

Using Object Variants to Support Context-Aware Interactions

Michael Grossniklaus and Moira C. Norrie

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

Abstract We discuss the need to extend general models and systems for context-awareness to include adaptation of interactions to context. Our approach was motivated by our experiences of developing mobile applications based on novel modes of interaction. We describe how we were able to support context-aware interactions using an object-oriented framework that we had already developed to support context-aware web applications.

1 Introduction

Context-awareness in web engineering involves the adaptation of applications to user situations. At the level of models and frameworks to support web engineering, several generic approaches have been proposed to allow application developers to determine what notions of context and adaptation are relevant to specific applications. General models and mechanisms have therefore been developed that can cater for various forms of adaptation that correspond to personalisation, internationalisation, multi-channel access, location-awareness etc. Furthermore, for full generality, it should be possible to adapt any aspect of a web application, including content, structure and presentation.

However, one aspect that has received relatively little attention is the need to adapt interaction processes to context and how existing models and mechanisms can be generalised to support this. Our experiences have shown that supporting mobile and ubiquitous applications often involves working with new modes of interaction resulting from the characteristics of the different devices used. The nature of these devices is such that the linearity of traditional web-based transactions may be lost and input data may be assembled from various sources and in different orders rather than being specified in a single step. This also means that users need to be carefully guided through the interaction so that they are aware of the current interaction state. An important factor here is that users can be supplied with context-dependent help information according to the interaction state.

In this paper, we describe how we were able to exploit an object-oriented framework that was developed to support context-aware web engineering to support context-aware interactions. We begin in Sect. 2 with a discussion of related

work and a motivation of our approach. Sect. 3 presents the main features of the object-oriented framework and how it supports context-awareness through a notion of multi-variant objects. In Sect. 4, we show how the mechanisms used to supported context-awareness in our framework could be used to support context-aware interactions. Section 5 provides a general discussion of the approach and directions of future work. Concluding remarks are given in Sect. 6.

2 Background

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 implementation platforms recently proposed. For example, the Web Modelling Language (WebML) [1] has been extended with primitives that allow adaptive and context-aware sites to be modelled [2]. To manage context information, the data model is extended with a context model that is specific to the developed application. To gather context information, two additional units—**Get URL Parameter** and **Get Data**—have been introduced. The first unit retrieves context information sent to the server by the client device encoded in the URL. The second unit extracts context information according to the context model from the database on the server. Each page that is considered to be context-dependent is associated in the model with a context cloud that contains the adaptation operation chains. These operation chains can be built from the standard WebML operation units as well as from units that have been introduced to model conditional or switch statements in the specification of workflows. When a context-aware page is requested, the corresponding operation chain is executed and the content of the page 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 notion of context. Apart from such content adaptation, it is also possible to adapt the navigation and the presentation. The newly introduced **Change Site View** unit can be used to forward a client from one site view to another, whereas the **Change Style** unit adapts the web site in terms of colours and font properties. Another extensions to WebML allows reactive web applications [3] to be specified. The proposed approach uses the Web Behaviour Model (WBM) in combination with WebML to form a high-level Event-Condition-Action (ECA) paradigm. WBM uses the notion of timed finite state automata to specify scripts that track the users' navigation on the web site. When a WBM script reaches an accepting state, the condition it represents is fulfilled and the corresponding actions in the form of a WebML operation chain are executed as soon as the associated event occurs. Based on this graphical ECA paradigm, applications such as profiling to infer a user's interests or updating specific values within the user model as well as adapting to this information can be specified and implemented automatically based on an intuitive model.

The Hera methodology [4] is a model-driven approach that integrates concepts from adaptive hypermedia systems with technologies from the semantic web. Faithful to its background of adaptive hypermedia systems, the specification of adaptation has always been an integral part of the Hera methodology [5]. Hera distinguishes between static design-time adaptation called adaptability and dynamic run-time adaptation called adaptivity. The design artefacts of all three models used in the development process can be adapted by annotating them with appearance conditions. Depending on whether the condition specifies an adaptability or adaptivity rule, they are evaluated during the generation step or at run-time. If a condition evaluates to true, the corresponding artefact will be presented to the user, otherwise it is omitted. Thus, alternatives can be specified using a set of mutually exclusive appearance conditions. Similar to the approach taken by WebML, web sites that have been designed with Hera are implemented by using the conceptual models to configure a run-time environment. The Hera Presentation Generator (HPG) [6] is an example of such a platform that combines the data stored as RDF with the models represented in RDFS to generate an adapted presentation according to user preferences as well as device capabilities. The presentation compiled by the Hera presentation generator is rendered as a set of static documents that contain the mark-up and the content for one particular class of clients. Hence, with this approach, it is only possible to implement appearance conditions that express design-time adaptability. More recently, an alternative implementation platform for Hera has been proposed based on the AMACONT [7] project. Based on a layered component-based XML document format [8], reusable elements of a web site can be defined at different levels of granularity. The document components that encapsulate adaptive content, navigation and presentation are then composed through aggregation and interlinkage into adaptive web applications. The proposed document format has three abstraction levels—media components, content unit components and document components—mirroring the iterative development process of most web sites. 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. The decision as to whether a component is presented to the user is made by the XSLT stylesheet that generates the presentation according to the current context. 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. Through the caching of partial results, intermediate steps can be reused multiple times leading to improved performance.

In UML-based Web Engineering (UWE) [9], adaptation is based on the Munich Reference Model [10] for adaptive hypermedia applications. The architecture and concepts of this reference model are based entirely on the previously discussed Dexter and AHAM reference models. However, while Dexter has been specified in Z without a graphical representation and AHAM has so far only been defined informally, the Munich Reference Model being written in UML offers both a graphical representation and a formal specification using the Ob-

ject Constraint Language (OCL). The model uses the same layering as Dexter—within-component, storage and run-time layers—and partitions the storage layer in the same way as AHAM into a domain, user and adaptation model. In contrast to the existing models, the Munich Reference Model distinguishes three forms of rule-based adaptation instead of two. To match the three layers of the UWE methodology, these forms of adaptation are 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 aspects-oriented modelling techniques [11]. As adaptation logic is orthogonal to the basic application logic, the cross-cutting nature of aspects provides a promising solution for separating the two. By introducing the concept of aspects, the UWE metamodel has been extended to support adaptive navigation capabilities such as adaptive link hiding, adaptive link annotation and adaptive link generation.

So far, we have looked at the most influential conceptual models for web engineering and in some cases their proprietary implementation platform. Apart from those, 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)* [12]. 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)* [13] which in turn provided the foundation for *Multidimensional Semistructured Data (MSSD)* [14]. Similar to semi-structured data that is often modelled using the Object Exchange Model (OEM), MSSD is represented in terms of a graph model that extends OEM with multidimensional nodes and context edges. In the resulting Multidimensional Object Exchange Model (MOEM), 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. In contrast to the conventional edges used in OEM, the label of a context edge specifies in which context the variant pointed to is appropriate. Using these specifiers, a MOEM graph can be reduced to a corresponding OEM graph for a given context. Based on this graph representation, a *Multidimensional Query Language (MQL)* [15] 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.

A general and extensible architecture that supports context-aware data access is proposed in [16]. Their approach is based on the concepts of profiles and configurations. Context is represented as a collection of profiles that each specify one aspect of the context such as the user, the device etc. Each profile contains a set of dimensions that capture certain characteristics and are associated to context values over attributes. Profiles are expressed according to the General Profile Model (GPM) [17] 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 [18]. While such profiles describe the context in which a request has been issued to the web information system, configurations express how the response should be generated. A configuration has three parts that match the general architecture of web information systems in terms of content, structure and presentation. The content part of the configuration is represented by a query formulated in relational algebra. The definition of the structure part is expressed using WebML to define the hypertext. Finally, the presentation part is specified using the notion of a logical stylesheet which unifies languages such as Cascading Stylesheets (CSS). 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 done based on adaptation rules consisting of a parametrised profile, a condition and a parametrised configuration [19]. The profile allows parameters instead of values to be used that are then assigned the values specified in the client profile. The condition constrains the values that are valid for the rule to be applied by formulating a logical expression over the parameters. Finally, the configuration includes the parameters value to adapt the content delivery. During the matching process, the client profile is compared to the 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.

3 Multi-Variant Objects

As presented in the previous section, most model-based approaches offer at least some support for context-aware web engineering. Some solutions even offer 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. Therefore, this functionality has often to be implemented over and over again leading to poor reuse of code and maintainability. In this section, we will present multi-variant objects as an enabling concept for context-aware query processing in information systems.

Multi-variant objects have 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 [20] 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. This model was chosen as, due to its generality, it is possible to use it to implement other conceptual models such as the Entity-Relationship (ER) model or the Resource Description Framework (RDF). Further, the feature of multiple instantiation, i.e. the ability of a single object to have multiple instances that exist on different paths along the inheritance graph, is something that is of frequent use in the domain of web

engineering. Imagine, for instance, a content management system that manages users who have certain roles in the administration of the web site. Based on these user roles, the types of the objects themselves will vary as they include different attributes and methods. In most object-oriented systems, this is usually modelled by defining the abstract concept of a user and then using inheritance to define concrete subtypes of this user. Most of these systems however do not provide a solution for the requirement that a user object needs to have two or more of these subtypes at the same time, whereas in reality users can have any number of roles, as someone can be, for example, both a site administrator and a content editor. In OM, the feature of multiple instantiation can be used to cater for exactly this kind of situation.

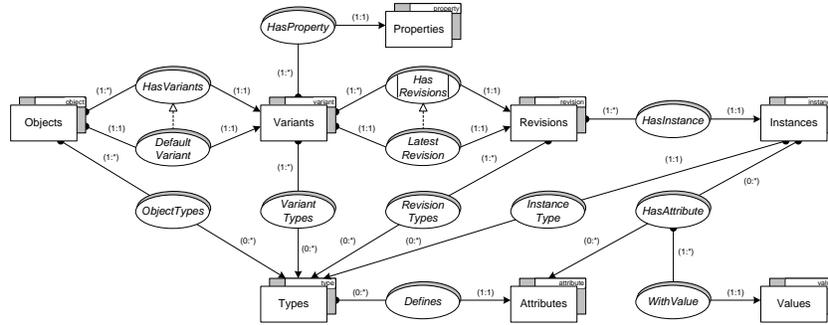


Figure 1: Conceptual data model of an object

Therefore, in the original OM model, 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 the type of which they are 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 Fig. 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 both objects and variants can be related to any number of types. A variant of an object is identified by the set of properties associated with it. Any variant can have an

arbitrary number of properties, each consisting of a name and a value. Finally, instances are still identified based on their type. Hence they can only be linked to exactly one of the types to which the object is related. Further, instances are associated with values and thus contain the actual data 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. Since several frameworks for context gathering, management and augmentation already exist, our intention was to provide a representation that is as general as possible. Acknowledging the fact that each application has its own understanding of context, this representation is based on the concept of a *context space* S that defines the names of the context dimensions that occur in an application. Each context dimension *name* can be associated with a *value* to form a *context value* $c = \langle name, value \rangle$. Then, a *context* $C(S)$ is a set of context values for the dimensions specified by S . Finally, a *context space* denoted by $C_*(S)$ is a special context that contains exactly one value for every context dimension of 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.

Context-aware queries over these multi-variant objects are evaluated using the matching algorithm shown in Fig. 2 to select the appropriate variants whenever objects are accessed by the query processor. The algorithm takes an object o and the current context state of the system $C_*(S)$ as inputs. First it retrieves all variants of o that are linked to it through the *HasVariants* association. After building the context state of each variant from the properties that are 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 that has obtained the highest score s_{max} . However, 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.

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. As it is not possible to discuss all issues involved in designing such a scoring function in the scope of this paper, we refrain from going into

```

MATCH( $o, C_*(S)$ )
1  $V_0 \leftarrow rng(HasVariants\ dr(\{o\}))$ 
2  $V_1 \leftarrow V_0 \propto (x \rightarrow (x \times rng(HasProperty\ dr(\{x\}))))$ 
3  $V_2 \leftarrow V_1 \propto (x \rightarrow (dom(x) \times f_s(C_*(S), rng(x))))$ 
4  $s_{max} \leftarrow max(rng(V_2))$ 
5  $V_3 \leftarrow V_2 \% (x \rightarrow rng(x) = s_{max})$ 
6 if  $|V_3| = 1 \wedge s_{max} \geq s_{min}$ 
7   then  $v \leftarrow V_3\ nth\ 1$ 
8   else  $v \leftarrow rng(DefaultVariant\ dr(\{o\}))\ nth\ 1$ 
9 return  $v$ 

```

Figure 2: Matching algorithm

further detail. Nevertheless, we will give an intuitive understanding of its effect by means of examples in the next section.

4 Context-Aware Interactions

Based on the concept of multi-variant objects, we implemented a context-aware content management system that was used as the server component of a mobile tourist information system. 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. A coarse overview of the architecture of the so-called EdFest system [21,22] is shown in Fig. 3. The range of clients that are supported by our system is shown on the left-hand side of the figure. Apart from traditional clients that are based on desktop PCs and PDAs, EdFest introduced a novel interaction channel based on interactive paper [23]. Our context-aware content management system is shown on the right-hand side of the figure. It consists of a web server that handles the communication with the clients, a server that manages publishing metadata [24,25] and an application database that stores the content of the EdFest application database. While the kiosk and PDA clients are implemented using standard HTML, the paper client actually consists of two publishing channels. The paper publisher [26] channel is used at design time to author and print the interactive documents from the content managed by the application database. The paper client channel is then active at run-time when the system is used by the tourists and is responsible for delivering additional information about festival venues and events by using voice feedback when the users interact with the digital pen on the interactive documents.

Of the four publishing channels, the paper and PDA client are mobile and have thus been integrated with the platform shown at the centre of the figure that manages various aspects of context. A range of sensors gather location, weather and network availability information that is then managed in a dedicated context database [27]. Context information is sent from the client to the server by encoding it in the requests sent to the content management server. This is one of the tasks of the client controller component. It acts as a transparent

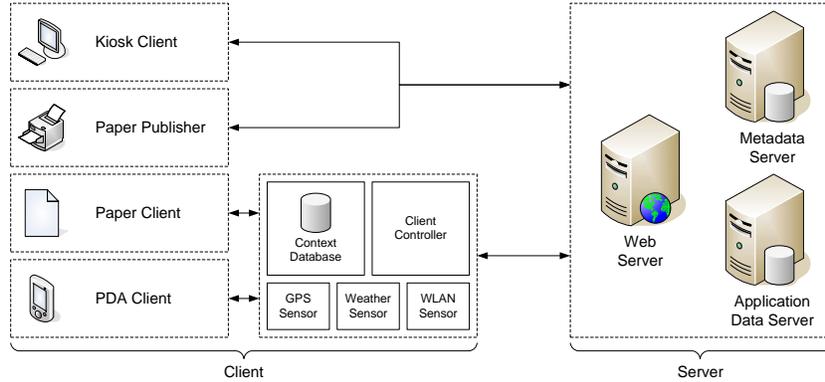


Figure 3: Overview of the EdFest architecture

proxy that intercepts requests and appends the current context state stored in the context database. Another task of this component is to act as a server on the client side, enabling the server to issue call-back requests to the client and thus allowing proactive behaviour to be implemented.

In this paper, we do not go into further details of the functionality offered by the EdFest system. A comprehensive description of the design and implementation of the system can be found in [28]. We instead focus on one particular functionality that demonstrates the need for context-aware interactions. The functionality we have chosen is the reservation process that allows tickets to be booked interactively. To understand what context-aware interactions are, Fig. 4 compares the interaction process of the prototype kiosk interface to the process on the paper client. At the top, Fig. 4(a) and (b) show the two different graphical user interfaces. The kiosk interface offers an HTML form with text fields in which the required information can be entered by the user. If all information has been entered, the data is sent to the server by clicking the submit button. With the paper client, the reservation process is quite different. Instead of entering all information and submitting the form with all data at once, a request is sent to the server for each parameter. The reason for this behaviour is that, in contrast to the web-based user interface, the tourist needs to be guided through the process by constant voice feedback. Also, this feedback serves as a confirmation of the data entered that could not be perceived by the user otherwise. Therefore, to book a ticket with the paper client, the tourist has to first start the reservation process by pointing to the icon labelled “Start reservation”. The system then instructs them to select the event for which they want to book tickets. This is done by selecting the event with the digital pen in a separate brochure listing all events. After an event has been selected, the number of tickets and the date are set in much the same way. The server checks if the booking is valid and, if so, sends a voice prompt to the client instructing the tourist to confirm the reservation by clicking on the icon labelled “reserve”. At the bottom, Fig. 4(c)

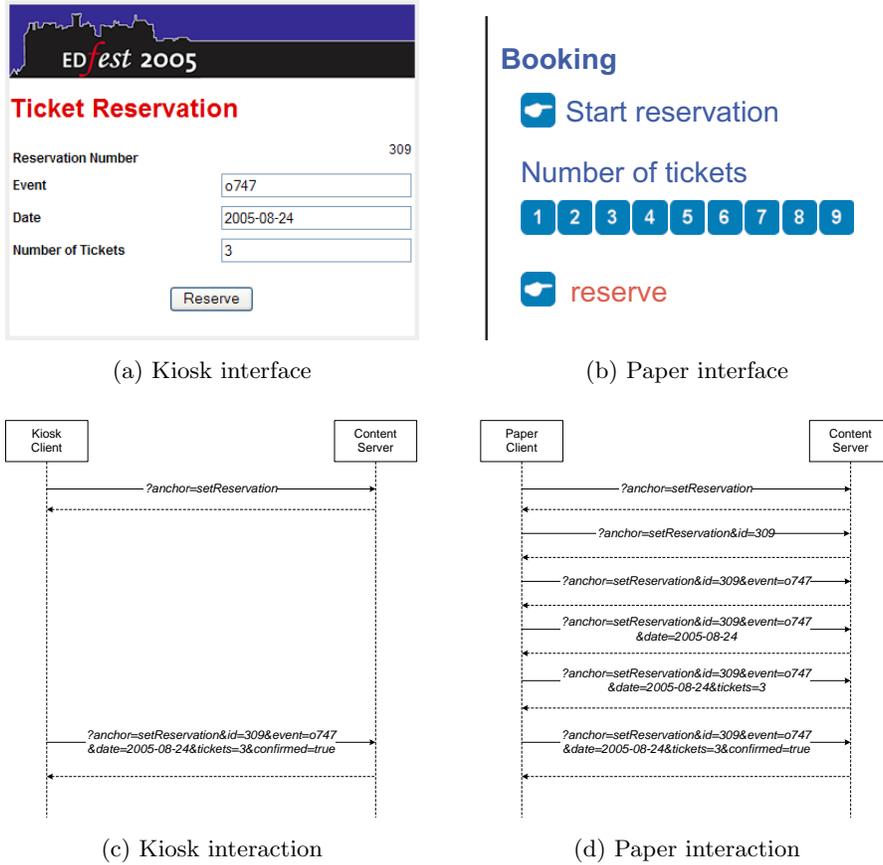


Figure 4: Comparison of the reservation process on different clients

and (d) illustrate the communication pattern that results from reserving tickets using the kiosk and paper client, respectively. As can be seen in the figure, accessing the reservation process from the kiosk client results in two request and response pairs where the first retrieves the empty form and the second uploads all values to the server for processing. The picture in the case of the paper client is quite different as each data value required to process the reservation request is sent to the server encoded in an individual request. Additionally, the already selected values have to be managed in a session on the client and retransmitted with every request.

Implementing the server-side application logic that handles the reservation process across multiple channels is a difficult task if the interaction patterns of the different channels are as heterogeneous as in the given example. In the EdFest system, our 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. All that has to be done to simulate method dispatching based on multi-variant objects is to define an object type that represents operations and treat the parameters specified by the client as context values.

Figure 5 gives a graphical representation of the multi-variant method object that was created to handle the `setReservation` process. As shown, for each context state that occurs in the process shown in Figure 4(d), a variant of the object is 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. The default variant is responsible for starting the reservation process by generating a reservation number and initiates 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.

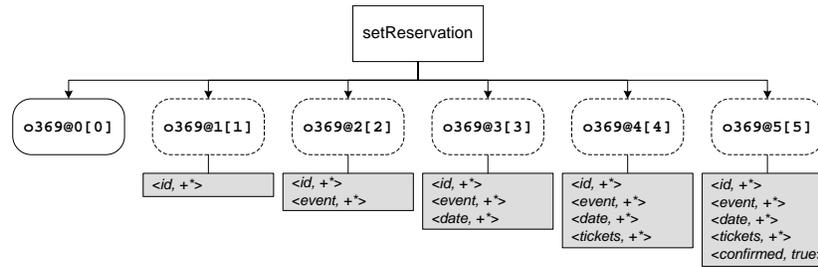


Figure 5: The `setReservation` object

The kiosk reservation process only needs to access the default variant and the variant shown on the far right in the figure. In the case of the paper client, however, the reservation process runs through all variants of the objects before completing. An interesting aspect of implementing such processes is the way in which errors made by the user are handled. Interacting with the paper client, it is impossible to cause an error by entering incorrect values into the reservation process as all data is chosen from the pre-authored paper documents. The tourist can, however, deviate from the process by prematurely selecting parameters that will only be gathered in a later step. In this case, 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.

While a tourist cannot deviate from the defined process in the web interface, it is possible to enter arbitrary values as well as to leave out certain parameters altogether. Hence, the system has to be able to additionally cope with these errors. The logic to check whether the form has been completed correctly by the user could be implemented on the client-side using embedded scripts. However, this solution is not generally possible on all required delivery channels as scripting capabilities, if present at all, vary substantially. Our approach to implementing process functionality based on an object with multiple variants is already able to handle cases where the tourist has failed to specify a required value. Even if they are not required in situations where the tourist fills in the form correctly, in the case of an error, the additional variants defined for the interactive paper process can be used for error handling in the kiosk interface. An omitted parameter will lead to the selection of one of these intermediate variants which will be rendered for the client as a form where the missing parameter is highlighted. 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 technique have to be applied.

5 Discussion

Using the ticket reservation process available in the EdFest system as an example, we have argued that interactive paper not only affects the way in which content is accessed and delivered but also the nature of information interaction. In EdFest, this problem was solved by creating context-aware operations that were realised based on multi-variant objects. Apart from the aspects already discussed, the interaction processes implemented for the interactive paper client have additional interesting characteristics. Looking back at the communication pattern between client and server given in Figure 4(d), a similarity to modern web applications can be observed. 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). As indicated by its name, AJAX is a combination of existing technologies that are used together to provide more interactive web pages. In AJAX, a web page uses client-side scripting to connect to a server and to transmit values without reloading the whole page. At the time of opening the connection, a response handler is registered that will be invoked as soon as the request has been processed. Using JavaScript, the response handler can then update the web page asynchronously by accessing the Document Object Model (DOM) of the mark-up used to render the current page. Web applications based on AJAX communicate with the server at a finer level of granularity that is not unlike the interaction processes encountered on the paper client. 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.

The use of context in this implementation raises an interesting question. We must ask ourselves whether it is sensible to apply the same mechanisms not only to data but also to programs. We have conducted preliminary research into this direction with the implementation of a prototype language that supports multi-variant programming [29]. 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 and will therefore continue to investigate the application of our version model to programming languages.

6 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. Our approach proposes to extend information systems with the concept of multi-variant objects that form the basis for context-aware query processing. Based on this concept, we have implemented a context-aware content management system that has since been used to implement several web-based systems. The most ambitious system implemented 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. As an example, we have discussed the implementation of the reservation process based on the EdFest interactive paper documents. In order to address the situation that the paper client requires a different communication pattern than traditional browser-based clients, 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. 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. (2002)
2. Ceri, S., Daniel, F., Matera, M., Facca, F.M.: Model-driven Development of Context-Aware Web Applications. *ACM Transactions on Internet Technology* **7**(2) (2007)
3. Ceri, S., Daniel, F., Facca, F.M.: Modeling Web Applications Reacting to User Behaviors. *Computer Networks* **50**(10) (2006) 1533–1546
4. Houben, G.J., Barna, P., Fräsinc̆ar, F., Vdovjak, R.: Hera: Development of Semantic Web Information Systems. In: Proceedings of International Conference on Web Engineering, July 14-18, 2003, Oviedo, Spain. (2003) 529–538
5. Barna, P., Houben, G.J., Fräsinc̆ar, F.: Specification of Adaptive Behavior Using a General-Purpose Design Methodology for Dynamic Web Applications. In: Proceedings of Adaptive Hypermedia and Adaptive Web-Based Systems, August 24-26, 2004, Eindhoven, The Netherlands. (2004) 283–286
6. Fräsinc̆ar, F., Houben, G.J., Barna, P.: Hera Presentation Generator. In: Special Interest Tracks and Posters of International Conference on World Wide Web, May 10-14, 2005, Chiba, Japan. (2005) 952–953
7. Fiala, Z., Hinz, M., Houben, G.J., Fräsinc̆ar, F.: Design and Implementation of Component-based Adaptive Web Presentations. In: Proceedings of Symposium on Applied Computing, March 14-17, 2004, Nicosia, Cyprus. (2004) 1698–1704
8. Fiala, Z., Hinz, M., Meissner, K., Wehner, F.: A Component-based Approach for Adaptive, Dynamic Web Documents. *Journal of Web Engineering* **2**(1-2) (2003) 58–73
9. Koch, N.: Software Engineering for Adaptive Hypermedia System. PhD thesis, Ludwig-Maximilians-University Munich, Munich, Germany (2000)
10. Koch, N., Wirsing, M.: 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. (2002) 213–222
11. Baumeister, H., Knapp, A., Koch, N., Zhang, G.: Modelling Adaptivity with Aspects. In: Proceedings of International Conference on Web Engineering, July 27-29, 2005, Sydney, Australia. (2005) 406–416
12. Wadge, W.W., Brown, G., Schraefel, M.C., Yildirim, T.: Intensional HTML. In: Proceedings of International Workshop on Principles of Digital Document Processing, March 29-30, 1998, Saint Malo, France. (1998) 128–139
13. Stavrakas, Y., Gergatsoulis, M., Rondogiannis, P.: Multidimensional XML. In: Proceedings of International Workshop on Distributed Communities on the Web, June 19-21, 2000, Quebec City, Canada. (2000) 100–109
14. Stavrakas, Y., Gergatsoulis, M.: 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. (2002) 183–199

15. Stavrakas, Y., Pristouris, K., Efandis, A., Sellis, T.: 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. (2004) 173–188
16. De Virgilio, R., Torlone, R.: 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. (2005) 9–15
17. De Virgilio, R., Torlone, R.: Management of Heterogeneous Profiles in Context-Aware Adaptive Information System. In: Proceedings of On the Move to Meaningful Internet Systems Workshops, October 31–November 4, 2005, Agia Napa, Cyprus. (2005) 132–141
18. De Virgilio, R., Torlone, R.: 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. (2006) 56–63
19. De Virgilio, R., Torlone, R., Houben, G.J.: 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. (2006) 21–24
20. Norrie, M.C.: 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. (1994) 390–401
21. Belotti, R., Decurtins, C., Norrie, M.C., Signer, B., Vukelja, L.: Experimental Platform for Mobile Information Systems. In: Proceedings of International Conference on Mobile Computing and Networking, August 28–September 2, 2005, Cologne, Germany. (2005) 258–269
22. Signer, B., Norrie, M.C., Grossniklaus, M., Belotti, R., Decurtins, C., Weibel, N.: Paper-Based Mobile Access to Databases. In: Demonstration Proceedings of ACM SIGMOD International Conference on Management of Data, June 27-29, Chicago, IL, USA. (2006) 763–765
23. Signer, B.: Fundamental Concepts for Interactive Paper and Cross-Media Information Spaces. PhD thesis, Eidgenössische Technische Hochschule, Zurich, Switzerland (2006)
24. Grossniklaus, M., Norrie, M.C.: Information Concepts for Content Management. In: Proceedings of International Workshop on Data Semantics and Web Information Systems, December 11, 2002, Singapore, Republic of Singapore. (2002) 150–159
25. Belotti, R., Decurtins, C., Grossniklaus, M., Norrie, M.C., Palinginis, A.: Interplay of Content and Context. *Journal of Web Engineering* 4(1) (2005) 57–78
26. Norrie, M.C., Signer, B., Weibel, N.: Print-n-Link: Weaving the Paper Web. In: Proceedings of the ACM Symposium on Document Engineering, October 10-13, 2006, Amsterdam, The Netherlands. (2006) 34–43
27. Belotti, R., Decurtins, C., Grossniklaus, M., Norrie, M.C., Palinginis, A.: Modelling Context for Information Environments. In: Proceedings of International Workshop on Ubiquitous Mobile Information and Collaboration Systems, June 7-8, 2004, Riga, Latvia. (2004) 43–56
28. Signer, B., Grossniklaus, M., Norrie, M.C.: Interactive Paper as a Mobile Client for a Multi-Channel Web Information System. To appear in *World Wide Web Journal* (2007)
29. Schwarzentrub, B.: Multi-Variant Programming. Semester project, Institute for Information Systems, ETH Zurich (2006)