Presentation Generation for Semantic Web Information Systems. PhD thesis, Technische Universiteit Eindhoven, Eindhoven, The Netherlands (2005)
21. Fiala, Z., Frăsincar, F., Hinz, M., Houben, G.J., Barna, P., Meissner, K.: Engineering the Presentation Layer of Adaptable Web Information Systems. In: Proc. Intl. Conf. on Web Engineering, Munich, Germany (2004)
22. De Virgilio, R., Torlone, R.: Management of Heterogeneous Profiles in Context-Aware Adaptive Information System. In: Proc. OTM Workshop on the Move to Meaningful Internet Systems, Agia Napa, Cyprus (2005)
23. De Virgilio, R., Torlone, R., Houben, G.J.: A Rule-based Approach to Content Delivery Adaptation in Web Information Systems. In: Proc. Intl. Conf. on Mobile Data Management, Nara, Japan (2006)
24. Katz, R.H.: Toward a Unified Framework for Version Modeling in Engineering Databases. ACM Computing Surveys 22(4) (1990)
25. Conradi, R., Westfechtel, B.: Version Models for Software Configuration Management. ACM Computing Surveys 30(2) (1998)
26. Norrie, M.C.: An Extended Entity-Relationship Approach to Data Management in Object-Oriented Systems. In: Proc. Intl. Conf. on the Entity-Relationship Approach, Arlington, TX, USA (1994)
27. Belotti, R., Decurtins, C., Grossniklaus, M., Norrie, M.C., Palinginis, A.: Interplay of Content and Context. Journal of Web Engineering 4(1) (2005)
28. Grossniklaus, M., Norrie, M.C.: Information Concepts for Content Management. In: Proc. Intl. Workshop on Data Semantics and Web Information Systems, Singapore (2002)
29. Norrie, M.C., Signer, B., Grossniklaus, M., Belotti, R., Decurtins, C., Weibel, N.: Context-Aware Platform for Mobile Data Management. Wireless Networks (2007), http://dx.doi.org/10.1007/s11276-006-9858-y
30. Signer, B.: Fundamental Concepts for Interactive Paper and Cross-Media Information Spaces. PhD thesis, ETH Zurich, Switzerland (2006)
31. Norrie, M.C., Signer, B., Weibel, N.: Print-n-Link: Weaving the Paper Web. In: Proc. ACM Symp. on Document Engineering, Amsterdam, The Netherlands (2006)
32. Grossniklaus, M., Norrie, M.C.: Using Object Variants to Support Context-Aware Interactions. In: Proc. Intl. Workshop on Adaptation and Evolution in Web Systems Engineering, Como, Italy (2007)