

Using UML and XMI for Generating Adaptive Navigation Sequences in Web-Based Systems

Peter Dolog and Wolfgang Nejdl

Learning Lab Lower Saxony,
University of Hannover,
Expo Plaza 1, 30539 Hannover, Germany,
{dolog, nejdl}@learninglab.de,
<http://www.learninglab.de/~dolog>,
<http://www.kbs.uni-hannover.de/~nejdl>

Abstract. Navigation in information intensive applications to retrieve required information is time consuming task. Without appropriate navigation support the browsing need not lead to required information. One of the navigation support is guiding a user through the information. The guidance has to be adapted according to different requirements and different goals the user can have. A system can adapt the appearance of information and links or links can lead to different alternatives of information based on observed user preferences, goals, level of knowledge. Parametrizing allow us to capture such variability already in the modelling of such applications. Such parametrized models can serve as an input to a generator which will generate appropriate navigation sequence based on the parameters' values. In this paper we discuss a method for generating such adaptive navigation sequences from the UML state diagrams. The method is discussed on a case of adaptive e-course. Latest advances in UML model representation by means of XML based metadata interchange format can be successfully utilized for adaptive generation of the adaptive navigation sequences and can speed up a prototyping of navigation support in adaptive web-based systems. Adaptive generation means that generator can be parametrized. The parameters' values are set by user interaction with a system. According to the parameters the generator can generate modified navigation support and appearance of information based on the observed user features. The widely accepted standard based means and tools for XML technology are used for implementing a method for transforming UML state diagrams into web site graph and visualisation of that graph.

Keywords: Adaptive web-based systems, generator, the UML state diagrams, XMI

1 Introduction

Information intensive applications on the web enable significantly higher number of users to access information they provide. This is on the one hand a benefit but on the other hand it requires to handle new conditions. One size fits all approach is not suitable for building such applications any more. The applications should reflect different requirements of users with different background, technical environment, political, social environment, interests, goals and so on.

Personalization and adaptation of such web based information intensive application is apparent in applications supporting several fields. The examples of such personalization can be found in customer relationship management, e-commerce systems, or adaptive educational systems. The applications adapt themselves to certain user features. It means that applications provide different information variants, appearance variants, or links based on user features or provide certain information or links just when some constraints on user features are fulfilled.

System models can aid the developers of such applications to overcome increasing complexity caused by the need to maintain different variants of information, its presentations and parameters which determine when the information should be presented. The models can serve for inputs into generator which generates parts of the final application.

In previous papers we dealt with different aspects of hypermedia application modelling [8]. We introduced the UML state diagram based method for adaptive hypermedia application modelling [9]. We extended the method by introducing domain engineering based feature modelling into hypermedia engineering arguing that hypermedia applications can benefit from domain conceptual and feature models reused in application family [10].

Since UML models can be stored in XMI file, the OMG standard XML metadata interchange format files, it is possible to process and manipulate that files by standard XML processing tools (XML parsers, XSLT parsers and so on) [20]. This simplifies building generators based on UML models and also simplifies production of hypermedia applications.

Adaptive hypermedia application can benefit from such model-driven approach even more. The model can be used as a base for generation of navigation sequences and adaptive presentation of different slots of information. The XML processing tools allow us to dynamically invoke parsing of XMI file whenever an event generated from user interface occurs and thus to provide dynamic (re-)generation of navigation objects and maps according to rules involved directly in the model.

In this paper we explain how the models (UML state diagram models for navigation and class diagrams for user modelling) can be utilized for generation of adaptive navigation sequences. The method utilizes the availability of XML based standard for storing UML models — XMI. This enables to take full advantage of XML technology which is very popular for making web based applications.

The rest of the paper is structured as follows. Section 2 is devoted to introduction to adaptive hypermedia. Section 3 summarizes our approach to the adaptive navigation modelling. This method is based on the notion of browsing and its modelling by state diagrams. Several other models are used to support the navigation modelling by state diagrams. Section 4 describes the implementation of a generator. The discussion is based on adaptive navigation map which visualization is presented in section 4.1. A method for transforming state diagrams into the navigation map is discussed in section 4.2. A system implementation where the generator adaptively (re-)generates the navigation map according to values of user features and generator template parameters is discussed in section 4.3. Section 5 provides the discussion about related work. Section 6 draws conclusions and some remarks on further work.

2 Background: Adaptive hypermedia

Adaptive hypermedia is an alternative to the traditional “one-size-fits-all” static approach in the development of hypermedia systems [4]. Adaptive hypermedia are characteristic by their dynamics, i.e. they adapt themselves to the certain conditions. The adaptive hypermedia maintains several user features. They utilize the user features to determine appropriate information presentation and navigation sequences to explore the sufficient set of information. they update a user model according to a user interaction and information he provided.

Such systems use several techniques to help a user. The first group of techniques is trying to improve a local navigation of a user and his orientation in currently presented page or fragment. For example, such adaptive systems can provide different text or different media variants, which serve information at different level of detail, to users with different level of knowledge or expertise in some field. They can switch between different media types according to different user preferences or his learning style. They are able to hide or appropriately annotate the parts of presented information which are not suited for current user based on values of his features a system maintain. These techniques are called *adaptive presentation techniques* [4].

The later group of techniques is trying to improve user global orientation in hyper-space. They are trying to provide him with appropriate navigation support to enable him to more easily explore required information. This includes techniques such as enabling, disabling, showing, hiding, or removing links when it is appropriate. They sort links according to user goals or preferences. They can generate link destination according to several user features. They can annotate links to indicate whether the links are appropriate or relevant. They are able to generate next appropriate information and thus to guide a user. These techniques are called *adaptive navigation techniques* [4].

In this paper we are focusing on the adaptive navigation techniques modelling and generating appropriate navigation support for a user of such systems.

3 Adaptive Navigation Modelling

In this section we describe a navigation from the browsing point of view. We summarize the process of adaptive navigation modelling. We show an example of adaptive navigation model on the case of simple JAVA e-lecture.

3.1 Navigation as a browsing

Navigation in hypertext or hypermedia can be seen from several points of view. One possible point of view is browsing. The browsing can be seen as a following certain path in a hypertext graph. According to that a navigation model should concentrate on a user interaction during hypertext presentation or on a change of a hypertext state when the user performs the act of navigating. Nodes in such path can be either single or composed from several subnodes. Node can represent also subpaths. Navigation model of the browsing reflect the possible paths through information chunks and their contextual grouping. To develop such navigation model, an analytical model of information

structure and user roles should exist. We suppose that the analytical model incorporates main user roles and information structures. They are represented as concepts with their features and mutual relationships in a UML class diagram (see e.g. [14]) or its extension for feature modelling [10].

Adaptivity description means introducing variability into navigation models. This variability reflects different possibilities for presentation of particular information page or fragment, different possibilities of annotation of the fragments and links, different link destinations, and so on. This variability can be derived from variable features of concepts from the conceptual feature model [10]. A subset of different aspects of information modelled in the conceptual model can be selected and placed into navigation model. These features are then connected by links with parameters which reflect different constraints based on user features what represent variability in browsing.

Basic modelling technique for navigation modelling in our approach is a state diagram, which enables to model dynamic character of navigation. Event/condition/action paradigm supported in transitions and states allow us to parameterize the states and transitions and to specify different possibilities for transitions targets.

We proposed six basic steps of navigation modelling process: *identifying basic interaction scheme*, *identifying states*, *identifying transitions*, *identifying events*, *creating a user model*, and *mapping the user model elements to the state diagram*. These steps can be performed in parallel and in iterations. Moreover, proposed approach can be used at several levels of abstraction (of a hypermedia system).

3.2 The Process of Adaptive Navigation Modelling

I. Identifying basic interaction scheme. The first step in navigation modelling is basic interaction scheme modelling. This is intended to identify a sequence of interactions between main system roles. The UML sequence diagram is used for these purposes.

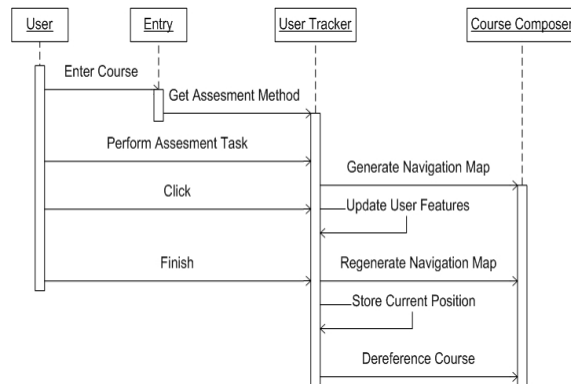


Fig. 1. A basic interaction scheme.

Figure 1 depicts three basic system roles (the Entry, the User Tracker, and the Course Composer) and the User role. When a user enters the system an assess-

ment test is generated. The user then performs an assessment task. A course is generated according to the assessed user feature values. It means that the course content is packaged into navigation sequence (path). When user interacts with the system by clicking for particular navigation object, the system evaluates and changes the user model state and regenerate navigation sequence and other navigation supporting features.

II. Identifying states. States in a navigation model fulfill the role of information chunks. They can be grouped into superstates. The states are created from an information model. There are two possibilities of mappings from information model: (1) a superstate mapped to a class with substates mapped to class attributes, and (2) a superstate mapped to a class instance with substates mapped to class instance attributes.

Parallel substates are mapped to attributes of a class or its instances, which are presented simultaneously. Attributes, which do not need to be presented simultaneously are grouped into ordinary substates. The classes, which are aggregates of another class are mapped to parallel or ordinary substates of that class' state. In addition, these substates are determined by the cardinality of the aggregation relationship. Specialised classes are mapped to ordinary substates. Special information chunks derived from several attributes and/or classes or special states needed for purposes of navigation can also be considered. States can be extended with a history. The history indicates that a user can start his browsing where he finished when exited system last time.

III. Identifying transitions. A transition represents an active interconnection between information chunks. Association relationships from information model are transformed to transitions. When it is needed, additional transitions can be incorporated into the model. The `fork` and `join` pseudostates, and `SyncState` are intended to model a synchronization of parallel states. The first two are intended for splitting or joining transitions. The latter is for synchronizing substates of parallel regions.

A condition can be assigned to the transition. A transition can also be conditionally forked; i.e. the transition can end in several states. Transition can also have associated time event for modelling sequential hypermedia timing. A transition can also have associated side effect actions, which together with transition conditions are very important for adaptation modelling.

IV. Identifying events. Events raise transitions in a state machine. Events can be directly mapped to presentation elements, which have associated actions. They are mediators between navigation model and presentation model of actions. Events can be joined to a generalization/specialization tree. An event can be mapped to more than one transition.

V. Creating a user model Adaptive navigation strongly depends on a user modelling. The user model incorporates various characteristics of users. Hypermedia application usage data are also represented in the user model. In our approach a user is modelled by a class diagram similarly to [14]. The user model is derived from user roles. Structure of the user model follows the well known Adaptive Hypermedia Application Model (AHAM) [3]. The user model should at least contain a class, which represents the level of user knowledge. Other classes (user preferences, goals, interests, knowledge, background, hyperspace experience, etc. — see e.g. [11]) can be incorporated when it is

needed. The user model contains operations for reading the current state of user characteristics and for updating user characteristics

Fig. 2 depicts a user model with the `UserKnowledge`, the `User` and the `Role` class. The `Role` class is intended to represent a user group. When a user is a member of the role he can not have his own level of knowledge associated because he takes role's level of knowledge (`{xor}`).

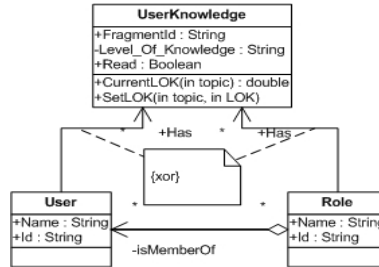


Fig. 2. A user model for e-lecture.

VI. Mapping user model elements to state diagram. The adaptive behavior is modelled by an introduction of features of user model classes into state diagrams. Accessible features of user model classes are mapped to guards conditions. The guards represent constraints on variable features of information and variable destinations of links. The guards are tested for specific values, which have to be satisfied when transition is raised. Operations are mapped to actions of transitions. They are used for upgrading the user model state or for specific operations with the user model and/or information chunks. Operations for retrieving current user model state can also be used in guard conditions. Guard conditions of transition specify local rules of adaptation. Global rules of adaptation, can be specified as guards of internal transitions, parts of entry, exit or do actions, and conditions of superstates.

3.3 A Navigation Model of an Adaptive Java e-Lecture

Fig. 3 depicts an adaptive navigation model for a simple JAVA e-lecture (introduction to JAVA). This is a part of e-lecture which is provided at the University of Hannover for Computer Science 1 course. The content of the lecture were created by lecturers of that course. In this paper we describe a model of the lecture as a result of sequencing redesign with the aim to provide adaptive navigation support in the lecture.

The lecture describes the main concepts of object-oriented programming in JAVA on the example of software version of BINGO game. The lecture starts with the `Requirements` for the game to communicate that each software projects starts with formulation of requirements. The lecture continues with the `A First O-O Analysis` where requirements for the BINGO game are analyzed and transformed to classes. This part of a lecture communicate the `Classes` and `Class Diagrams`. Student can invoke the

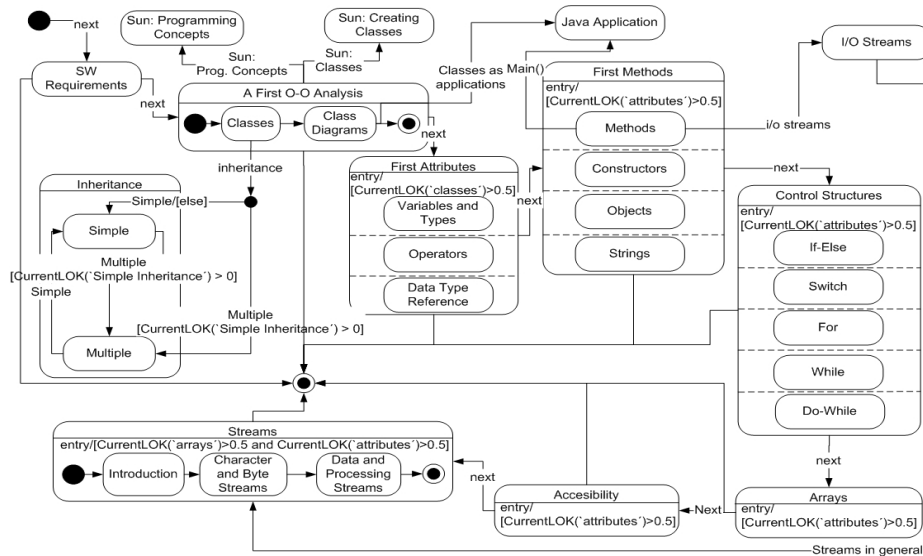


Fig. 3. A part of an adaptive navigation model for a JAVA e-lecture.

explanation of Inheritance where simple and also multiple inheritance is communicated as well. This part also points to main programming concepts in external Sun JAVA introduction. The lecture then goes more into detail of main concepts in o-o programming, namely the First Attributes, the First Methods, the Control Structures, the Arrays, the Accessibility (public, private, protected), and the Streams.

The Inheritance, the A First O-O Analysis, and the Streams are modeled as substate machines. It means that they are intended to be subsequences in this JAVA e-lecture. Only one of the concepts modelled as substates is then displayed at once. On the other hand, the main concepts about the First Attributes, the First Methods, and the Control Structures are presented in one window and thus the fragments are modelled as concurrent states.

The fragments can be represented for example by audio or video. In that case they model separate channels of presentation. When synchronization between these channels is needed, the state diagram technique allow us to introduce the synchronization symbols to model such situation.

There are several *variabilities* depicted in the model. Variability is considered in case of the Inheritance. When a user has already some knowledge about inheritance ($\text{CurrentLOK}(\text{'Simple Inheritance'}) > 0$), he is directed to the Multiple inheritance, otherwise he is pointed to the Simple inheritance. The user can switch between these two fragments. The transition (link) from the Simple inheritance to the Multiple inheritance is constrained as well.

Variability is considered also in all states (parts of e-lecture) except of the SW requirements and the A first O-O Analysis. It means that this parts are

suitable for a user when he has some knowledge about concepts mentioned in the guards in the `entry` actions.

The rules in the `entry` actions are depicted just in the high level states. This can be interpreted in several ways. One interpretation can be that rules can be valid just for high level state. When the parts of lecture which are represented by these states are not displayed the rules are de facto propagated into its substates because a user does not have any possibility to invoke the subparts. Another interpretation can be that they are propagated by generator and thus the same rules appear for the substates in an implementation. Another possibility is to write similar rules for substates. This is needed especially when they differ from those introduced at higher level states.

More complex rules on variability can be introduced in larger courses. Several presentation possibilities for all concepts can appear as variants (substates) in the state diagram together with rules when they can be invoked or annotated. Particular substate can have several outgoing transitions and rules can determine which transition is valid (is provided) to a user by means of link or button. For space limitation we are not able to provide such complex models here.

The operations for updating user profile are not presented in the figure (are collapsed) for simplicity reasons. They are usually modelled as actions of transitions or as exit actions of states. It means that a user profile is updated when particular link is clicked or when information fragment modelled by state is exited.

Time events can generate an update of a user profile as well. For example a level of knowledge about certain topic can be increased after some time spend in certain state. This can be modelled by internal transition of a state which is raised by a time event. The transition can reference an action for updating the user profile.

4 Generator implementation

The adaptive navigation model presented in previous section represents adaptive sequence of steps through the web site of a course or lecture (or through certain information). In this section we discuss a visualization of such web site map (graph). We discuss a method which transforms the state diagram model into the web site graph (navigation map). We describe an implementation of the generator based on the transformation method utilizing the XMI and the XSLT. We discuss an implementation of a system which use the generator to adaptively (re-)generate the navigation map as well.

4.1 Visualisation of the navigation map

One approach to the visualization of the navigation map is depicted in Fig. 4. We distinguish:

- *Folder symbol* — represents a composite information fragment composed from other composite information fragments, simple information fragments, groups of links or simple links;
- *Dashed box symbol* — represents a composite information fragment, which has to be presented concurrently with other composite information fragments (the dashed boxes) depicted on the same level;

- *Document symbol* — represents a simple information fragment; only links can be nested under the simple information fragment;
- *Arrow symbol* — represents a simple link to another composite or simple information fragment; the arrow symbols can be nested into the folder when they represent different alternatives of link destination from particular document (e.g. grayed folder Inheritance with two link alternatives: Simple and Multiple);

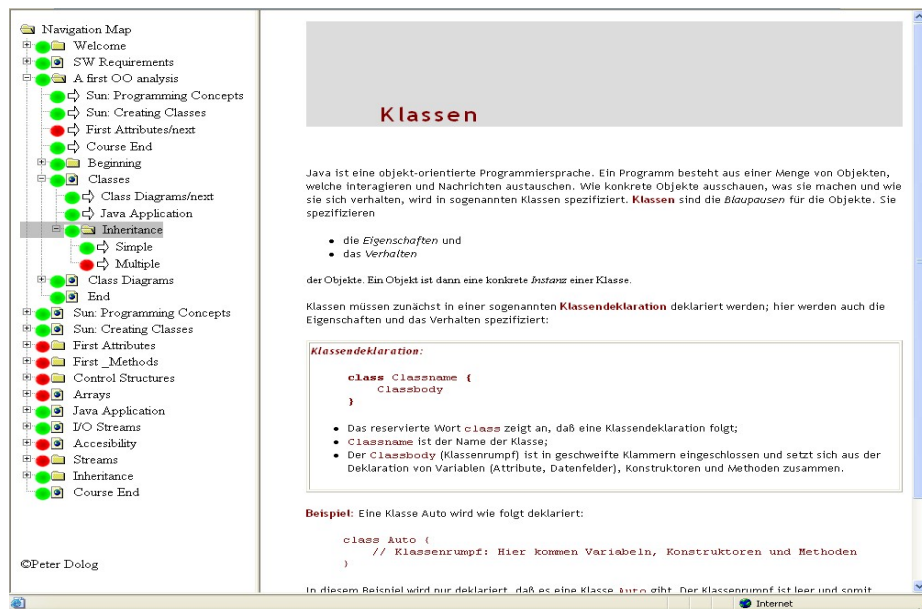


Fig. 4. Visualization of navigation graph for java e-lecture.

A *composition* is represented by the plus/minus symbol for showing/hiding enclosed items and by the left hand indent of enclosed items.

Each symbol can have the content associated. Arrows have the content and name of the corresponding target node associated. The arrows which represent guidance to the next fragment according to the course sequence has associated the “/next” text besides the name of the target fragment.

We have implemented described structure for web browsers. We have also implemented the functions for filling this structure and for interpreting the user actions on this structure. Current position of the user in the navigation graph is indicated by grayed background of the presented element. As it is obvious from Fig. 4, the navigation map was generated from the navigation model of JAVA e-lecture in UML state diagram depicted in Fig. 3.

Adaptive navigation support is realized depending on the generator parameters. It means that links can be either annotated by appropriate color from traffic light metaphor

or showed, hidden or sorted according to a user features values. When a node is annotated by green symbol it is appropriate for a user, when it is annotated by red symbol then it is not appropriate for a user. Yellow or other symbols can be used to indicate other information about the document, e.g. already read. The operations assigned to the transitions or to entry or exit events of the states are transformed to the same operation over internal representation. The skeleton for implementation of such operations is generated. The code of this operations have to be filled manually. Operations are created as member functions of user model classes where they are specified. We have implemented operations from the user model depicted in the figure 2.

4.2 Transforming State Diagrams into Navigation Map

A Metod. The input to the transformation method is a state diagram. The output is the navigation map. The transformation is carried out according to the following method:

1. Find the initial pseudostate and transform it to the *Folder* symbol. Nest outgoing transitions from the initial state as *Arrow* symbols under the created *Folder* symbol. Set the content of the *Arrow*-s to the outgoing transitions target states content. Make visible the *Arrows* whose guards of transitions are satisfied or annotate the *Arrows* by appropriate symbol from traffic lights metaphor (red, green, and yellow).
2. Transform each composite state to:
 - (a) The *Folder* symbol if it is a state with alternative substates or substatemachine.
 - (b) The *Dashed box* symbol if it is a concurrent region.

Apply these rules for all substates. Put substates under their composite ancestor. Make visible the components whose entry guards are satisfied or annotate the symbols by appropriate symbol from traffic lights metaphor (red, green, and yellow).
3. Transform each simple state to the *Document* symbol. Make visible the fragments, whose entry guards are satisfied, or annotate the nodes by appropriate symbol from traffic lights metaphor (red, green, and yellow).
4. Transform each transition to the *Arrow* symbol:
 - (a) If the target of a transition is a state, assign the content target of this state to the symbol.
 - (b) If the target is a pseudostate (junction, choice, join, fork), transform it to as much *Arrow* symbols as the number of outgoing transitions of the pseudostate. Group the arcs under the *Folder* symbol with the name of the incoming transition of the pseudostate. Assign the content of the outgoing transitions target states to these *Arrow* symbols.

Associate the *Arrow*-s to their sources (make them subelements of their sources) and make visible the *Arrow*-s whose guard is satisfied or annotate the *Arrow*-s by appropriate symbol from traffic lights metaphor (red, green, and yellow).

5. Find the final state and transform it to the *Document* symbol. Associate content to it and make visible the document whose guard is satisfied or annotate the *Document* by appropriate symbol from traffic lights metaphor (red, green, and yellow).

```

...
<UML:SimpleState xmi.id = 'a53' name = 'SW Requirements'
  isSpecification = 'false'>
  <UML:ModelElement.taggedValue>
    <UML:TaggedValue xmi.id = 'a54' isSpecification = 'false'
      dataValue = '127-0-0-1-206be6:f479423af6:-7ffb'>
    <UML:TaggedValue.type>
      <UML:TagDefinition xmi.idref = 'a39' />
    </UML:TaggedValue.type>
  </UML:TaggedValue>
  <UML:TaggedValue xmi.id = 'a55' isSpecification = 'false'
    dataValue = 'Course-2/course_info1/lesson_01/sco01.htm'>
  <UML:TaggedValue.type>
    <UML:TagDefinition xmi.idref = 'a51' />
  </UML:TaggedValue.type>
</UML:ModelElement.taggedValue>
<UML:StateVertex.outgoing>
  <UML:Transition xmi.idref = 'a56' />
</UML:StateVertex.outgoing>
<UML:StateVertex.incoming>
  <UML:Transition xmi.idref = 'a52' />
</UML:StateVertex.incoming>
</UML:SimpleState>
...

```

Fig. 5. A part of the XMI document for the state diagram of the Java e-lecture: SW Requirements simple state with reference of incoming and outgoing transition.

XMI. The UML model is represented by an XMI (XML Metadata Interchange) document [13]. An XMI document is an XML document where the tags are elements from the UML metamodel. An example of the XMI document fragment for the state diagram is depicted in Fig. 5. The XMI document can be generated directly using a CASE tool (e. g. ArgoUML [1], or its commercial version Poseidon). Figure 5 depicts the SW Requirements simple state as a one of the state machine subvertexes together with the outgoing and the incoming transition references.

XSLT templates as generation rules. Initialization of the navigation map is carried out by applying functions for inserting symbols into the map structure. The script of these functions can be generated directly from the XMI document by applying an XSLT templates. We use an XSLT parser for these purposes. The XSLT templates are used to transform one xml document to another text, html or text document. We use the XSLT to transform XMI file to code fragments for filling the navigation map. An example of the XSLT template part is depicted in fig. 6.

Figure 6 depicts a fragment of a template where an arrow symbol is generated (`gLnkLnk(...)` function in the template) for a simple state. The simple state is a target of a transition (`mode="transition"`). The symbol has to be inserted under the source state of the transition. This is carried out by `levelp` parameter. The tran-

```

...
<xsl:template match="UML:SimpleState" mode="transition">
  <xsl:param name="levelp"/>
  <xsl:param name="namep"/>
  <xsl:param name="stateidp"/>
  <xsl:param name="transnamep"/>
  <xsl:variable name="fldname" select="@name"/>
  <xsl:variable name="cond" select=
    "UML:State.entry/UML:CallAction/UML:Action.script/
    UML:ActionExpression/@body"/>
  <xsl:choose>
    <xsl:when test="$cond">
      <xsl:text>if (</xsl:text><xsl:value-of select="$cond"/>
      <xsl:text>) {</xsl:text>
      <xsl:value-of select="concat('aux', $levelp+1)"/> =
        insDoc(<xsl:value-of select="concat('aux', $levelp)"/>,
        gLnkLnk(2, "<xsl:value-of select="$fldname"/>
        <xsl:if test="$transnamep='next'"/>/
        <xsl:value-of select="$transnamep"/></xsl:if>",
        "<xsl:apply-templates select=
        'UML:ModelElement.taggedValue'"/>",
        "<xsl:value-of select="$fldname"/>",
        "<xsl:if test="$visibility">green</xsl:if>",
        "<xsl:value-of select="$stateidp"/>"))
      <xsl:text>}
      </xsl:text>
      <xsl:text>else{
      </xsl:text>
      ...
    </xsl:when>
    ...
  </xsl:choose>
</xsl:template>
...

```

Fig. 6. The example of XSLT template part for transforming simple state as a target of a transition.

sition is attached under group of transitions if it is one of the outgoing transitions of pseudostate. This is again reflected in the `levelp` parameter. The name of the state is also included into arrow generation parameters as a target information fragment of the link. The `stateid` parameter is used to identify the exact position of the target information fragment in the map when user click the arrow. This allows to change the indication of current position of the user in the map. If transition name is “next” then this string is added to the target state name to indicate that this link leads to the next information fragment in planned path. The global `visibility` parameter is used to switch between the presentation options. One option is annotation with traffic lights metaphor, another option is to hide the fragments and links which do not satisfy guards.

Another possibility for parameters is to switch between the navigation map presentations. One possibility is to show the (whole) map as we presented in this paper.

When a user enters the system the first time, initial navigation map is generated. The initial generation is derived from the entry state and according to the rules for visibility and collapsing. The user profile is initialized according to the entry user assessment (entry form).

Events generated by user actions at the user interface invokes associated actions which process and store new values to a user profile. They also initiates regeneration of navigation tree according to new values from the user profile. We use a standard XSLT parser for a regeneration of navigation map. The regeneration is invoked through standard interface of that parser.

The XMI document is generated by standard export facility of Poseidon UML tool. We have implemented the XSLT templates for transforming the XMI representation to the format described in the section 4.1 (Visualization of the navigation map). We have also developed the user interface control mechanism, which interprets user actions over navigation model (tree).

5 Related work

Several XML based transformations have been introduced for the web-based information systems such as generators for WCML [19] or WebML [5]. Modelling techniques provided by these languages emphasize only structural view. Both, behavioral modelling techniques and transformation methods for the techniques to implementation environments are underestimated. We showed, that the state diagrams and their duality (structural and behavioral features) are very useful and also that we are able to use them as a base for the adaptive navigation design and construction.

Rapid prototyping of web sites based on UML models was presented in [17]. They use the use case models, activity diagrams and web site structure models to generate and prototype web sites. Our approach deal with dynamics and adaptation by employing state diagrams. It can be seen as a complementar to the approach presented in [17].

Architecture of our system is similar to the one presented in [15]. They generate HTML pages from declarative specification based on graph theory. They recognize page graph and data graph. They concentrate also on structural features of page sites and do not deal with adaptation. Our approach can be seen as a complementar.

Hypermedia and web site design methods mostly employs the structural techniques to model navigation (see for example OOHDM [18], UWE [14], WebML [5], or others reviewed in [8]). We argue that navigation is based on interaction between user and a system and thus behavioral techniques seems to be suitable for modelling the navigation. This is especially important in adaptive systems where the navigation support should be generated according to a user behavior and user model evolution. HMBS/M [7] and χ Trellis [12] are based on behavioral techniques. However, the former only allows to map states to class instances. Both approaches do not emphasize on adaptation and user modelling.

Provided guidelines for user modelling and the integration of a user model and navigation model is another advantage. User modelling is supported by UWE [14], WebML [5] and W2000 [2] but independently from navigation.

The web site architecture modelling is discussed in [6]. The approach emphasize structural point of view in web site architecture modelling. Our approach is orthogonal to the one presented in [6] and integrating these two approaches can improve both of them.

6 Conclusions and Further Work

In this paper we described an approach to adaptive navigation support (adaptive navigation sequences) modelling. Application of the UML state diagram modelling technique is the first advantage of the proposed approach. A hypermedia application reacts to user actions. Thus, it seems that state-transition diagrams are more natural for modelling navigation than structural techniques in reviewed methods.

We proposed the transformation of the UML state diagram into the navigation map (web site graph) for hypermedia applications allowing rapid prototyping and implementation of adaptive navigation sequences in the web. We have implemented the transformation method for proposed visualization. The method allow us to short the gap between the navigation design for hypermedia applications and the implementation of such a navigation. The XMI document as a representation of the UML state diagram and parametrized XSLT templates as representations of the transformation rules are means for the implementation of proposed method.

We implemented a system where the generator can be parameterized and dynamically invoked according to user actions. The generator regenerates the navigation map. The generator is able to change the presentation style of guidance for a user, i.e. from traffic light metaphor to showing and hiding fragments and links. The generator can also change the presentation style of the navigation map from whole map to the relevant parts of those maps.

We would like to concentrate on improving the system based on generator from state diagrams in the further work. We also want to improve the system to be able to specify dynamic queries in states and links what will allow us to parameterize the content selection when clicking particular link or showing particular information fragment.

References

- [1] ArgoUML CASE tool web page. Available at: <http://www.argouml.org/>.
- [2] L. Baresi, F. Garzotto, and P. Paolini. Extending UML for modeling web applications. In *Proc. of 34th Annual Hawaii International Conference on System Sciences (HICSS'34)*, Maui, Hawaii, Jan. 2001. IEEE Press.
- [3] P. D. Bra, G.-J. Houben, and H. Wu. AHAM: A dexter-based reference model for adaptive hypermedia. In K. Tochtermann, J. Westbomke, U. Wiil, and J. Leggett, editors, *Proc. of ACM Conference on Hypertext and Hypermedia*, pages 147–156, Darmstadt, Germany, Feb. 1999.
- [4] P. Brusilovsky. Adaptive hypermedia. *User Modeling and User-Adapted Interaction*, 11(1-2):87–100, 2001.
- [5] S. Ceri, P. Fraternali, and A. Bongio. Web Modeling Language (WebML): a modeling language for designing web sites. *Computer Networks and ISDN Systems*, 33(1–6):137–157, June 2000.

- [6] J. Conallen. *Building Web Applications with UML*. Addison Wesley, 2000.
- [7] M. R. de Carvalho, M. C. F. de Oliveira, and P. C. Masiero. HMBS/M - an object oriented method for hypermedia design. In *Proc. of V Brazilian Symposium on Multimedia and Hypermedia Systems - SBMIDIA'99*, pages 43–62, Goinia, June 1999.
- [8] P. Dolog and M. Bieliková. Hypermedia systems modelling framework. *Computing and Informatics*, 21(3):221–239, Dec. 2002.
- [9] P. Dolog and M. Bieliková. Navigation modelling in adaptive hypermedia. In P. D. Bra, P. Brusilovsky, and R. Conejo, editors, *Proc. of 2nd Conference on Adaptive Hypermedia and Adaptive Web-based Systems*, pages 586–591, Malaga, Spain, May 2002. Springer LNCS 2347.
- [10] P. Dolog and M. Bieliková. Towards variability modelling for reuse in hypermedia engineering. In Y. Manolopoulos and P. Návrát, editors, *Proc. of 6th East-European Conference on Advances in Databases and Information Systems — ADBIS'2002*, pages 388–400, Bratislava, Slovakia, Sept. 2002. Springer LNCS 2435.
- [11] P. Dolog and W. Nejdl. Challenges and benefits of the semantic web for user modelling. In *In Proc. of AH2003 — Workshop on Adaptive Hypermedia and Adaptive Web-Based Systems, User Modelling Conference 2003*, Pittsburgh, PA, June 2003.
- [12] R. Furuta and P. D. Stotts. A formally-defined hypertextual basis for integrating task and information, 1994. Tech. Report TAMU-HRL 94-007.
- [13] O. M. Group. OMG XML metadata interchange (XMI) specification, version 1.1, Nov. 2000. Available at <http://www.omg.org/>. Accessed on June 1, 2002.
- [14] N. Koch. Software engineering for adaptive hypermedia systems? In P. D. Bra, editor, *Proc. of Third Workshop on Adaptive Hypertext and Hypermedia, 8th International Conference on User Modeling*, July 2001.
- [15] A. Levy, D. Florescu, D. Suci, J. Kang, and M. Fernandez. Catching the boat with Strudel: experiences with a web-site management system. In *In SIGMOD'98*, 1998.
- [16] D. Petrelli, D. Baggio, and G. Pezzulo. Adaptive hypertext design environments: Putting principles into practise. In *Proc. of International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH2000)*, pages 202–213, Trento, Italy, Aug. 2000. Springer, LNCS 1892.
- [17] T. Schattkowsky and M. Lohmann. Rapid development of modular dynamic web sites using uml. In J.-M. Jézéquel, H. Hussmann, and S. Cook, editors, *In Proc. of 5th International Conference on UML 2002 - The Unified Modeling Language*, pages 336–350. Springer, LNCS 2640, Oct. 2002.
- [18] D. Schwabe and G. Rossi. An object-oriented approach to web-based application design. *Theory and Practise of Object Systems (TAPOS), Special Issue on the Internet*, 4(4):207–225, Oct. 1998.
- [19] C. Segor and M. Gaedke. Crossing the gap - from design to implementation in web-application development. In *Proc. of Information Resources Management Association International Conference 2000*, Anchorage, USA, May 2000.
- [20] P. Stevens. Small-scale xmi programming: A revolution in uml tool use? *Automated Software Engineering*, 10:7–21, 2003.