Journal of Web Engineering, Vol. 7, No. 3 (2008) 200–219

SUPPORTING DIFFERENT PATTERNS OF INTERACTION THROUGH CONTEXT-AWARE DATA MANAGEMENT

MICHAEL GROSSNIKLAUS and MOIRA C. NORRIE
Institute for Information Systems, ETH Zurich
8092 Zurich, Switzerland
{*grossniklaus,norrie*}*@inf.ethz.ch*

Received November 11, 2007
Revised May 13, 2008

Ubiquitous and mobile computing often introduce novel modes of interaction with different interaction patterns than those typical of traditional desktop applications. Therefore, there is a need to extend general models and systems for context-awareness to include adaptation of interaction styles to context. We present an object-oriented data management system that supports context-awareness through a notion of multi-variant objects and describe how it was used to implement context-aware interaction patterns. Our approach was motivated by our experiences of developing a mobile application that offered an interface based on a set of interactive paper documents alongside a regular web interface and we use this example to explain the issues and our solution in detail.

Keywords: Context-Awareness, Patterns of Interaction, Multi-Channel Access, Mobile Tourist Information Systems, Version Models

1 Introduction

Developments in ubiquitous and mobile computing have led to challenges of how to support a wide variety of novel forms of user interaction. A general aim is to minimise explicit user interactions to allow users to focus on the task at the hand. This can be done by using implicit actions such as user motion or changes in the environment to control or influence system operation. At the same time, the limitations of mobile devices in supporting traditional styles of interaction based on keyboard, mouse and visual display, have driven researchers to investigate other forms of interaction for users on the move. Our experiences have shown that supporting mobile and ubiquitous applications often involves working with, not only new modes of interaction, but also radically different patterns of interaction. In contrast to traditional interfaces, input data may be assembled from various sources and in different orders rather than being specified either in a single step or a well-defined sequence of steps. Thus, there is a tendency to move from linear to non-linear interaction patterns.

Context-awareness has become a common feature of ubiquitous and mobile applications to allow them to adapt to user situations. Most of the applications, however, have requirements that surpass the forms of adaptation that can be done statically or on a per request basis. Often, these applications need to be able to dynamically adapt over a series of requests as the interaction pattern between client and server can also depend on the user, their environment and the task at hand. Yet this is an aspect of context-awareness that has received little

attention so far. Web technologies have been widely adopted as a platform for the development of ubiquitous and mobile applications. Correspondingly, web engineering no longer deals only with methods and tools to support the development of applications accessed through traditional desktop browsers, but also with issues such as multi-channel access and context-dependent adaptation.

Adaptation of hypertexts [5] in terms of content, structure and presentation is a well-researched topic in web engineering. At the level of models and frameworks to support web engineering, several generic approaches have been proposed to empower application developers to determine what notions of context and adaptation are relevant to specific applications. General context models have been developed that can cater for various forms of adaptation that correspond to personalisation, internationalisation, multi-channel access, location-awareness etc. The need to adapt processes consisting of several interaction steps is also widely recognised by the web engineering community [34] and several conceptual and model-driven approaches exist that support this form of adaptation. In these approaches, different styles of user interaction are often addressed by adapting the structure or navigation to the different context factors that are relevant to an application. Motivated by the requirements for context-awareness that have been identified at the conceptual level, we have addressed the issue of how these can be supported at the implementation level. In this paper, we present a solution based on previous work on context-aware data management [19].

Some model-based approaches already offer support for implementing context-aware applications by means of an integrated implementation platform tailored to the capabilities and requirements of the respective model. Most approaches, however, rely on standard components such as application servers, content management systems or relational databases to implement the modelled specifications. Unfortunately, as we will see, these implementation platforms do not provide native support for context-awareness. As a consequence, this functionality has often to be implemented over and over again leading to poor reuse of code and maintainability. This shortcoming is especially visible at the level of data management as application data models often become obfuscated with context-related concepts. We, therefore, decided to investigate how an object-oriented data management system extended to support context-aware applications could be used to support all required forms of adaptation in an elegant way. In previous work, we have already demonstrated how such a database system can be used to adapt the content of a web application to context [3]. In this work, we will show how dynamic adaptation of the structure or navigation of an application at the conceptual level can be mapped to context-aware database operations to support different interaction patterns at the implementation level.

We use an example of a mobile tourist information system that we developed in a previous project to motivate the need for context-aware interaction patterns and to describe how these could be implemented using our data management system. Alongside more traditional web interfaces, the system offers a set of interactive paper documents that can be used to access information about festival events and venues using digital pen and paper technology and an audio output channel. A complex transaction such as reserving tickets for an event requires several data parameters to be selected and it would be too restrictive to insist that this be done in a particular order. Without a visual display, users need to be carefully guided through the interaction so that they are aware of the current interaction state and therefore

an important factor was to supply users with context-dependent help information according to the interaction state.

We begin in Section 2 with a discussion of related work and a motivation of our approach. Section 3 gives an overview of context-aware data management and how it can be realised by extending an object-oriented database system with the notion of multi-variant objects. In Section 4, we introduce the idea of context-aware patterns of interaction that we propose as an implementation level concept to realise the kind of adaptive and non-linear navigation that is often required in mobile or multi-channel systems. As a motivation, we use the well-known example of a ticket reservation process and present how this process was realised within a mobile tourist information system that uses paper as an input channel. In Section 5, we present the general implementation of this tourist information system and then discuss specifically how the mechanisms used to provide context-aware data management can be applied to support context-aware interaction patterns. Section 6 provides a general discussion of the approach and some of the outstanding issues. Finally, concluding remarks are given in Section 7.

2 Related Work

The need for context-awareness is well documented in the field of web engineering. Its impact can be witnessed in several model-based approaches and a few recently proposed implementation platforms. For example, the Web Modelling Language (WebML) [8] has been extended with primitives that allow adaptive and context-aware sites to be modelled [7]. To manage context information, the data model is extended with a context model. Two additional units have been introduced to gather context information. Each context-dependent page is associated with a context cloud that specifies the adaptation operations. When a context-aware page is requested, the corresponding operations are executed and the page is adapted accordingly. However, in order to adapt the content itself, the context-dependent entities in the data model have to be associated with entities representing the relevant context dimensions. Depending on the complexity of the application, this can lead to a very cumbersome data model that is no longer true to the orthogonal nature of context.

The specification of adaptation is an integral part of the Hera methodology [23]. Hera distinguishes between static design-time adaptation called adaptability and dynamic run-time adaptation called adaptivity. All design artefacts can be adapted by annotating them with appearance conditions. Web sites designed with Hera are implemented by using the models to configure a run-time environment. The Hera Presentation Generator (HPG) [17], for example, combines the data stored as RDF with the models represented in RDFS to generate a presentation that is adapted at design-time according to user preferences as well as device capabilities. An alternative implementation platform for Hera is based on the AMACONT [15] project. Using a layered component-based XML document format [16], reusable elements of a web site can be defined at different levels of granularity. Adaptation at run-time is realised by allowing components of all granularities to have variants.

In UML-based Web Engineering (UWE) [25], adaptation is based on the Munich Reference Model [26] for adaptive hypermedia applications. The architecture and concepts of this reference model are based entirely on the Dexter [22] and AHAM [9] reference models. The use of UML, however, offers both a graphical representation and a formal specification using

the Object Constraint Language (OCL). The Munich Reference Model distinguishes three forms of rule-based adaptation—adaptive content, adaptive links and adaptive presentation. A shortcoming of this rule-based approach is that the rules exist outside the model and thus have no graphical representation. A possible solution to this problem has been proposed through the use of aspect-oriented modelling techniques [1].

The Web Site Design Method (WSDM) [11] emphasises audience modelling to classify and identify the different kinds of users of the web site. In WSDM, all models are built on audience classes to adapt the web site for different kinds of users. Adaptive behaviour in WSDM [6] is specified using the Adaptation Specification Language (ASL) that allows the structure of web sites to be influenced at run-time. Based on audience modelling, WSDM can also be used to design localised web sites [10]. Although localised features can be specified quite easily at the conceptual level, things become complicated as the development process moves to the implementation phase. From the point of view of data modelling, the different localities have to be mapped onto a standard database schema. De Troyer and Casteleyn [10] discuss, as an example, labelling the columns or tables of a relational database with the name of the corresponding locality. While such an approach might be sufficient if very few localities need to be supported, it scales poorly in the case of several potentially overlapping localities, and even becomes infeasible if additional dimensions other than language are required.

As previously mentioned, we believe that adaptive navigation specified at the conceptual level translates to context-aware patterns of interaction at the implementation level. The concept of volatile functionality [36, 18] that comprises volatile content, navigation and presentation, is related to our work in the sense that volatile navigation also deals with frequently changing styles of how content is accessed. The object-oriented design methodology OOHDM [37] has recently been extended to allow the modelling of volatile requirements at the conceptual level and an implementation platform called CAZON is provided that builds on JSP tag libraries and traditional databases to integrate volatile functionality. As a consequence of using these existing technologies that are not aware of context, similar issues as the ones discussed for localised sites in WSDM arise at the implementation level.

Another group of web-based systems that provide more complex forms of interaction are Rich Internet Applications (RIA). The need for design methodologies that allow such applications to be specified and implemented has been identified [35] and corresponding proposals based on several existing modelling methods have been made. For example, WebML has been extended [4] with additional units at the conceptual level. At the implementation level, the WebRatio CASE tool was modified to support the generation of client-side code. Another approach with the same goal is the Rich User Experience (RUX) model [27]. RUX models are implemented using the RUX-Tool that generates the client-side functionality. As the implementation platforms of these approaches are again built on standard technologies that are oblivious to the requirements of context-awareness, they suffer from the already described drawbacks. However, these approaches are intended to support the generation of advanced client-side user interfaces, rather than the dynamic adaptation of navigation and interfaces to a context. Therefore, the requirement for context-aware patterns of interaction is less pronounced in these systems.

So far, we have looked at the most influential conceptual models for web engineering and, in some cases, their underlying implementation platform. Apart from these, general technologies

to support context-awareness and adaptation haven been developed. An example of such a solution is the web authoring language Intensional HTML (IHTML) [44]. Based on version control mechanisms, IHTML supports web pages that have different variants and adapts them to a user-defined context. The concepts proposed by IHTML were later generalised to form the basis for the definition of Multidimensional XML (MXML) which in turn provided the foundation for Multidimensional Semistructured Data (MSSD) [42]. Similar to semi-structured data that is often modelled using the Object Exchange Model (OEM), MSSD is represented in terms of the Multidimensional Object Exchange Model (MOEM). Based on this representation, the Multidimensional Query Language (MQL) [43] allows the specification of context conditions to formulate queries that process data across different contexts.

A general and extensible architecture that supports context-aware data access is proposed in [12]. In this approach, profiles are used to describe the context in which a request has been issued to the web information system and are expressed according to the General Profile Model (GPM) [13]. Additionally, configurations express how the response should be generated and consist of three parts that match the general architecture of web information systems in terms of content, structure and presentation. A matching process compares client rules to adaptation rules consisting of a parametrised profile, a condition and a parametrised configuration [14]. 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.

It is apparent that numerous proposals for context-dependent adaptation of web applications have been made in the past. While these approaches differ in terms of the form and scope of adaptation that they support, most of them address the problem of adaptive web sites at the conceptual level. Model-driven methodologies provide comprehensive support to design context-aware web applications and are, therefore, of paramount importance in the development of such systems. Nevertheless, we feel that the functionality specified at the conceptual level should also be supported by adequate technologies at the implementation level. As we have seen in this section, some model-driven approaches provide their own implementation platforms. For the lack of better alternatives, these platforms are built on traditional technologies that feature no native support for context-awareness. This often makes it necessary to either build a layer of indirection on top of these technologies or to distort the models with context information that should be managed elsewhere. In the remainder of this paper, we will present our approach to implementing adaptive systems and, in particular, context-dependent interaction processes based on previous work in context-aware data management that is summarised in the next section.

3 Context-Aware Data Management

In this section, we will present an overview over context-aware data management at the level of a database system that we have developed in previous work [21, 19]. At the heart of our solution stands an object-oriented version model that supports the notion of multi-variant objects and that was implemented within a database system developed at our institute [45]. As this database management system is built on the concepts defined by the OM [29] model, we have decided to define 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.

To support multiple instantiation, i.e. the ability of a single object to have multiple instances that exist on different paths along the inheritance graph, an object in the original OM model is represented by a number of instances—one for every type of which the object is an instance. For the purpose of multi-variant objects, we have broken this relationship between the object and its instances and introduced the additional concept of a variant. As shown in the conceptual data model represented in Figure 1, in the extended OM model, an object is associated with a number of variants which in turn are each linked to a set of 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. As in the traditional OM model, objects can be instantiated with multiple types and therefore revisions can be related to any number of instances.

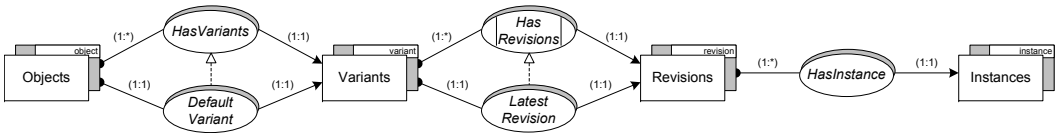


Fig. 1: Conceptual data model of an object

Before presenting how context-dependent queries are evaluated by our system, it is necessary to briefly introduce the notion and representation of context that we are using. In the setting of our context-aware information system, context information is regarded as optional information that is used by the system for augmenting 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 information system by the client application. Therefore, it is necessary that client applications can influence the context information that is used during query processing by the information system. To support this, a common context representation that is shared by both components is required. 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 $\forall\, i : 1 \leq i \leq n \Rightarrow name_i \in$ NAMES and therefore $S \subseteq$ NAMES.*

Each context dimension *name* can be associated with a *value* to form a *context value* $c = \langle name, value \rangle$.

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

Then, a *context* $C(S)$ is a set of context values for the dimensions specified by S.

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* $\forall\, c_i, c_j \in C : i \neq j \Rightarrow name_i \neq name_j$.

Finally, a *context space* denoted by $C_\star(S)$ is a special context that contains exactly one value for every context dimension of S.

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

While contexts are used to describe in which situation a particular variant of an object is appropriate, the current context state of the system governs how context-dependent queries are evaluated. To evaluate context-aware queries over multi-variant objects, the matching algorithm shown in Figure 2 is used. Whenever the query processor accesses an object, the object variant that best matches the current context state of the system $C_\star(S)$ is selected by the algorithm based on the values returned for each variant by the scoring function f_s.

```
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

Our system allows the default scoring function to be substituted with an application-specific function. To give an overview of what is involved in designing a scoring function, we will describe the *general scoring function*, as specified in Definition 5, that is used in our system as a default. The general scoring function allows for situations where more complex context descriptions are required. Greater flexibility in specifying the current system context as well as to describe object variants is achieved by partitioning the above mentioned given set VALUES. Based on these subsets, the *value* field of a context value c can be used to specify more than one value such as set, range or wildcard values. For context values specified using these given sets, a condition ($\cong$) has been defined that determines if two values match. Additionally, the general scoring function also supports two mechanisms that have been introduced to give more control over the matching process. First, an application can assign weights to each of its context dimensions that will then be used by the system when it computes the score of

an object variant. Second, required and illegal matches can also be specified by prefixing the corresponding context value with + or −, respectively. 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*

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

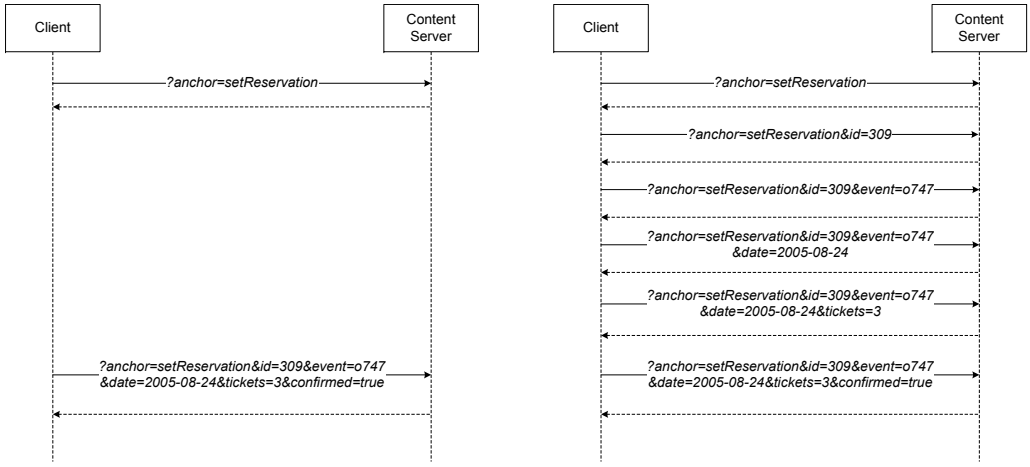
The context dimension weight function w *returns a weight* $w(n) \in \mathbb{R}^+$ *for every name* $n \in N$*, where* N *represents the set of the names of all context dimensions. Finally, the matching function for prefixed values* $f_{\pm}$ *is defined as*

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

4 Context-Aware Patterns of Interaction

As presented in Section 2, the need to alter the sequence of steps required to complete a process is often mapped to adaptation of the structure or navigation of a web site to the requirements of a client. Several approaches that deal with this issue at the conceptual level have already been proposed. In this section, we return to the well-known example of a ticket reservation process that can either be completed in one step or with multiple intermediate steps. However, in contrast to conceptual and model-driven approaches discussed before, we will focus exclusively on the implementation of processes that can have an arbitrary number steps. We introduce the idea of context-aware interaction patterns that we propose as an implementation technique for adaptive navigation. Further, by presenting the paper-based user interface of a mobile tourist information system, we will make a case for non-linear interaction processes, a concept that has received little attention to date. As we will show, context-aware patterns of interaction also provide an elegant solution for implementing non-linear interaction processes.

Different navigation structures that are defined for a process at the conceptual level can also be witnessed by examining the sequence of requests and responses between client and server. As an example, two communication patterns that result from different navigation structures are shown in Figure 3. The sequence of requests and responses that is generated by a one-step navigation is depicted in Figure 3(a), whereas the communication pattern of the



(a) One-step process (b) Multi-step process

Fig. 3: Communications patterns of a reservation process

same process when completed in multiple steps is given in Figure 3(b). As can be seen in the figure, completing the reservation process in one step results in two request and response pairs where the first retrieves the initial response and the second uploads all values to the server for processing. The picture in the case of the multi-step process is quite different, as each data value required to process the reservation request is sent to the server encoded in an individual request. A well-known issue that arises with multi-step processes, is the fact that either the client or server needs to manage a session to keep track of the parameters that have already been set. Depending on the capabilities of the client device, this session can be managed on the client which would result in the communication pattern shown in Figure 3(b). If the client browser does not support sessions, as is often the case in mobile devices, the server has to manage the session. In this case, it is necessary to rewrite the links contained in the server's responses to include a session identifier. The requests of the communication pattern in this case would then only carry information about the session and the current value.

From an implementation point of view, supporting multiple navigation structures in a web application at the same time can be quite challenging. Handling intermediate steps introduces an overhead in terms of application code as the server component that handles the requests has to check which parameters have already been set and update the the application database accordingly. Also, intermediate steps require additional error checking and handling routines as values are no longer validated at the end of the process but after each step. In summary, before the server can respond to a request, the code that implements a certain process first has to figure out at which intermediate step of the process the client is. As the current step of the process is determined based on the parameters that are passed to the server, these values can be seen as contributing to the context in which the functionality was called. The solution for context-aware data management presented in the previous section has specifically been designed to address this problem of matching a context state to a set of alternative versions and to resolve the best matching variant. We, therefore, propose to take advantage of this existing

solution for context-awareness to factor out the server code that deals with analysing the transmitted parameters to determine what needs to be done. The OMS database management system [45] that we have extended with the notion of multi-variant objects represents both data and metadata as objects. As a consequence, it is possible to define object methods as well as database macros that have multiple implementation variants, each for a given step in the process. At runtime, the context resolver will then take care of selecting the appropriate version of the multi-variant operation that implements the update functionality.

As mentioned above, we have used context-aware interaction patterns to implement the reservation functionality in the EdFest tourist information system [41, 31, 40]. The tourist information system was designed to assist visitors to the city of Edinburgh during the art festivals held each year during the month of August. In contrast to other tourist information systems that attempt to replace existing delivery channels, such as newspaper listings, festival brochures and flyers, with one device that integrates all functionalities, our system aims at recognising the reasons why these channels have been established. Our approach is, therefore, rather based on augmenting and consolidating existing means of conveying information to tourists than replacing them with new delivery channels. The EdFest mobile information browser is based on the interactive paper technology described by Norrie et al. [30]. To interact with digital paper an electronic pen is required that determines its position on the paper and sends information requests to the content server. Since digital paper technologies capable of dynamically displaying information on paper are currently not readily available, the result of a user interaction can generally not be displayed on paper but has instead to be delivered through another channel.

Existing multi-channel information systems that support clients, such as web browsers, mobile phones, media phones or PDAs have addressed the requirement of making all aspects of the content delivery context-dependent. Although these traditional delivery channels differ substantially in their capabilities, the way in which users interact with them is rather similar. All of these devices display information graphically, use links for navigation and forms for data acquisition. The addition of interactive paper to the set of delivery channels invalidates this property of multi-channel information systems. In the case of interactive paper, the communication patterns between client and server, as well as the application logic required to process the interaction on the server, diverge considerably from traditional delivery channels.

To substantiate this claim, Figure 4 shows the paper interfaces of reservation process within the EdFest system. As can be seen from the figure, the reservation process involves multiple documents, a bookmark shown in Figure 4(a) and a brochure containing events as the one presented in in Figure 4(b). The reservation process is started when a tourist points with the digital pen to the icon labelled *Start reservation*, triggering a voice response from the server asking them to select the event. Events can be selected in the brochure document by touching the icon below the event title with the pen. The tourist is then prompted to specify the date of the performance that they would like to book tickets for. The date is also selected in the brochure by pointing into the timeline at the bottom of the corresponding event description. Then, the tourists are asked to select the number of tickets on the bookmark. After they have done so, a summary of their reservation is read to them, which they can acknowledge by clicking on the *reserve* icon on the bookmark. The booking is then processed and a voice confirmation is sent to the client.

(a) Bookmark (b) Brochure

Fig. 4: Paper interface of the reservation process

To conclude this section, we will have a look a some of the considerable differences in both requirements and properties between the paper channel and more traditional delivery channels in more detail. One such issue is the way in which the different channels handle and check for errors. In a traditional form-based interface, a user is free to input any values they like. As a consequence, the server implementation needs to parse the values, check them and, if necessary, engage in error handling. In the case of interactive paper, the situation is quite different. Since all values that can be selected by the user have been preauthored and printed as physical documents, it is virtually impossible to select illegal values. However, the digital paper presents other challenges that might not be apparent at first sight. A minor issue is the fact that links cannot be rewritten once the physical documents have been printed. This fact limits the number of choices available to implement session management between client and server as described above. A more serious problem is the possibility of non-linear navigation processes. To guide a user through a multi-step process, a traditional user interface can display a sequence of pages that are interlinked to take the user from one step to the next or, in the case of an error, to the faulting step. This constitutes a considerable amount of control over the user as the only thing they can do is follow the process step by step or abort it prematurely. With interactive paper, the users have the whole user interface including all links readily available in the form of the physical documents and are free to point their pen wherever they choose. As a consequence, there is almost no control over the user's navigation process as they can step through the interface in any order. For example, in the reservation process described above it would also be possible to select the number of tickets before the actual event has been selected. This depart from linear interaction patterns is, however, not restricted to interactive paper applications. In the fields of mobile and ubiquitous computing, user studies carried out with other pervasive and ambient interfaces have indicated that interaction processes might divert more fundamentally from the ones known from desktop applications. In the next section, we will discuss how these additional challenges of context-aware interaction patterns have been implemented with multi-variant operations.

5 Implementation

Having introduced the idea of context-aware patterns of interaction and their mapping to multi-variant operations, we will discuss their implementation in detail in this section. We start with a brief discussion of the overall implementation of the EdFest mobile tourist infor-

mation system followed by an in-depth presentation of the implementation of context-aware interaction patterns. An overview of the architecture of the EdFest system is given in Figure 5. The server component of the tourist information system is shown on the right-hand side of the figure, whereas the range of currently supported clients is shown on the left. The server module consists of three major components. Communication with clients is handled by a web server that retrieves and renders content in the appropriate format. The content is stored on an application data server shown on the far right-hand side of the figure, whereas publishing metadata such as presentation templates as well as structural and navigation information are stored on a metadata server.

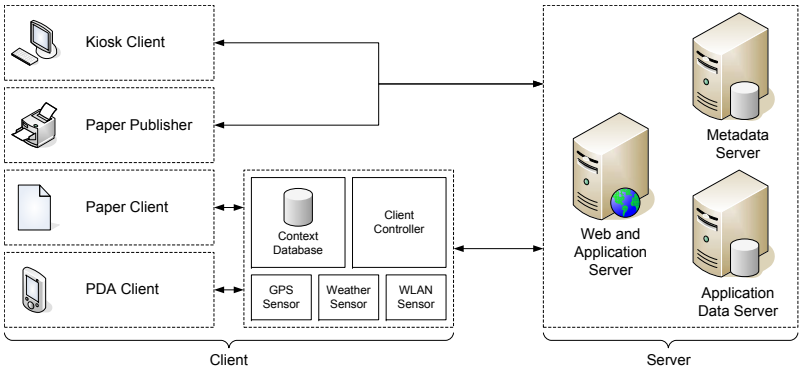


Fig. 5: Overview of the EdFest architecture

On the left-hand side of the figure, the clients for which the system currently provides support are listed. The *Kiosk Client* shown at the top represents the traditional access channel of browsing tourist information using an HTML browser on a stationary computer. Through the kiosk web site, all functionality of the EdFest system can be accessed. Below the kiosk client, the two output channels for the interactive paper client are listed. The *Paper Publisher* client is responsible for creating and printing interactive paper documents based on a specific XML format. The rendering of this XML is then done by the paper publisher when it generates a printable document. More detailed information about the publishing process of interactive paper as used in the EdFest system can be found in Norrie et al. [33, 32]. At run-time, the *Paper Client* uses a cross-media information server [39] to access the digital content managed by the server whenever tourists interact with the augmented paper using their digital pen. A tourist pointing to a particular area on the paper triggers the retrieval of content from the server. As mentioned before, it is currently not possible to display the response of such a request on paper. The EdFest system, therefore, uses voice output to deliver information to the tourist. As an alternative to the interactive paper client, a mobile *PDA Client* can be used in the scope of the EdFest project. As most PDAs are capable of rendering content formatted with HTML, the required delivery channel is very similar to the one used for the kiosk computers. To account for the reduced screen size of these devices, minor adaptations in terms of how information is presented are however necessary.

As shown in Figure 5, both the PDA client as well as the paper client connect to the content management system through a proxy that runs on each client device. Since context

gathering, augmentation and processing have been explicitly excluded from our solution for context-aware data management, this functionality has to be provided by another component of the EdFest system. In the present architecture, this function is fulfilled by a proxy server that appends context information to every request which passes through. For the management of context on this server, a dedicated context database [2] is used. This so-called context engine allows sensor and context types to be defined in a database. It then accesses the physical sensors which are represented as instances of the defined sensor types. The data gathered from the sensors is processed and stored as high-level context information based on the context types specified by the application. Apart from physical context information that is gathered from these physical sensors, the context engine also manages logical context information about the client. For example, it also records which user is currently interacting with the system and what delivery channel is currently being used.

5.1 Realising a Mobile Multi-Channel Information System

In the following, we will discuss how the information services of the EdFest tourist information system have been implemented based on existing technologies as well as the presented solution for context-aware data management. In Figure 6, a detailed architecture of the previously introduced server component is shown. For reasons of space, we have simplified the figure to only include one of the two database servers that are shown in Figure 5. From an implementation point of view both servers share the same architecture and it is, therefore, not necessary to include both in the figure. As can be seen from the figure, the web server component of the content server has been implemented using standard technologies such as Java Server Pages (JSP) and Java Servlets. The database server component is built based on the OMS object-oriented database management system developed at our institute and that has been extended with the presented concepts for context-aware data management.

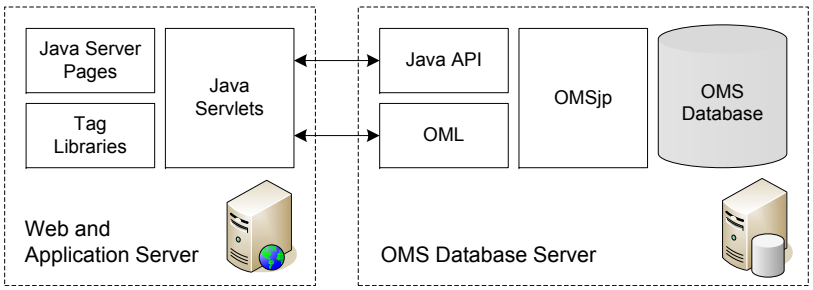


Fig. 6: Server architecture

A sequence diagram detailing the interactions between the different components of the content server is shown in Figure 7. The task of the web server component is to accept incoming requests from the clients of the system and extract the current context information from the request. It then invokes the Java Servlet that handles the information service corresponding to the client's request. The Servlet first consults the metadata server for structure information that determines how to construct a response to the current request. The response is then assembled using content provided by the application data server. In the final step, the response is transformed using presentation information from the metadata server and

sent back to the client. Since both the metadata and application data servers are managed by a database system that supports context-awareness, all aspects of the delivery process can be dynamically adapted to context as all queries issued by the web server component return context-dependent results. More information about the information concepts that are managed by the metadata server and the generation process used to render responses is given in Grossniklaus and Norrie [20].

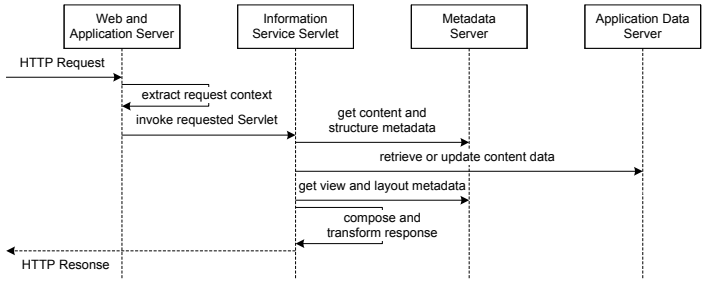


Fig. 7: Interactions between the components of the content server

Applications that need to interface with an OMS database have a choice of two different interfaces. Programmatic access to a database is given through a Java application programming interface that follows the specification of OMSjp [24], a Java library that provides uniform access to heterogeneous OMS database implementations. Within the family of OMS database systems, the OMSjp library plays the same role as Java Database Connectivity (JDBC) does for relational databases. We will see an example of how OMSjp is used to access a database later in this section. As an alternative to OMSjp, an OMS database can be accessed by issuing query statements to the database server. Since OMS databases are object-oriented databases, these queries are expressed using the Object Model Language (OML) [28] rather than SQL. OML is also the language used in OMS databases to implement object methods and database macros and thus we will discuss an OML example later on when presenting the implementation of the reservation functionality at the database level.

5.2 Supporting Context-Aware Patterns of Interaction

As discussed in Section 4, we propose to support context-aware interaction patterns based on multi-variant operations. This solution was inspired by the method dispatching strategies found in object-oriented programming languages. Many object-oriented languages allow methods to be overloaded, i.e. support the definition of multiple versions of the same method with different sets of arguments. At run-time, they select the so-called most specific method from the set of applicable methods, based on the number and type of arguments given by the caller of the method. In its basic nature, virtual method dispatching is not unlike selecting the best matching variant of an object. Figure 8 gives a graphical representation of the versioned object that handles the reservation process. As shown, for each context state that occurs in the process shown in Figure 3(b), an alternative version of the object has been defined. As the context values that will be sent by the client cannot be known beforehand, the context states describing the variants use the value $+*$ which indicates that a value for the corresponding context dimension has to be set but the actual value is not important. Variant `o369@0[0]`

is responsible for starting the reservation process by generating a reservation number and initiating a session on the client. All other variants of the object extract the provided context data, update the application database accordingly and send back a response that guides the visitor to the next step, except for variant `o369@5[5]` that informs the tourists that they have completed the reservation process successfully.

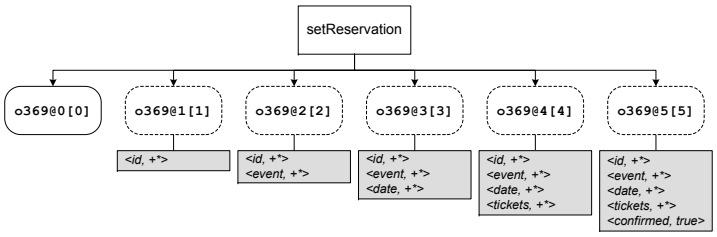


Fig. 8: The setReservation operation object

A one-step reservation process would only need to access the default variant and the variant shown on the far right in the figure. In the case of a multi-step process, however, the reservation process runs through all variants of the objects before completing. In Figure 9, the OML implementation for one of the variants of the database macro that handles the reservation process is shown. As shown in Figure 7, this macro is invoked by the information service Servlet to update the content of the application database. This implementation has been designed for the context state $C_v(S) := \{\langle id, +*\rangle, \langle event, +*\rangle, \langle date, +*\rangle\}$ and thus the code can safely assume that the parameter values for `id`, `event` and `date` have all been set. The reservation object that corresponds to the given identifier is retrieved on line 4. If it does not exist, the implementation has to assume that it was called in a non-linear way and creates a new reservation object with the given identifier in the database on lines 5–9. It then associates the reservation with the given event on line 10–11 and updates the date of the reservation on line 12. As the reservation object is defined to be the output parameter of the macro on line 2, the object is returned to the caller of the macro.

```
1  macro setReservation($id: integer, $event: string, $date: date)
2     ($reservation: reservation)
3  begin
4     $reservation := first(all $r in Reservations having ($r.id = $id));
5     if ($reservation = null) begin
6        $reservation := create object;
7        dress $reservation with reservation (id = $id);
8        insert into Reservations:  [ $reservation ];
9     end;
10    $eventObj := first(all $e in Events having ($e.id = $event));
11    insert into ForEvent:  [ ($reservation, $eventObj) ];
12    $reservation.date := $date;
13 end;
```

Fig. 9: Variant `o369@3[3]` of macro setReservation

As can be seen from the sequence diagram given in Figure 7, the caller of the macro is the web server component, more specifically the Java Servlet that handles the request to process

a reservation step. The Java implementation of the corresponding handler method is given in Figure 10 and uses the OMSjp interface to access our extended object-oriented database system. On lines 2–3, the method obtains a reference to the extended database concept of OMSjp and uses it on lines 4–6 to influence the current context state of the database. Note that the new context state is a combination of the current context state and the macro parameters given to the handler method. The method then executes the setReservation macro on lines 7–8 and retrieves the associated presentation information on lines 9–11. In a final step, the macro result and the presentation information are combined and rendered in XML on line 12 before they are returned to the client. It is important to note that the setting of the context on line 6 has an influence on all subsequent database operations on lines 7–10 as well as on the query executed on line 11.

```
1  public Node setReservation(final Object...  params) {
2     final OMSDatabaseExt db =
3        (OMSDatabaseExt) DatabaseManager.instance().getDatabase();
       // Set the database context
4     final Context context =
5        ContextManager.instance().getCurrentContextState();
6     db.setContext(this.merge(params, context));
       // Retrieve and execute the database macro
7     final OMSMacro macro = db.getMacro("setReservation");
8     final OMSValue[] result = db.executeMacro(marco, params);
       // Retrieve the template for the macro
9     final OMSBinaryCollection hasStyle =
10       ((OMSBinaryCollection) db.getCollection("hasStyle");
11    final Template template = (Template) hasStyle.dr(marcro).range().first();
       // Render and return the result
12    return XMLRenderer.instance().render(result, template);
13 }
```

Fig. 10: OMSjp code to invoke macro setReservation

From both the code of the database macro and the handler method it is apparent that there is no overhead to determine at what step of a process the client currently is. This concern is handled entirely by the context-aware database system using the matching algorithm presented in Section 3. The only additional overhead required by our approach is to set the context before accessing the database. Naturally, this reduced implementation overhead comes at the price of lower performance in terms of how fast requests can be handled. At the moment, the implementation of the matching algorithm is very inefficient in the sense that it needs to compare all variants of an object to the current context state to select the best matching alternative. Thus, it is easily outperformed by an implementation that checks the provided parameters directly to determine what it should do. However, user studies with the EdFest prototype at the Edinburgh Festivals in 2005 have also indicated that the performance achieved by our approach is satisfactory as responsiveness has never been an issue mentioned by the test users.

An interesting aspect of implementing processes in this way is how non-linear navigation processes can be handled. If a user deviates from the intended process by prematurely selecting parameters that will only be gathered in a later step, the value will nevertheless be stored in the client's session but the response will be the same as before, asking the tourist to select

the value corresponding to the current step. When this value is finally selected by the user, all steps that have been executed out of order are skipped automatically as those values have already been stored in the session on the client. Preliminary feedback from test users at the Edinburgh Festivals in 2005 has shown that this rather resilient way of handling the stepping through a process can be a feasible and comprehensible form of interaction. Apart from non-linear navigation processes, value checking and error handling in multi-step processes has been a further motivation for implementing context-aware interaction patterns using multi-variant operations. The logic to check whether all values provided by the user are correct could be implemented on the client-side using a scripting language. However, this solution is not always possible on all required delivery channels, as scripting capabilities, if present at all, vary substantially. Our approach is already able to handle cases where a user has failed to specify a required value. Even if they are not required in situations where all values are correct, in the case of an error, the additional variants defined for the multi-step process can be used for error handling in the single-step process. Although context matching can provide a solution to missing values, it is not capable of addressing the problem of handling errors caused by incorrect data. To also implement this functionality, traditional parsing and error handling techniques have to be applied.

6 Discussion

The ticket reservation process in the EdFest system has been used to show how novel modes of interaction can affect not only the way in which content is accessed and delivered, but also the nature of the interaction process. User studies carried out with the system actually indicated that, within the field of ubiquitous computing, the divergence from traditional linear interaction patterns typical of desktop applications is likely to be much greater than we anticipated. Users not only selected data parameters in different orders, skipping those considered to be irrelevant, but they also wanted to be able to select the data outside of the document areas that we considered as defining the transaction context. For example, to select the date of the reservation, instead of selecting the date in the event entry of the brochure, they would try to specify a date value by selecting a date from anywhere that dates appeared within any of the documents. One can envisage future scenarios where a user could select the date of a reservation by simply pointing to a date in the calendar on their wall.

The multi-step interaction processes have additional interesting characteristics. Looking back at the communication pattern between client and server given in Figure 3(b), we can observe a similarity to modern web applications. In order to prevent page reloads and provide immediate feedback to the user, many web sites nowadays use a technique called Asynchronous JavaScript and XML (AJAX). In AJAX, a web page uses client-side scripting to connect to a server and to transmit values without refreshing the whole page. Web applications based on AJAX communicate with the server at a finer level of granularity that is not unlike multi-step interaction processes. The solution presented here to handle such processes could therefore form the basis for integrating delivery channels that support AJAX with those that do not.

As a result of our efforts to support context-aware interaction processes, we asked ourselves the question of whether it would be sensible to apply the same mechanisms to programs as well as data. We have conducted preliminary research into this direction with the implementation of a prototype language that supports multi-variant programming [38]. The language is an

extension of Prolog that allows predicate implementations to be defined for a given context state. The current context state of the system is managed by library predicates that allow context values to be set and removed. Before a context-aware Prolog program can be executed, it needs to be loaded by a special parser that replaces all predicate calls in the program with a call to a dispatching predicate that takes context into consideration. Experiences gained from a set of example programs have shown that the approach has its merits even though writing context-aware programs can be quite challenging, especially if context-dependent predicates are allowed to modify the context state. Naturally, our prototype implementation suffers from a few limitations and problems such as poor performance. Also, it is still unclear how to combine context-dependent predicate invocation with the backtracking mechanism of Prolog. Nevertheless, we believe that the potential benefits of this approach outweigh these challenges.

7 Conclusions

In this paper we have motivated the need for implementation platforms that allow context-aware applications to be implemented in a flexible and elegant way. In previous work, we have extended a database system with the concept of multi-variant objects that form the basis for context-aware data management and query processing. The most ambitious system implemented based on this extended database system so far is a mobile tourist information system targeted at visitors to the Edinburgh art festivals. Apart from traditional client devices, this EdFest system also supports a mobile paper-based client. In contrast to supporting conventional delivery channels where it is sufficient to adapt the content, structure and presentation, a paper-based interface also requires that the interaction process is adapted dynamically. In order to address this requirement, we have created context-dependent interaction processes. Technically, these interaction processes were realised through different implementation variants of the database macro implementing the corresponding application logic. In this setting, context has been used to dispatch the request made by the client to the desired implementation similar to object-oriented programming languages that dispatch a call to an overloaded method dispatching based on the parameters provided by the caller.

References

1. H. Baumeister, A. Knapp, N. Koch, and G. Zhang. Modelling Adaptivity with Aspects. In *Proceedings of International Conference on Web Engineering, July 27-29, 2005, Sydney, Australia*, pages 406–416, 2005.
2. R. Belotti, C. Decurtins, M. Grossniklaus, M. C. Norrie, and A. Palinginis. Modelling Context for Information Environments. In *Proceedings of International Workshop on Ubiquitous Mobile Information and Collaboration Systems, June 7-8, 2004, Riga, Latvia*, pages 43–56, 2004.
3. R. Belotti, C. Decurtins, M. Grossniklaus, M. C. Norrie, and A. Palinginis. Interplay of Content and Context. *Journal of Web Engineering*, 4(1):57–78, 2005.
4. A. Bozzon, S. Comai, P. Fraternali, and G. Toffetti Carughi. Conceptual Modeling and Code Generation for Rich Internet Applications. In *Proceedings of International Conference on Web Engineering, July 10-14, 2006, Menlo Park, CA, USA*, pages 353–360, 2006.
5. P. Brusilovsky. Methods and Techniques of Adaptive Hypermedia. *User Modeling and User-Adapted Interaction*, 6:87–129, 1996.
6. S. Casteleyn, O. De Troyer, and S. Brockmans. Design time support for adaptive behavior in web sites. In *Proceedings of ACM Symposium on Applied Computing, March 9-12, 2003 Melbourne, FL, USA*, pages 1222–1228, 2003.
7. S. Ceri, F. Daniel, M. Matera, and F. M. Facca. Model-driven Development of Context-Aware

Web Applications. *ACM Transactions on Internet Technology*, 7(2), 2007.
8. S. Ceri, P. Fraternali, A. Bongio, M. Brambilla, S. Comai, and M. Matera. *Designing Data-Intensive Web Applications*. The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann Publishers Inc., 2002.
9. P. De Bra, G.-J. Houben, and H. Wu. AHAM: A Dexter-based Reference Model for Adaptive Hypermedia. In *Proceedings of ACM Conference on Hypertext and Hypermedia: Returning to Our Diverse Roots, February 21-25, 1999, Darmstadt, Germany*, pages 147–156, 1999.
10. O. De Troyer and S. Casteleyn. Designing Localized Web Sites. In *Proceedings of International Conference on Web Information Systems Engineering, November 22-24, 2004, Brisbane, Australia*, pages 547–558, 2004.
11. O. De Troyer and C. J. Leune. WSDM: A User-Centered Design Method for Web Sites. *Computer Networks and ISDN Systems*, 30(1-7):85–94, 1998.
12. R. De Virgilio and R. Torlone. A General Methodology for Context-Aware Data Access. In *Proceedings of ACM International Workshop on Data Engineering for Wireless and Mobile Access, June 12, 2005, Baltimore, MD, USA*, pages 9–15, 2005.
13. R. De Virgilio and R. Torlone. Modeling Heterogeneous Context Information in Adaptive Web Based Applications. In *Proceedings of the International Conference on Web Engineering, July 11-14, 2006, Palo Alto CA, USA*, pages 56–63, 2006.
14. R. De Virgilio, R. Torlone, and G.-J. Houben. A Rule-based Approach to Content Delivery Adaptation in Web Information Systems. In *Proceedings of the International Conference on Mobile Data Management, May 9-13, 2006, Nara, Japan*, pages 21–24, 2006.
15. Z. Fiala, M. Hinz, G.-J. Houben, and F. Frăsincar. Design and Implementation of Component-based Adaptive Web Presentations. In *Proceedings of Symposium on Applied Computing, March 14-17, 2004, Nicosia, Cyprus*, pages 1698–1704, 2004.
16. Z. Fiala, M. Hinz, K. Meissner, and F. Wehner. A Component-based Approach for Adaptive, Dynamic Web Documents. *Journal of Web Engineering*, 2(1-2):58–73, 2003.
17. F. Frăsincar, G.-J. Houben, and P. Barna. Hera Presentation Generator. In *Special Interest Tracks and Posters of International Conference on World Wide Web, May 10-14, 2005, Chiba, Japan*, pages 952–953, 2005.
18. J. Ginzburg, G. Rossi, M. Urbieta, and D. Distante. Transparent Interface Composition in Web Applications. In *Proceedings of International Conference on Web Engineering, July 16-20, 2007, Como, Italy*, pages 152–166, 2007.
19. M. Grossniklaus. *Context-Aware Data Management – An Object-Oriented Version Model*. VDM Verlag, 2007.
20. M. Grossniklaus and M. C. Norrie. Information Concepts for Content Management. In *Proceedings of International Workshop on Data Semantics and Web Information Systems, December 11, 2002, Singapore, Republic of Singapore*, pages 150–159, 2002.
21. M. Grossniklaus and M. C. Norrie. An Object-Oriented Version Model for Context-Aware Data Management. In *Proceedings of International Conference on Web Information Systems Engineering, December 3-6, 2007, Nancy, France*, pages 398–409, 2007.
22. F. Halasz and M. Schwartz. The Dexter Hypertext Reference Model. *Communications of the ACM*, 37(2):30–39, 1994.
23. G.-J. Houben, P. Barna, F. Frăsincar, and R. Vdovják. Hera: Development of Semantic Web Information Systems. In *Proceedings of International Conference on Web Engineering, July 14-18, 2003, Oviedo, Spain*, pages 529–538, 2003.
24. Institute for Information Systems, ETH Zürich. OMSjp – A Uniform Interface to Heterogenous OMS Platforms. `http://www.globis.ethz.ch/research/oms/platforms/omsjp`, 2004.
25. N. Koch. *Software Engineering for Adaptive Hypermedia System*. PhD thesis, Ludwig-Maximilians-University Munich, Munich, Germany, 2000.
26. N. Koch and M. Wirsing. The Munich Reference Model for Adaptive Hypermedia Applications. In *Proceedings of International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, May 29-31, Malaga, Spain*, pages 213–222, 2002.

27. M. Linaje, J. C. Preciado, and F. Sánchez-Figueroa. A Method for Model Based Design of Rich Internet Application Interactive User Interfaces. In *Proceedings of International Conference on Web Engineering, July 16-20, 2007, Como, Italy*, pages 226–241, 2007.
28. A. Lombardoni. *Towards a Universal Information Platform: An Object-Oriented, Multi-User, Information Store*. PhD thesis, Eidgenössische Technische Hochschule, Zurich, Switzerland, 2006.
29. M. C. Norrie. An Extended Entity-Relationship Approach to Data Management in Object-Oriented Systems. In *Proceedings of International Conference on the Entity-Relationship Approach, Arlington, TX, USA*, pages 390–401, 1994.
30. M. C. Norrie, A. Palinginis, and B. Signer. Content Publishing Framework for Interactive Paper Documents. In *Proceedings of ACM Symposium on Document Engineering, November 2-4, 2005, Bristol, United Kingdom*, pages 187–196, 2005.
31. M. C. Norrie, B. Signer, M. Grossniklaus, R. Belotti, C. Decurtins, and N. Weibel. Context-Aware Platform for Mobile Data Management. *Wireless Networks*, 13(6):855–870, 2007.
32. M. C. Norrie, B. Signer, and N. Weibel. General Framework for the Rapid Development of Interactive Paper Applications. In *Proceedings of Workshop on Collaborating over Paper and Digital Documents, November 4, 2006, Banff, Canada*, pages 9–12, 2005.
33. M. C. Norrie, B. Signer, and N. Weibel. Print-n-Link: Weaving the Paper Web. In *Proceedings of the ACM Symposium on Document Engineering, October 10-13, 2006, Amsterdam, The Netherlands*, pages 34–43, 2006.
34. L. Olsina, O. Pastor, G. Rossi, and D. Schwabe. International Workshop on Web-Oriented Software Technologies (IWWOST 2003). `http://www.dsic.upv.es/~west/iwwost03/`, 2003.
35. J. C. Preciado, M. Linaje, F. Sánchez, and S. Comai. Necessity of Methodologies to Model Rich Internet Applications. In *Proceedings of International Symposium on Web Site Evolution, September 26, 2005, Budapest, Hungary*, pages 7–13, 2005.
36. G. Rossi, A. Nieto, L. Mengoni, N. Lofeudo, L. Nuño Silva, and D. Distante. Model-Based Design of Volatile Functionality in Web Applications. In *Proceedings of Latin American Web Congress, October 25-27, 2006, Cholula, Mexico*, 2006.
37. D. Schwabe and G. Rossi. An Object Oriented Approach to Web-based Applications Design. *Theory and Practice of Object Systems*, 4(4):207–225, 1998.
38. B. Schwarzentrub. Multi-Variant Programming. Semester project, Institute for Information Systems, ETH Zurich, 2006.
39. B. Signer. *Fundamental Concepts for Interactive Paper and Cross-Media Information Spaces*. PhD thesis, Eidgenössische Technische Hochschule, Zurich, Switzerland, 2006.
40. B. Signer, M. Grossniklaus, and M. C. Norrie. Interactive Paper as a Mobile Client for a Multi-Channel Web Information System. *World Wide Web Journal*, 10(4):529–556, 2007.
41. B. Signer, M. C. Norrie, M. Grossniklaus, R. Belotti, C. Decurtins, and N. Weibel. Paper-Based Mobile Access to Databases. In *Demonstration Proceedings of ACM SIGMOD International Conference on Management of Data, June 27-29, Chicago, IL, USA*, pages 763–765, 2006.
42. Y. Stavrakas and M. Gergatsoulis. Multidimensional Semistructured Data: Representing Context-Dependent Information on the Web. In *Proceedings of International Conference on Advanced Information Systems Engineering, May 27-31, 2002, Toronto, Canada*, pages 183–199, 2002.
43. Y. Stavrakas, K. Pristouris, A. Efandis, and T. Sellis. Implementing a Query Language for Context-Dependent Semistructured Data. In *Proceedings of East-European Conference on Advances in Databases and Information Systems, September 22-25, 2004, Budapest, Hungary*, pages 173–188, 2004.
44. W. W. Wadge, G. Brown, M. C. Schraefel, and T. Yildirim. Intensional HTML. In *Proceedings of International Workshop on Principles of Digital Document Processing, March 29-30, 1998, Saint Malo, France*, pages 128–139, 1998.
45. A. P. Würgler. *OMS Development Framework: Rapid Prototyping for Object-Oriented Databases*. PhD thesis, Eidgenössische Technische Hochschule, Zurich, Switzerland, 2000.