# Dynamic Generation of Context-Adaptive Web User Interfaces through Model Interpretation

Steffen Lohmann , J. Wolfgang Kaltz, Jürgen Ziegler
University of Duisburg-Essen
IIIS, Lotharstrasse 65
47057 Duisburg
{lohmann, kaltz, ziegler}@interactivesystems.info

## ABSTRACT
The model-driven generation 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 dynamic generation of user interfaces. For graphical user interfaces that allow the control of operational features, three parts of the generation process are distinguished: selection, presentation and parameterization. The presented approach is based on a framework for model interpretation and for the generation of context-adaptive Web applications. It is discussed together with an exemplary case study for better illustration.

## Categories and Subject Descriptors
D.2.2 [**Software Engineering**]: Design Tools and Techniques – *user interfaces.* H1.2 [**Models and Principles**]: User/Machine Systems – *human factors.* H5.2 [**Information Interfaces and Presentations**]: User Interfaces – *graphical user interfaces, user-centered design.* H.5.4 [**Information Interfaces and Presentations**]: Hypertext/Hypermedia – *architectures, user issues.*

## General Terms
Design, Human Factors.

## Keywords
context-adaptive Web user interfaces, context-aware integration, model interpretation, ontology-driven modeling, automatic generation, parameterization.

## 1. INTRODUCTION
The systematic development of complex applications requires a significant effort in modeling throughout the whole life cycle. For reasons of efficiency and consistency, 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 an application framework following this design paradigm by interpreting models at run-time for dynamic generation of context-adaptive Web applications (cp. [4]).

Using this framework as reference, we discuss in this paper how graphical user interfaces for operational features can be dynamically generated and adapted according to the context of use to enhance user interaction and reach better usability. First, we provide some background knowledge by discussing related work in Web application modeling, and by providing a summary of our modeling approach and the corresponding reference framework.

## 2. MODELING AND GENERATION OF WEB APPLICATIONS
Several approaches for the systematic development of Web applications (Web Engineering) build upon a conceptual model that describes the basic concepts and relations of the application's domain. Further aspects such as the application's navigational structure, its process flow or presentation issues are defined on the basis of this conceptual model. To allow the development of context-adaptive Web applications, additional modeling is required.

### 2.1 Related Work
The *UML-based Web Engineering* (UWE) approach [7] explicitly addresses adaptability issues and provides separate user and adaptation models. UML is used for modeling; the models are stored in XMI. For generation of application code from the UWE models the development framework *Apache Cocoon* has been extended [8]. 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 completed manually first. Furthermore, UWE concentrates on the modeling and adaptation of navigational and presentational issues (as do many Web Engineering approaches). The integration of operational features and the generation of corresponding user interfaces are not covered by UWE.

For the XML-based *Web Modeling Language* (WebML) [2] an attempt has been made recently to integrate operational features via Web Services [9], but it is not discussed how to generate user interfaces for these features. Further, some possibilities for the consideration of context in WebML were examined [1], but they are restricted to a level where whole pages can be marked as being relevant for certain contextual conditions.

Generally speaking, existing Web engineering approaches do not consider contextual influences in their modeling and application generation processes to a full degree. They typically consider either information about the user or about the location (see also [6]). The majority of approaches that discuss adaptivity are

concerned with the issue of how the application's navigation or contents can be adapted. The generation of context-adaptive user interfaces for operational features is not addressed by existing Web engineering approaches.

## 2.2 Ontology-driven modeling

Our approach originates in the WISE project [10], where ontologies are used for Web application modeling. In comparison to other modeling techniques, ontology-driven software engineering allows for advanced semantic expressive power in modeling and model exchange (cp. [3]). Especially for the interoperable integration of contextual knowledge, ontology-based modeling seems auspicious. The model base of our approach is a repository consisting of the following models (cp. figure 1):

- A *domain ontology*, describing concepts and conceptual relations of the application's domain as well as referencing resources used by the application.
- Several *context ontologies*, describing concepts and conceptual 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 relations between concepts of the domain ontology and concepts 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 concepts and the defined context relations.

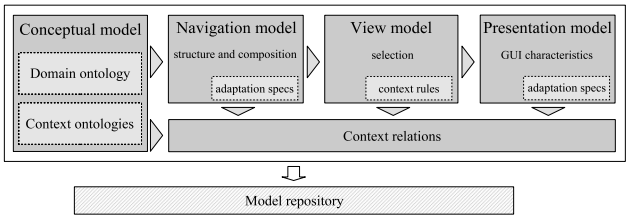


**Figure 1. Context modeling in the WISE methodology**

## 2.3 The CATWALK Framework

CATWALK [4] is a component-oriented framework that interprets the models at run-time to generate a context-adaptive Web application. It is based on Apache Cocoon; figure 2 gives an architectural overview. 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 in CATWALK 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 artifacts (such as XSLT stylesheets for XML transformation). The model repository is accessed in each generation step via a Cocoon pseudo-protocol and the corresponding model parts are interpreted at run-time. A central component (the *Adaptation Engine*) supports adaptive system behavior by interpreting context relations and adaptation specifications and considering the respective contextual state (provided by the *Context Manager* component).

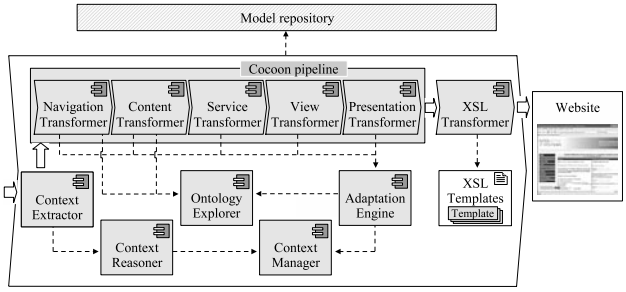


**Figure 2. CATWALK component architecture**

Our understanding of *context* is rather generic, including different aspects such as the user's profile, the current task and location, and the device used. In [5] we provide a formal definition of context.

# 3. GENERATION OF CONTEXT-ADAPTIVE USER INTERFACES

Contextual knowledge can affect different parts of the generation process. For the operational features of an application, the following must be addressed: the question of in which situations a user interface for an operational feature should be generated for the current Web page (selection), what the user interface should look like (presentation), and which values can be preselected (parameterization). In the following we discuss the different parts of the generation process in detail. For better illustration we describe an example scenario: a Web portal for automobile services that provides a car rental functionality.

## 3.1 Representation

The CATWALK framework follows a service-oriented approach where operational features offered by the Web application are encapsulated in Web Services. Each feature is conceptually divided in its discrete operations and for each operation an ontology entry is defined. The entries are interconnected by ontological relations; one of the entries is called the *function representative* and stands for the whole function. Knowledge about dependencies between the feature's different operations is thus part of the domain ontology.

Consider for example a car rental feature consisting of four discrete operations, which are realized by two Web Services – one for the selection of the (1) desired vehicle type and (2) equipment, and another for the (3) booking and (4) payment procedure. For each operation, an ontology entry is created referencing the corresponding WSDL elements. Additionally, parameters of the service can be represented in the domain ontology (see section 3.4).

## 3.2 Selection

A first step in the generation process consists of the selection of the operational features for which user interfaces should be generated in the current Web page. In this step, the whole operational feature is considered, not its discrete operations. The navigation model defines the navigational structure of the Web application by means of relations between elements of the domain ontology. This ontology includes the function representatives. 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 function representatives 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 activation degree of concepts (cp. [4]). 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 should be generated in the current Web page.

Let us consider the car rental example scenario and suppose that a user has accessed the homepage of the portal. A relation between the entry of the homepage and the function representative of the reservation feature has been modeled in the navigation model. Furthermore, a context relation has been modeled between the function representative and the 'owns car' concept of the user context ontology. This concept is activated if the user owns a car and deactivated if the user does not own 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 is activated. 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.

Thus, two users do not necessarily see the same user interfaces though they chose an identical navigational path through the application. The context-dependent selection of appropriate 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).

## 3.3 Presentation

After determining which operational features should be provided in the current Web page, in the next step appropriate user interfaces must be build. Depending on the situational relevance of an operational feature and the contextual conditions, alternative presentation forms come into question.

The CATWALK framework is designed to support the definition of various GUI patterns for this purpose. Each pattern consists of an XSLT-template and optionally an additional CSS-stylesheet. Similar to the modeling of the navigational structure, ontological relations between entries of patterns and other entries of the domain ontology (e.g. function representatives) are defined in the presentation model. Furthermore, relations between pattern entries and context ontology concepts are modeled in the context relations model. Adaptation specifications define which pattern should be selected in accordance to the context relations and activated concepts. At run-time, CATWALK reads the parts of the Web Services' WSDL-files which are required for the current operations, and applies the respective XSLT-stylesheet in accordance to the contextual situation.

The left screen in figure 3 shows a provisory implementation for the car rental scenario. Suppose the car rental feature is provided as a guided tour. In the first step the user chose the vehicle type; now the user has to determine the car model, some equipment details as well as dates, times, pick-up and return location. In case of the example scenario the user accesses the Web portal via a desktop PC – 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. For other client devices such as PDAs or cellular phones, alternative patterns and

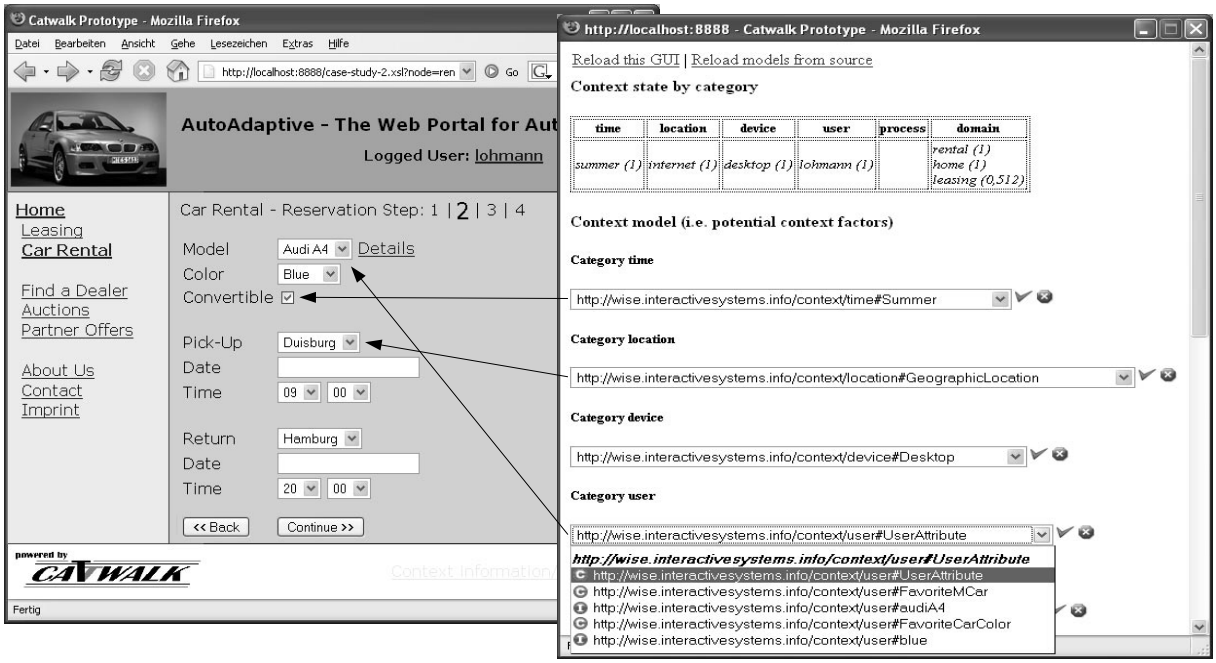


**Figure 3. Contextually adapted user interface of a car rental feature.**

respective context relations and adaptation specifications can be defined. 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 bigger GUI-elements can be selected).

The user interface is generated from the Web Service's WSDL description. The XSLT-stylesheet renders for each parameter a suitable XHTML form element for either the selection or the input of a return value, depending on the data type and the number of provided return values (the interplay of form elements is not considered here). The parameter name is used for labeling, permitted return values can be derived from the XML Schema definition.

## 3.4 Parameterization

In the last step, the user interface is pre-parameterized according to the contextual situation to provide initial support for user interaction. To achieve this, context relations must be defined between parameters of the operational features and concepts of the context ontologies. The arrows linking the two screens in figure 3 illustrate these relations. The activated concepts of the context ontology (screen on the right hand side) determine the parameter selection in the Web page (screen on the left hand side). In the example given, the parameter 'model' is mapped with the user's favorite car model and 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 and belongs to the permitted return values it is selected as default in the user interface (e.g. by preselection in the drop-down listbox).

Generally, the danger of erroneous mapping or incorrectly retrieved context exists and can confuse instead of support the user.

# 4. CONCLUSION AND FUTURE WORK

In this paper, we have presented an approach for the incorporation of contextual knowledge in the dynamic generation of Web user interfaces for operational features. Contextual influences are considered in the Web application modeling process right from the start for different parts of the generation process: selection, presentation, and parameterization. The approach builds upon the CATWALK framework that provides run-time generation of context-adaptive Web applications. 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. 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 integration of reliability issues for contextual knowledge would be a useful extension. Other topics for future work are the definition of various context-specific GUI patterns as well as a better support for the modeling of interaction processes. The empirical investigation of different adaptation strategies and their effect on usability issues are further topics of interest.

# 5. REFERENCES


[1] Ceri, S., Daniel, F., and Matera, M.: Extending WebML for Modelling Multi-channel Context-aware Web Applications. In *Proceedings of the 4th International Conference on Web Information Systems Engineering WISE - MMIS'03 Workshop*, IEEE, 2003, 615-626.

[2] Ceri, S., Fraternali, P., and Bongio, A.: Web Modeling Language (WebML): A Modelling Language for Designing Web Sites. *Computer Networks, 33(1–6)*, 2000, 137-157.

[3] Hesse, W.: Ontologies in the Software Engineering process. In (R. Lenz et al., Ed.): *EAI 2005 - Proceedings of the Workshop on Enterprise Application Integration, GITO,* Berlin 2005.

[4] Kaltz, J.W., and Ziegler, J.: Supporting Systematic Usage of Context in Web Applications. In *19th International FLAIRS Conference, special track on Modeling and Applying Contexts in the Real World (MAC-06).* AAAI, 2006.

[5] 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)*, 439-458.

[6] Kappel, G., Pröll, B., Retschitzegger, W., and Schwinger, W.: Customisation for Ubiquitous Web Applications - a Comparison of Approaches. *International Journal of Web Engineering Technology 1*, 2003, 79-111.

[7] Koch, N.: *Software Engineering for Adaptive Hypermedia Systems: Reference Model, Modeling Techniques and Development Process*. PhD thesis, Ludwig-Maximilians-Universität München, 2001.

[8] Kraus, A., and 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), 1*, 2002.

[9] Manolescu, I., Brambilla, M., Ceri, S., Comai, S., and Fraternali, P.: Model-driven design and deployment of service-enabled web applications. *ACM Transactions on Internet Technology, 5(3)*, 2005, 439-479.

[10] Wissen, M., and Ziegler, J.: A Methodology for the Component-Based Development of Web Applications. In *Proceedings of 10th International Conference on Human - Computer Interaction, 1*, Crete, Greece, 2003.