Model-Driven Dynamic Generation of Context-Adaptive Web User Interfaces

Steffen Lohmann, J. Wolfgang Kaltz, and Jürgen Ziegler

University of Duisburg-Essen,
Lotharstrasse 65, 47057 Duisburg, Germany,
{lohmann, kaltz, ziegler}@interactivesystems.info

Abstract. The systematic development of user interfaces that enhance interaction quality by adapting to the context of use is a desirable, but also highly challenging task. This paper examines to which extent contextual knowledge can be systematically incorporated in the model-driven dynamic generation of Web user interfaces that provide interaction for operational features. Three parts of the generation process are distinguished: selection, parameterization, and presentation. A semantically enriched service-oriented approach is presented that is based on the CATWALK framework for model interpretation and generation of adaptive, context-aware Web applications. Automation possibilities are addressed and an exemplary case study is presented.

Keywords. Context-aware Web User Interfaces, Web Service Integration, Ontology-based Modeling, Model Interpretation, Model-Driven User Interface Generation, Parameterization, Semantically Enriched SOA.

1 Introduction

The systematic development of complex applications requires a significant effort in modeling throughout the whole life cycle. A promising approach is to use these models not only as design basis for subsequent manual implementation or for semiautomatic generation of application code, but rather consider these models as an inherent part of the system. Changes in the models are then directly visible in the application (or in a prototype used for testing). We developed CATWALK, a Web application framework that follows this design paradigm by interpreting ontology-based models at run-time for dynamic generation of adaptive, context-aware Web applications (cp. [6]).

Building upon this framework, we investigate in this paper how Web user interfaces for operational features can be dynamically selected, generated, and adapted according to the context of use with the motivation to enhance user interaction and reach better usability. By operational features, we mean interactive application functionality that goes beyond hypertext navigation (cp. [1]). By context, we understand the generic meaning of the term, including various aspects such as the user's profile, current task and goal, the location, time, and device used. In [7], we give a formal definition of context for Web scenarios.

T. Kühne (Ed.): Models in Software Engineering, LNCS 4364, pp. 116–125, 2007.

http://www.springeronline.com/3-540-69488-9

First, we provide some background information by discussing work related to the modeling and generation of adaptive Web applications and by giving an overview of the CATWALK architecture and the underlying ontology-based modeling method.

2 Related Work

Several existing approaches that address the systematic development of Web applications (Web Engineering) use conceptual models to describe the application's domain. Further aspects such as the application's navigational structure or presentation issues are defined on the basis of these conceptual models. Additional modeling is required for the definition of adaptive system behavior.

The *UML-based Web Engineering* (UWE) approach [9] explicitly addresses adaptivity issues in Web Engineering by providing extra user and adaptation models. UML is used for modeling; the models are stored in XMI. The development framework *Apache Cocoon* has been extended for the generation of application code from the UWE models [10]. However, user and adaptation models are not considered thus far by the code generation framework and the generated Java classes and XSLT stylesheets cannot be executed directly, but need to be manually completed first. Furthermore, UWE addresses primarily the modeling and adaptation of content, navigation and presentation; the integration of operational features and the generation of corresponding user interfaces are not covered by UWE.

The *XML-based Web Modeling Language* (WebML) [4] supports the integration of operational features via Web Services in modeling and application generation [11], but it is not discussed in detail how user interfaces for these features are generated. Further, possibilities for the consideration of context in WebML have been proposed [3], but not in conjunction with the modeling and generation of user interfaces for operational features.

The model-driven generation of user interfaces is also a major research topic in the Human-Computer Interaction (HCI) community. The development of so-called *Multiple* or *Plastic User Interfaces* gains growing interest in the last couple of years (for an overview see e.g. [13]). The focus is on the transformation from abstract platform independent descriptions to concrete user interfaces for various platforms. However, further contextual influences on the different levels of the generation process are rarely addressed in these approaches.

Generally speaking, existing user interface engineering approaches do not consider contextual influences in their modeling and application generation processes to a full degree. They typically consider either information about user preferences or about the location (see [8] for a survey) or address the model-driven generation of multiple-platform user interfaces. Web Engineering approaches that address adaptivity are primarily concerned with issues of how the application's navigation or contents can be adapted. The generation of adaptive, context-aware Web user interfaces for the interaction with operational features is not covered by existing approaches.

3 Ontology-Based Web Application and Context Modeling

Our approach is rooted within the WISE research project [14], where ontologies are used for conceptual Web application modeling. Ontology-based software engineering allows for advanced semantic expressive power in modeling and model exchange compared to other modeling techniques (cp. [5]). Especially for the interoperable integration of contextual knowledge, ontology-based modeling appears promising. The model base of our approach is a repository consisting of the following models (see also Figure 1):

- A *domain ontology*, defining concepts, instances, and relations of the application's domain as well as referencing resources used by the application.
- Several *context ontologies*, defining concepts, instances, and relations of the context of use which are relevant for adaptive system behavior.
- A *context relations model*, defining contextual influences, e.g. by means of weighted relations between entries of the domain ontology and entries of the context ontologies.
- A *navigation*, a *view*, and a *presentation model*, each containing *adaptation specifications* that define rules for adaptive system behavior based on the ontology entries and the defined context relations.

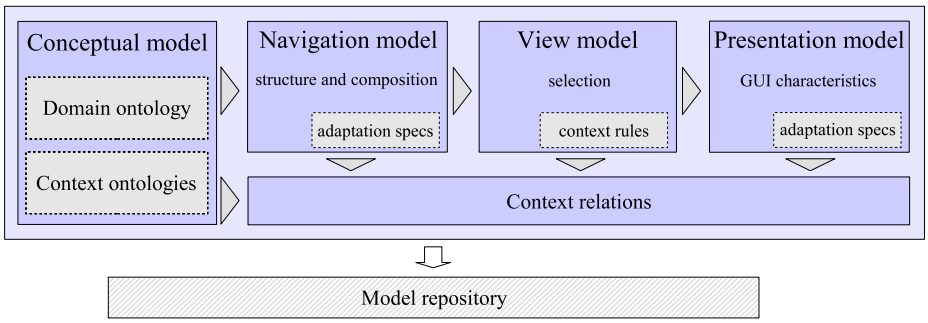

Fig. 1. Application and context modeling in the WISE methodology

4 The CATWALK Framework

Catwalk [6] is a component-oriented framework that interprets the models at run-time to generate an adaptive, context-aware Web application. It is based on Apache Cocoon; Figure 2 gives an architectural overview. The components of Catwalk can be assigned to one of two categories. The first category consists of components that provide core functionality for context-awareness and

reasoning. The second category consists of components that are responsible for adaptive Web application generation. White arrows indicate the process flow: each client request is matched in the *Cocoon pipeline* and processed through a series of components responsible for application generation, ultimately resulting in a response to the client (e.g. a Web page). Arrows with dotted lines indicate calls between components. Each component implements a specific concern, in the sense of the *separation of concerns* architectural design principle. A component is implemented by one or more Java classes and may use additional artefacts (such as XSLT stylesheets for XML transformation). The model repository is accessed via a Cocoon pseudo-protocol in each generation step and the corresponding model parts are interpreted at run-time. A central component (the *Adaptation Engine*) coordinates adaptive system behavior by interpreting context relations and adaptation specifications and considering the respective contextual state (provided by the *Context Manager* component).

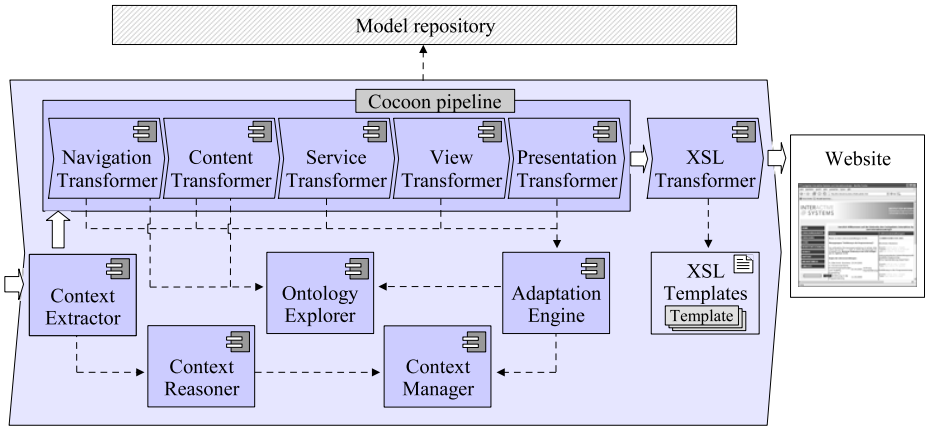

Fig. 2. Component architecture of the CATWALK framework

5 Generation of Adaptive, Context-Aware User Interfaces

Contextual knowledge affects different levels of the user interface generation process. The following questions can be addressed: In what situations should a user interface (or a part of it) be generated (selection)? Which values can be preselected (parameterization)? What should the user interface look like (presentation)?

With these questions in mind, we now take a closer look at our approach to incorporate contextual knowledge in the different steps of the user interface

generation process. This is accompanied by an example scenario for better illustration: a Web portal for automobile services that provides a car rental functionality. Beforehand, we shortly address the representation of operational features in our approach.

5.1 Representation

CATWALK follows a service-oriented approach – operational features that are offered by the Web application are encapsulated in Web Services. Representing operational features is therefore primarily a Web Service composition and coordination problem. Two dimensions can be distinguished: one, defining how to combine discrete Web Services to more complex functionalities, and another, defining the order in which Web Services are executed. The main challenge is the definition of mappings and transformations between the different Web Services' input and output values and the consideration of preconditions and effects. The required modeling effort depends on the degree of semantic description that is provided by the Web Services' interfaces. The aim of the *Semantic Web Services* approach is to provide best possible semantic descriptions of Web Services to support the (partial) automation of Web Service discovery, composition, execution, and interoperation (see e.g. OWL-S [12]). If solely syntactic descriptions of Web Services are given (such as is the case for WSDL), knowledge about the capabilities of the involved Web Services must be part of the application models.

Context-aware Web Service composition and coordination models are not in the focus of this paper and shall not be further discussed here (for details, see e.g. [2]). For the remainder of this paper, we make the following generalized assumption: Each operational feature is implemented by n Web Services ($n = 1$ is possible). These Web Services are referenced in the domain ontology together with necessary information about their structure and interrelations.

Consider for example a car rental feature consisting of four parts, which are realized by three Web Services – the first implements the selection of the desired vehicle type(1) and the car model and equipment details(2), the second the booking(3) and the third the payment(4). An ontology entry is created for each part of the operational feature referencing the corresponding WSDL description. The user shall interact with these Web Services in sequence – the ontology entries are interconnected by appropriate relations and assigned to a master concept that represents the whole feature. Additionally, parameters of the Web Services are mapped (see section 5.3).

Alternatively, to briefly mention automation and extension possibilities for Semantic Web Service scenarios, solely a formal semantic description of the desired operational feature would be defined instead of explicit references to Web Services. Then, the challenge is the automated discovery and composition of Web Services that realize the desired operational feature.

5.2 Selection

The first step in the user interface generation process consists in the dynamic selection of those operational features for which user interfaces should be presented in the current Web page.

The navigation model defines the navigational structure of the Web application by means of relations between entries of the domain ontology. The navigational structure is mapped onto the user's current navigational position to identify application items, including operational features, that should potentially be offered in the current Web page. Furthermore, context relations between operational features and concepts of the context ontologies are defined. Adaptation specifications in the navigation model determine the current relevance of operational features in dependence of the context relations and the degree of activation of context concepts (cp. [6]). According to these specifications and the relations defined in the context relations model, none, one or several appropriate operational features are selected for which user interfaces have to be presented in the current Web page. This selection mechanism is independent of the exact representation of the operational features in the domain ontology. It is merely concerned with the selection of operational features that fit into the contextual situation and the user's navigational position.

Let us consider the car rental example scenario and suppose that the user has accessed the homepage of the Web portal for automobile services. A relation between the ontology entry of the homepage and the ontology entry of the reservation feature has been defined in the navigation model. Furthermore, a context relation has been modeled between the reservation feature and the 'owns car' concept of the user context ontology; this concept is activated if the user owns a car. At last, an adaptation specification has been defined, stating that a user interface for the reservation feature should be presented if the related context concept ('owns car') is deactivated. As a result of this modeling, the reservation feature will be presented directly on the homepage to users who do not own a car whereas car owners reach it only via navigation. The context-dependent selection of user interfaces should be considered as supporting rather than withholding. Clearly, all essential functionality should always be alternatively accessible by the user (e.g. via navigation).

5.3 Parameterization

In the next step, the user interface is pre-parameterized according to the contextual situation to provide initial support for user interaction. To achieve this, context relations are defined between parameter entries of the operational features and concepts of the context ontologies. The arrows linking the two windows in Figure 3 illustrate these relations. The activated concepts of the context ontologies (window on the right hand side) determine the parameter value selection in the Web page (window on the left hand side). In the example given, the parameter 'Model' is mapped with the user's favorite car model. Likewise, the

parameter 'Color' with the user's favorite car color. The parameter 'Convertible' is mapped with the season and the 'Pick-up' point with the user's current location. If a mapped context concept is activated, the corresponding value is handled as the default value and is preselected in the user interface. The way a value is preselected depends on the type of user interface element, which is determined in the subsequent generation step in accordance to the contextual conditions.

Again, automation possibilities depend on the degree of semantic description and shared understanding of the concepts. The automatic retrieval of context concepts that match the required parameter values (e.g. the favorite car model) is conceivable in a semantically rich scenario. Ultimately, if for every parameter value a matching context concept is found or modeled, no user interaction at all is required. However, user interaction (and control) might be necessary and useful in many cases.

Generally, the danger of erroneous mapping or incorrectly retrieved context exists that confuses instead of support the user. Suppose another city than Duisburg would be preselected as pick-up location in the example scenario that is further away from the user's current location. Then, the user would possibly assume that there exists no pick-up location in Duisburg and would accept the preselection without verifying his (false) assumption.

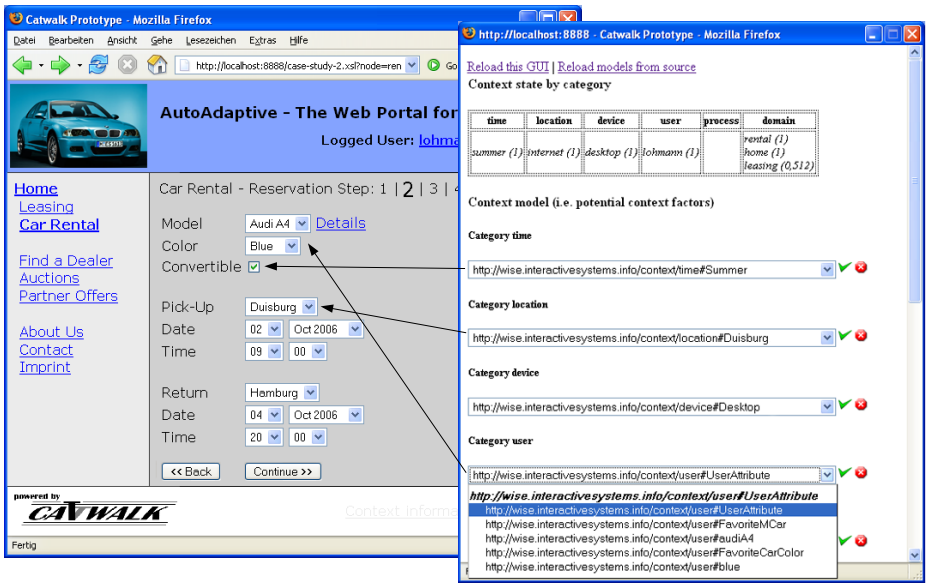

Fig. 3. Contextually adapted user interface of a car rental feature.

5.4 Presentation

In the last step, the user interface elements' look and feel is built. Alternative presentation forms come into question, depending on the situational relevance of an operational feature and the contextual conditions. CATWALK is designed to support the definition of various UI patterns for this purpose. Each pattern consists of an XSLT-template (and optionally an additional CSS-stylesheet) and is referenced in the domain ontology. Similar to the modeling of the navigational structure, relations between pattern entries and entries of operational features are defined – this time in the presentation model. These relations determine for each operational feature or a set of features which patterns are suitable. Furthermore, relations between pattern entries and concepts of the context ontologies are modeled in the context relations model. Adaptation specifications define which pattern should be selected for an operational feature in accordance to the context relations and activated concepts.

The left window in Figure 3 shows a possible implementation of a user interface for a part of the car rental example scenario. Suppose the user chose the vehicle type in the first step and now has to select the desired car model, some equipment details, as well as dates, times, pick-up, and return locations. The user accesses the Web portal via a desktop PC in the example scenario – the corresponding concept of the device context ontology is activated. Due to the modeled context relations and the adaptation specifications of the presentation model, a pattern is selected that is suitable for desktop PCs. Alternative patterns and respective context relations and adaptation specifications can be defined for other client devices such as PDAs or cellular phones. Varying patterns can also be used in dependence of a feature's relevance or for different user types (e.g. for visually handicapped people, a CSS-stylesheet defining larger GUI-elements might be selected).

Information required for the generation of suitable user interface elements is either stored in the domain ontology or have to be retrieved from the Web Service. Two ways of retrieval have to be distinguished: using the Web Service's WSDL description, in particular the XML Schema definitions, or using Web Service calls. The first allows the creation of a suitable (X)HTML form element for every XML Schema element by the pattern's XSLT-stylesheet, depending on the type of Schema element and (possibly) the number of provided value options. In the example scenario, the Web Service's WSDL description defines the complex type 'Model' that includes a list of car models that are all permitted parameter values. The XSLT-stylesheet transforms these values in an HTML dropdown listbox. Likewise, it transforms the complex type 'Color' that defines a list of car colors. The Boolean type 'Convertible' is transformed in an HTML checkbox. Corresponding data types are assigned to the parameters 'Date' and 'Time' in the WSDL description - common date and time picker elements are created. Again, semantically richer Web Service descriptions and a shared understanding of interface parameters would help to enhance the transformation of interface parameters to HTML form elements.

In the simplest way, the Web Service is invoked after submission of the whole HTML-form for the first time. Then, the selected values and their interdependencies are validated by the Web Service and possibly a message informing about conflicting values is send back. In such a case, the user might have to select different parameter values again and again until all value conflicts are resolved (e.g., the desired car might not be available for a specific location, date, and time). A more comfortable way is to trigger Web Service calls after certain user interactions to update the list of parameter values that can be selected by the user taking interdependencies into account. *Ajax*-based techniques are appropriate for such an implementation. Ultimately, the Web Services determine how sophisticated the user interface can be by providing or not providing such functionality. In the simplest case, the XML Schema element name is used for caption and text boxes are created allowing the input of parameter values (with preselected recommendations, see section 5.3).

6 Conclusion and Future Work

The homogenous integration of user interfaces for interaction with operational features is an emerging issue in the course of the evolution of Web applications from simple information systems to complex interactive applications. In the presented service-oriented approach, contextual influences have been considered in the modeling process right from the start for different parts of the generation process: selection, parameterization, and presentation. The approach builds upon an ontology-based modeling method and upon the CATWALK framework that provides run-time generation of adaptive, context-aware Web applications from these models.

We have shown how the incorporation of contextual knowledge can support user interaction and may lead to better usability that could make the additional modeling effort worthwhile in certain cases. We have also discussed some automation possibilities, especially in conjunction with Semantic Web Services. Likewise, it has become apparent that incorrect adaptation can confuse the user and reduce interaction quality. Thus, automation possibilities are restricted to some degree and careful modeling is demanded. The empirical investigation of adaptation effects is a difficult task; however, heuristic methods should provide a good basis for design decisions in many cases.

The presented approach is independent of specific context sensing mechanisms. However, possibilities for the exchange and the evaluation of externally sensed context information would be useful extensions. Other topics for future work include the definition of various context-specific UI patterns as well as a better support for the modeling of interaction processes. The empirical investigation of different adaptation strategies and their effects on usability issues are further topics of interest.

Acknowledgements

This work was partially supported by the German Federal Ministry of Education and Research (BMBF) under grant no. 01ISC30F.

References

1. Baresi, L., Garzotto, L., Paolini, P.: From Web Sites to Web Applications: New Issues for Conceptual Modeling. In Proceedings of the Workshops on Conceptual Modeling Approaches for E-Business and The World Wide Web and Conceptual Modeling: Conceptual Modeling for E-Business and the Web, London, UK. Springer LNCS 1921 (2000) 89–100
2. Ben Mokhtar, S., Fournier, D., Georgantas, N., Issarny, V.: Context-aware Service Composition in Pervasive Computing Environments. In Proceedings of the 2nd International Workshop on Rapid Integration of Software Engineering techniques (RISE'05), Heraklion Crete, Greece. Springer LNCS 3943 (2006) 129–144
3. Ceri, S., Daniel, F., Matera, M., Facca, F.: Model-driven Development of Context-Aware Web Applications. ACM Trans. Inter. Tech. (TOIT) 7(2) (2007), to appear
4. Ceri, S. et al.: Designing Data-Intensive Web Applications. Morgan Kaufmann (2002).
5. Hesse, W.: Ontologies in the Software Engineering Process. In Proceedings of the 2nd Workshop on Enterprise Application Integration (EAI'05), Marburg, Germany. CEUR 141 (2005)
6. Kaltz, J.W.: An Engineering Method for Adaptive, Context-aware Web Applications. PhD thesis, University of Duisburg-Essen. Utz (2006). Also published online at http://purl.oclc.org/NET/duett-07202006-093134
7. Kaltz, J.W., Ziegler, J., and Lohmann, S.: Context-Aware Web Engineering: Modeling and Applications. RIA - Revue d'Intelligence Artificielle, Special Issue on Applying Context Management 19(3) (2005) 439–458
8. Kappel, G., Pröll, B., Retschitzegger, W., and Schwinger, W.: Customisation for Ubiquitous Web Applications - A Comparison of Approaches. Int. J. Web Eng. and Technol. (IJWET) 1(1) (2003) 79–111
9. Koch, N.: Software Engineering for Adaptive Hypermedia Systems: Reference Model, Modeling Techniques and Development Process. PhD thesis, Ludwig-Maximilians-University Munich (2001)
10. Kraus, A., Koch, N.: Generation of Web Applications from UML Models using an XML Publishing Framework. In Proceedings of the 6th World Conference on Integrated Design and Process Technology (IDPT'02), Pasadena, USA (2002)
11. Manolescu, I. et al.: Model-Driven Design and Deployment of Service-Enabled Web Applications. ACM Trans. Inter. Tech. 5(3) (2005) 439–479.
12. Martin, D. et al.: Bringing Semantics to Web Services: The OWL-S Approach. In Proceedings of the 1st International Workshop on Semantic Web Services and Web Process Composition (SWSWPC'04), San Diego, USA. Springer LNCS 3387 (2003) 26–42.
13. Seffah, A., Javahery, H.: Multiple User Interfaces: Crossplatform Applications and Context-Aware Interfaces. J.Wiley (2003)
14. WISE - Web Information and Service Engineering http://www.wise-projekt.de (2006/Oct/28)