Prototyping Navigation in Web-Based Information Systems Using WebML

Jaroslav KURUC[1], Peter DOLOG[2] and Mária BIELIKOVÁ[1]

[1]*Institute of Informatics and Software Engineering, Faculty of Informatics and Information Technologies, Slovak University of Technology, Ilkovičova 3, SK-84216 Bratislava, Slovakia*
{kuruc,bielik}@fiit.stuba.sk

[2]*L3S Research Center, University of Hannover, Expo Plaza 1, D-30539 Hannover, Germany*
dolog@l3s.de

Abstract. This paper describes an approach to prototyping navigation in web-based information systems. The WebML language is used as a tool for modeling navigation aspects of such applications. As the information systems are intended to support user tasks, proper navigation mechanisms are crucial to make the user access to information efficient. Our approach is intended to support prototyping of navigation as a mean for communicating design decisions and model evolution. It is based on extracting a state machine from the WebML hypertext model.

Keywords: Web Modeling Language (WebML), hypertext model, navigation prototyping, state machine visualization, XSLT.

1 Introduction

Several model driven approaches to development of web-based information systems showed effective extensions of software design methods to support modeling specific features of the web. Content, navigation, and presentation design views are common determinants of all of the methods as reviewed for example in [8, 7].

While many methods are also supported by case tools, which enable a seamless transition from design models directly into implementation with just a small intervention of a designer, they do not concentrate on communication between stakeholders and designers during the model evolution. We experienced importance of this aspect in a project for developing web-based scientific conference management system CONFESS [9].

Prototyping has proved very useful to communicate design decisions to stakeholders in software projects. In this paper we report on an approach for prototyping navigation in web-based information systems using the Web Modeling Language (WebML) [3, 4], a language for design of data-intensive web applications. The prototyping is based on abstracting from several details of the WebML hypertext model needed for a generation of final web-based application and emphasizing just features important for communicating the high level user oriented navigation model.

Karel Ježek (ed.), DATAKON 2004, Brno, 23.-26. 10. 2004, pp. 1-10.

State machines provide us with means to model abstract navigation models from user interaction point of view [6, 2]. The state machines have been chosen as a base for the prototyping navigation.

The proposed solution has following advantages:

- it provides the designer with a simple and effective technique of extracting the user interaction aspect of navigation from the WebML models;
- it supports the navigation analyst in his task of communicating navigation design decisions and navigation model evolution.

The paper is structured as follows. A short overview of WebML as a tool for web-based data-intensive applications development and navigation modeling is given in Section 2. Our improvement of design the application front-end using navigation prototyping is proposed in Section 3. Next (Section 4), we describe the WEBModelLer system, which implements navigation prototyping in the WebML. Finally, we give conclusions and proposals for further work.

2 Web Modeling Language

2.1 Overview

The Web Modeling Language is a visual language for conceptual modeling of web applications. WebML also supports corresponding XML representation, which is used as a source for a generator, which can translate the model into the skeleton of the developed system (web page templates).

WebML defines the following conceptual models [3]:

- *data model* – defines the data used by the application. It exploits the classical Entity-Relationship model with generalization hierarchies, typed attributes and cardinality constraints;
- *hypertext model* – defines the organization of the front-end interfaces of the application; it concentrates on an application hyperspace topology of the system – defines the organization of the content using basic units, pages, links and unit composition within pages;
- *content management model* – defines operation units that can be used to manage and update the content and can also invoke externally defined programs.

The process of web-based application development with the WebML incorporates also presentation specification (definition of the layout and graphical appearance of the pages).

WebML explicitly defines a simple user model, which serves mainly for authentication and authorization purposes. The data model contains the following mandatory entities: `User`, `Group` and `Site-View`. The `User` unit stands for the user account, the `Group` entity stands for the user role and the `Site-View` entity determines a part of the hypertext model unique for particular user role. The user is a member of one or more groups, where every group has assigned one site-view. Public site-views, which all users can access can be also specified.

2.2 WebML and Navigation Modeling

One of the basic characteristics of web-based applications is the navigation. This feature enables accomplishing required tasks by means of browsing through different paths within the application hyperspace. The complex application hyperspace can result in a problem of being lost.

The WebML hypertext design phase associates definition of three views: page structure of the particular site-views, contents of the pages and operations performed over the contents (hypertext and content management models).

Several types of units are defined:

- *container* – encapsulates units into logical parts (e.g., page, page area or an alternative page);
- *content unit* – defines content of a page (e.g., index unit, data unit, multidata unit, entry unit);
- *operation unit* – defines operations executed over the content units (e.g., login unit, create unit, delete unit, modify unit, connect unit, generic operation unit).

All mentioned units are connected by links. Links express the navigation within the application information hyperspace as well as information transfers from one unit to another. The link can also trigger a computation, such as the content update operation activated by a `submit` link that users follow after entering new data.

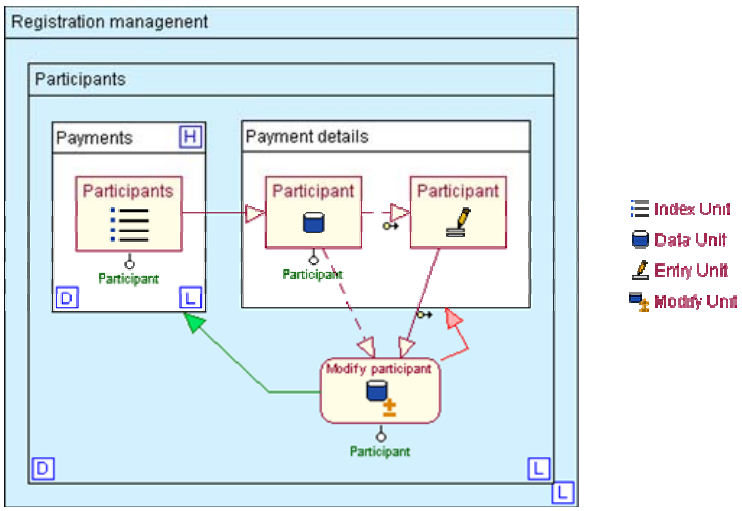

Figure 1. Sample fragment of the WebML hypertext model.

Figure 1 depicts an excerpt of a WebML hypertext model focusing on an entity modification. Depicted fragment of the hypertext model contains the `Payments` page with the `Participants` index unit surrounded by the `Participants` page area and consequently by the `Registration management` page area. The index unit represents the list of participants and the cost of payments they have paid. Selecting one item from the list of the `Payment details` page is showed. This page contains an entry unit (form), where the user is able to change the payments amount. After

submitting this form, the `Modify participant` operation unit is fired. The system attempts to save changed data. Providing the operation succeeds, the `Payments` page is showed again, which is modeled by an *OK-link*. In case of failure, the `Payment details` page is showed, which is modeled by a *KO-link*. Page areas/pages with the Landmark attribute (letter L on the right bottom) are accessible by non-contextual links at any time. When the page area is selected, the default child page area/page (letter D on the left bottom) is showed. The `Payments` page is defined as a home page (letter H on the right top), i.e. it is showed when particular site-view is selected.

3 Extracting Abstract Navigation from WebML

3.1 State Machines as Abstract Navigation Models

As described in [6], state machines emulate the user interaction point of view in navigation. The WebML hypertext model depicts pages and links. A page represents interactive part of the system. Elements on the page allow a user to change the system's state by following one of the links leading from this page. Link is a tool to change the system's state. The link can lead to presenting data (another page) or it can also execute operations or external programs.

Considering mentioned characteristics of the WebML hypertext model we proposed following principles of its transformation to a state machine:

- page, as an interactive part, is represented by a state in the state machine,
- link, as a tool for changing the system's state, is represented by a transition between the states in the state machine.

Since we abstract page contents links leading from/to page contents are taken as links leading from/to page itself.

Except pages and links also operation units defined during hypertext design are considered. Operations do not display the content, but execute processing as a side effect of the navigation of a link. Two kinds of operation units are considered during the hypertext design. Each of them is transformed separately.

First, there are operation units with only one possible outcome. They always succeed. This kind of the operation unit has in the hypertext model one incoming and one outgoing link. It is transformed as follows: incoming link is represented by a transition between the state, which represents a page the incoming link is leading from, and the state which represents a page the outgoing link is leading to. No other state or transition representing the operation unit or the outgoing link is needed.

Second, there are operation units with more possible outcomes. They can succeed or fail (captured by OK-links or KO-links respectively). Prototyping navigation using a state machine leaves the possibility of the branch selection on the designer. The operation unit in this case is represented by a pseudostate of kind junction. An action representing operation defined by the operation unit is assigned to a transition leading to this pseudostate. The pseudostate has as many outgoing transitions as the number of possible outcomes of the operation. Every outgoing transition has a guard condition representing an operation resulting into particular outcome.

3.2 Method for Transformation WebML Hypertext Model to State Machine

We proposed the transformation method as follows. The input of the method is a WebML hypertext model. The output is the state machine, which maps the navigation from the input hypertext model.

The transformation is carried out according to the following procedure:

1. Create a composite state representing the application itself.
2. Create a transition from the initial state to the state created in step 1.
3. In the created state, for each WebML site-view:
   a. Create nested composite state representing particular site-view.
   b. If it is not protected, create a transition from the state created in step 1 to the nested state created in step 3.a.
   c. In the created nested state, for each top-level WebML unit:
      i. If it is an operation unit with more than one possible outcome:
         I. Assign an action representing operation of the operation unit to an incoming transition.
         II. Create a nested pseudostate of kind junction representing branch selection depending on the action outcome.
         III. For each possible action outcome, apply step 5 and continue.
         IV. For created outgoing transition, add a guard condition which maps a result of the operation.
      ii. If it is a page area, create nested composite state representing particular page area.
      iii. If it is a page:
         I. Create nested state representing particular page.
         II. For each link leading from this page or from units in this page, apply step 5 and continue.
      iv. If it is marked as landmark, create transition from the direct ancestor state to the state representing this unit.
      v. If it is a page area, in state representing this area:
         I. Apply step 3.c and continue.
         II. Create a transition from the initial state to the state representing default page area/page of this area.
   d. Create a transition from the initial state to the state representing home page of this site-view.
4. Go to the step 6.
5. If the destination of the link is:
   a. a page, create a transition to the state representing particular page.
   b. a page unit, where the page is not nested in another page, create a transition to the state representing particular page.
   c. a page unit, where the page is nested in another page, create a transition to the state representing parent page.
   d. an operation unit with only one possible outcome, create a transition to the state representing target of the possible outcome.
   e. an operation unit with more than one possible outcome, create a transition to the state representing particular operation unit.
6. End.

As an example of the method output, let us recall the example depicted in Figure 1. The WebML fragment in Figure 1 is transformed to the states/transitions in the resulting state machine depicted in Figure 2. Page areas/pages are represented by states and the default page areas/pages are modeled by transitions leading from initial states. A landmark is modeled by a transition from the parent composite state to the child state, i.e. the states are accessible at any time. Home page is modeled by a transition from the topmost initial state to the state representing the `Payments` page. Selection of the particular participant and possibility to edit corresponding payment is modeled by the `Details` transition to the `Payment details` state. Submission of the form is modeled by the `Change` transition, which leads to a junction pseudostate, where the following transition is selected according to the `Modify participant` operation result. In case the operation succeeds, transition with the `result=true` guard condition is followed. In other case, transition with the `result=false` guard condition is followed.

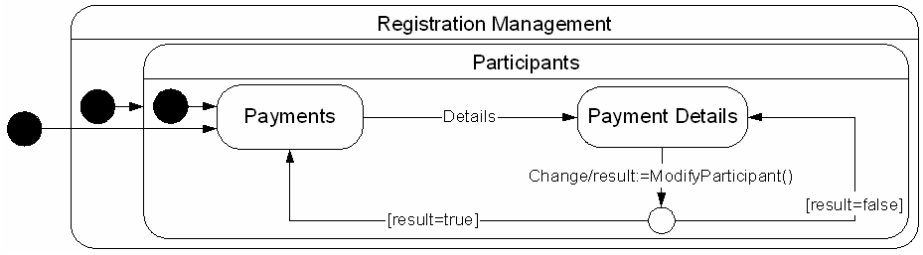

Figure 2. State machine fragment mapping the navigation from Figure 1 sample.

3.3 Visualizing State Machine

We have decided for a visualization approach of a state machine where we show current state with its outgoing transitions. After selecting one of the transitions, the state machine sets the current state to a new state into which the selected transition leads and consecutively shows this state.

Our aim was to support the design of navigation by prototyping its behavioral aspects. We proposed graphical representation of the current state showing a context of this state relative to other states together with GUI elements representing transitions leading from this state. We also designed graphical representation consistent with the WebML notation (WebML units graphical notation is preserved).

After transforming the hypertext model into a state machine the prototyping navigation starts by visualizing the initial state. The starting state represents the application itself (in our example the CONFESS system). This state contains transitions to all public site-views. In our case, there is only one public site-view represented by the `Public Site-View` state. After selecting the transition leading to this site-view, the corresponding state representing the `Public Site-View` is selected as current state (Figure 3).

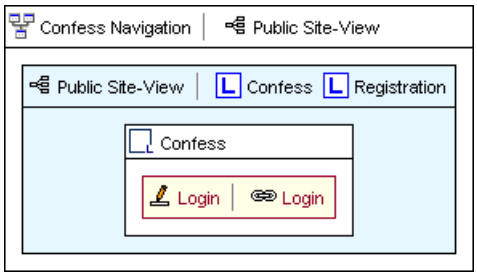

Figure 3. `Public Site-View` state.

By selecting the `Public Site-View` state, the default site-view page state is selected (the `Confess` page in our case). This page contains only one transition called `Login` leading from the WebML entry unit also called `Login`. The site-view state shows also transitions leading to states representing pages with the Landmark attribute (in our example transitions leading to the `Confess` and `Registration` pages). Transitions to public site-views are still available. By selecting the `Login` transition, the pseudostate representing branch following the login operation is showed (Figure 4). Note that the prototyping tool shows all possible transitions from the pseudostate, which is a target of the transition where the login operation is modeled as a side-effect action. The purpose of our prototyping tool is to show all the branches leading from a pseudostate representing the operation unit after the login operation. The figure depicts them as links.

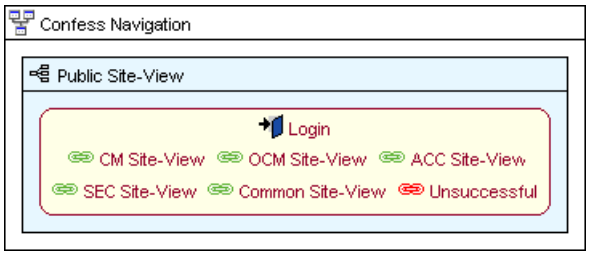

Figure 4. Login unit state.

When a pseudostate representing the operation unit is selected, only transitions representing results of this operation unit are showed. No other transitions like landmarks or site-views are displayed.

In our example (Figure 4), there are 5 OK-links called `CM Site-View`, `OCM Site-View`, `ACC Site-View`, `SEC Site-View`, `Common Site-View` representing logon in particular user role (Conference Manager, Organization Committee Member, Accountant, Secretary and Common user role) and 1 KO-link called `Unsuccessful`. By following one of the OK-links, a site-view state representing particular user role is selected and showed. By following KO-link `Unsuccessful`, the resultant state of the unsuccessful logon operation is showed (the `Login` page in our case again). When the designer selects for example the `OCM Site-View` transition, the state representing this site-view is selected and showed (Figure 5).

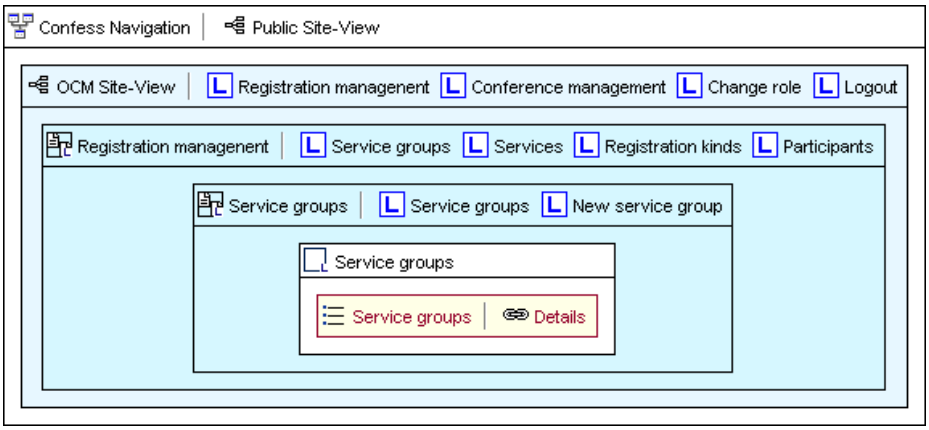

Figure 5. `OCM Site-View` state.

Similarly to `Public Site-View`, default page state `Service groups` is selected and showed. This page contains only one transition called `Details` leading from the WebML `Service groups` index unit. This page is wrapped by the `Service groups` page area, which is wrapped by the `Registration management` page area. Page area state, similarly to the site-view state, shows transitions leading to pages with the Landmark attribute.

4 WEBModelLer

The proposed transformation method and the visualization of the state machine were implemented in a prototype called WEBModelLer. Transformation method of structural WebML hypertext model into a state machine is realized by the XSLT engine. Source XML representation is the original WebML hypertext model representation. Target XML representation is the internal state machine representation used by the JStateMachine library (JSM, www.jstatemachine.org). To achieve graphical representation consistent with the WebML notation some additions to the internal XML representation of the state machine were proposed and implemented.

The visualization of the state machine is implemented using the JSM library. This library supports an interface for loading the state machine from its internal XML representation, listing transitions leading from the current state and changing current state by following one of these transitions. Thus, the process of the visualization embodies current state draw and the creation of the GUI elements representing transitions leading from this state. In our implementation transitions are presented using the buttons, where the click on the button stands for the follow of the particular transition.

The method of navigation prototyping using WebML hypertext model was validated on the CONFESS system. We designed the system using WebML. For illustration, the most complex site-view contained about 18 pages, 53 content units, 26 operation units and 125 links. Consequently, the designed hypertext model was transformed into the state machine and visualized. In the resulting state machine, no

more than 15 transitions (including landmarks) are shown at a time (due to proposed visualization method where only one page and links leading from this page are showed).

5 Related works

User interaction oriented navigation modeling was studied in [6, 10]. The state machines have been employed to model navigation between pages by means of transitions between states. Method for transforming UML state diagrams into web site navigation graph and visualization of that graph is proposed in [6]. In addition, side effect actions, transition guards, and events have been also employed to model adaptive behavior of web applications based on user profiles. The work reported in this paper presents the method for transforming WebML hypertext model into state machine and shows that the state machines can be useful foundation for tools used for prototyping.

The WebML hypertext model semantics was studied in terms of state machines already in [5]. The experiment with prototyping tool described in this paper proved the usefulness of state machines suggested as a dynamic view of WebML hypertext model. An integration of WebML hypertext and UML-Guide state machine model have been reported in [2]. The UML state diagrams are used to model and generate user oriented adaptive guides visualized using the navigation graph through a generic web applications modeled by and generated from WebML.

The focus of this paper is to prototype the behavioral aspect of the navigation by means of state machines as underlying model. The behavioral aspect of work described in this paper relates to the work on extending WebML hypertext from a workflow point of view [1].

6 Conclusions

In this paper, we have described an approach to navigation prototyping when designing data-intensive web applications using WebML. The WebML navigation model is represented by a static diagram where pages and links are defined. Our approval is based on extracting a state machine from the WebML hypertext model. The state machine serves as an underlying model in a prototyping tool. The tool is intended to support a navigation analysis task by helping to communicate the navigation model evolution and design decisions in the model.

Since the transformation/visualization method we proposed is an abstraction of the WebML hypertext model, it should be used in parallel with the regular way of creating the WebML hypertext model [3]. A possibility to follow the system state's changes can be also considered as a validation of the navigation model. This approach was proved as helpful alternative way to prototype a navigation in the hypertext model experienced in the CONFESS project.

In future work, we intend to improve our prototyping tool with possibility of two directional transformation between static hypertext model and its dynamic part which models the navigation.

Acknowledgements

This work has been supported by the Grant Agency of Slovak Republic grant No. VG1/ 0162/03 "Collaborative accessing, analysis and presentation of documents in internet environment using modern software tools".

References

1. Brambilla, M.: Extending hypertext conceptual models with process-oriented primitives, In *Proc. of* ER 2003 – Int. Conf. on Conceptual Modeling, October 2003, Chicago, LNCS 2813, Springer Verlag, pp.246-262.
2. Ceri, S., Dolog, P., Matera, M., Nejdl, W.: Model-Driven Design of Web Applications with Client-Side Adaptation. In *Proc. of ICWE 2004 – International Conference on Web Engineering*, July 2004, Munich, Germany, Springer Verlag.
3. Ceri, S., Fraternali, P., Bongio, A., Brambilla, M., Comai, S., Matera, M.: Designing Data-Intensive Web Applications. Morgan Kaufmann Publ., 2003.
4. Ceri, S., Fraternali, P., Matera, M.: Conceptual Modeling of Data-Intensive Web Applications. *IEEE Internet Computing*. July/August 2002, 20-30.
5. Comai, S., Fraternali, P.: A Semantic Model for Specifying Data-Intensive Web Applications Using WebML. In *Int. Semantic Web Workshop*, Stanford, USA, July 2001.
6. Dolog, P., Nejdl, W.: Using UML and XMI for Generating Adaptive Navigation Sequences in Web-Based Systems. In *Proc. of UML 2003 - Sixth Int. Conf. on the Unified Modeling Language: Modeling Languages and Applications*, October 2003, San Francisco, USA, Springer Verlag, LNCS 2863.
7. Dolog, P., Bieliková, M.: Hypermedia Systems Modelling Framework. *Computers and Informatics*. Vol. 21, No. 3, 2002, 221-239.
8. Fraternali, P.: Tools and Approaches for Developing Data-Intensive Web applications: A survey. *ACM Computing Surveys*, 31(3): 227-263, Sept. 1999.
9. Habala, R., Krupa, R., Kuruc, J., Marko, V., Šindelář, M.: CONFESS: Conference Support System. Product documentation, Team Project, Slovak University of Technology Bratislava, 2002.
10. De Oliviera, M.C., Turine, M.A., Masiera, P.C.: A Statechart-Based Model for Hypermedia Applications. *ACM Transactions on Information Systems*, Vol. 19, No. 1, January 2001, 28-52.