Modelling Browsing Semantics in Hypertexts Using UML

Peter Dolog*
dolog@dcs.elf.stuba.sk

Mária Bieliková*
bielik@elf.stuba.sk

Abstract: Navigation is one of the basic characteristics of a hypertext. This feature enables browsing through different paths within the hypertext document. On the other hand a problem of being lost in hyperspace can arise. One solution to this problem is improving a hypertext structure. This can be achieved by creating a model of hypertext dynamic behaviour. This paper presents our approach to modelling navigational structure of hypertext using the Unified Modelling Language (UML). Our goal is to define a unified framework for modelling hypertext browsing semantics. We specify semantics for diagrams capable of interaction and navigation modelling. An example of a university course specification is given to illustrate the use of the proposed extensions to UML techniques.

Key words: interaction, navigation, browsing semantics, hypertext modelling, UML.

1 Introduction

A hypertext modelling is a very active research area today. It is interesting mainly due to hypertext character of the Web content and an extensive use of hypertext systems in many applications. In spite of many contributions to hypertext modelling, researching framework for unified approach, which enables effective modelling of static and dynamic aspects of the hypertext, remains on the agenda.

In this paper we describe an approach of adopting the Unified Modelling Language (UML) for modelling browsing semantics of a hypertext, where dynamic properties of the hypertext in terms of reader's experience are studied. In the past few years the UML became one of the most used languages for visual modelling of complex software systems during different phases of their development. But the UML is not used only for software systems specification. The language is used also for process modelling, high level systems modelling, etc. The established and unified notation enables developers concentrate on a problem solving rather than on such issues as whether the specific kind of arrows should be drawn, or whether the class should be drawn as a circle or as a rectangle.

The UML was developed for software intensive systems modelling. Adopting the UML in other application area requires careful analysis of specific features of intended application area entities to be modelled. The significant feature of the hypertext system is a possibility of nonsequential (nonlinear) navigation between pieces of information or different documents, i.e. there is no single order that determines the sequence in which the text is to be read [3]. The user thus can browse the hypertext in many ways. In complex hypertexts the user may lose his position (the starting point, previous information read, etc.) without an appropriate help (e.g., guidance by a system).

The model can help the authors of web content to specify the possible paths through domain represented by a hypertext. The previous and next relationships can be easily derived from

---

* Department of Computer Science and Engineering, Faculty of Electrical Engineering and Information Technology, Slovak University of Technology, Ilkovičova 3, 812 19 Bratislava, Slovak Republic

the model. The starting and ending points can be also explicitly specified in the model. Moreover, the model can capture different navigational paths according a user model or context of browsing that helps with a navigational guidance.

The UML unifies techniques developed for different object-oriented software systems analysis and design methods. However, some concepts or semantics important for the specification of hypertexts are missing in the UML. Missing parts reflect the above discussed features of hypertext systems.

The goal of our research is to define a unified framework for modelling of browsing semantics, which is a crucial part of the hypertext model. The rest of the paper is organised as follows. Section 2 starts with related work. In the next section we briefly describe requirements for modelling browsing semantics in the hypertext. Section 4 is devoted to the semantics of the UML. We present our proposal of the extension for specifying the browsing semantics in the hypertext. Next, in Section 5, an example of the use of proposed techniques is illustrated by means of the case study. The paper concludes with our conclusions and some remarks for the future work.

2 Current Approaches to Hypertext Modelling

A number of methods and techniques (formal or semiformal) aimed at providing support for hypertext systems modelling have been proposed in the literature. Existing approaches consider two basic views of the modelled system. They capture information about the static structure or dynamic behaviour of a hypertext, or both. Approaches to the behaviour modelling could be divided into two groups: functional approaches and approaches based on automaton modelling. These approaches represent different views of the hypertext behaviour. A review of some known approaches to hypertext systems modelling classified according to application domain, navigational and presentation characteristics is given in [5].

The static structure defines the kinds of entities important to a system and to its realisation together with the relationships among the entities. Modelling of the static structure is based on the graph theory (formal models) or on the class or entity relationship models (semiformal models). Semiformal models visualise the structure of a document or domain being presented by document, and interconnections between contained information chunks. HDM [9], W3DT (earlier name SHDT) [2], the domain modelling techniques in OOHDM [14] belong to this category. An example of the formal model for hypertext databases using hypergraphs is presented by Tompa [16]. The model facilitates the separation of structure from the content.

Functional modelling of dynamic behaviour is based on the algebraic theory or on description of hypertext functions. In [8], the hypertext is modelled as a set of domain and information objects, a set of predicates and attributes. Tompa's structural model [16] is combined with functional modelling of behaviour by extending a hypergraph with operations.

The automaton modelling is based on the automaton theory or on the specialisation of statecharts. A hypertext is modelled as some kind of a hyperprogram. The automaton approach is adopted in the Trellis project [7, 11, 15]. The authors employed the petri net formalism for describing browsing semantics. Their model evolved from classical petri nets used in αTrellis [15] till high-level petri nets used in χTrellis [7] for collaboration protocol and in caT [11] for adaptation. Petri nets are used for analysis and specification of browsing semantics also in [4]. Another approach is based on an extension of the statechart technique with history and timing mechanism [13].

Several other approaches were proposed for navigation modelling. The authors of OOHDM proposed the method for design of navigation with navigational classes, anchors inside classes and navigational context [1]. In [10] new diagrammatic technique for specifying interaction called the user interaction diagram (UID) is proposed. It is derived from a use case diagram. The authors also define steps to derive a navigation model from the UID.

3 Requirements for Modelling Browsing Semantics

The browsing semantics serves for expressing a dynamic behaviour of a hypertext. It concentrates on a user interaction during hypertext presentation or on a change of a hypertext state when the user performs the act of navigating. The model of browsing semantics should capture dynamic behaviour of the hypertext. Due to the space limitations we discuss here only the most important requirements for techniques capable of modelling browsing semantics in the hypertext.

At first, the notion of a *hypertext state* should be determined. The state of a hypertext is represented by the content presented to a user. The state can be atomic or composite. An example of a composite state is a web page where the user can navigate between individual parts of this page. Firing one or more *transitions* changes the state. Thus, we need techniques for modelling transitions and for describing the type of event(s) which fire transitions in a hypertext. The state is changed when a link is clicked or the mouse pointed to a predefined area, or another mechanism for navigating in the hypertext is fired (e.g., next or previous relationship).

Another requirement is capability of keeping history of the previous states. The following situation can arise: a user opens the document and he wants get into the state in which the system was during the user's last visit or even the older state. Important point is the ability to relate the conditions or operations to the transition, or even more relate conditions to a particular state, or before/after the state change. Considering hypermedia, the parallel states, their synchronisation, forking and branching should be possible in the specification language.

The question is to what extend are the mentioned requirements covered by known approaches. Some of them were mentioned in the previous section. In our opinion, existing methods and techniques cover only the subset of the mentioned requirements. The problem is also with a number of proposed different notations. We show some techniques defined in the UML that are suitable for modelling browsing semantics in a hypertext. The UML and especially its subset – *State Machines*, are with minor extensions of semantics capable to cover all of the mentioned requirements.

4 Extension of UML Semantics

The UML provides mechanisms for specifying the static structure and dynamic behaviour of a modelled system. The UML also contains organisational constructs for arranging models into packages that permit partition of a large system into smaller pieces. As we mentioned in the previous sections, we express browsing semantics by behaviour modelling. In order to express the required semantics, we adopted the approach taken from [12]. It is based on the specification of semantics for structural and behavioural object models provided by the UML using metamodelling. UML semantics is described in semiformal way, using the class diagram of metaclasses and a textual description.

The metamodeling of web pages or web systems is not a new topic. W3DT is built on the author's view of a web site metamodel [2]. In [6], the metamodel for domain modelling, navigational modelling and presentation modelling is proposed. Both approaches present the own view of the authors and they do not contain a metamodel for managing models of browsing semantics.

The architecture of the UML is based on a four-layer metamodel structure, which consists of the following layers: user objects, model, metamodel, and meta-metamodel. The meta-metamodeling layer defines the language for describing a metamodel (elements are for example metaclass, metaattribute and metaoperation). The metamodel layer is an instance of a meta-metamodel. It defines the language for describing model (the elements are for example class, attribute, operation and component). The model layer is an instance of a metamodel. It defines the language that describes an information domain (the elements are concrete named classes, for example opportunity, user and course). The user objects layer is an instance of a model. The user objects layer describes a specific information domain (the elements are concrete instances of named classes – objects) [12].

We are interested in the language for modelling. As a result we use for expressing extension of semantics the language from meta-metamodel layer. Because the semantics is instantiated on the metamodel layer, we extend the metamodel.

The complexity of the UML metamodel is managed by organizing it into logical packages. Behavioural features of a modelled system are expressed in the UML by the *Behavioural Elements* package. This package consists of the following subpackages: *Collaborations*, *Use Cases*, *State Machines* and *Activity Graphs*. The *Activity Graphs* subpackage is derived from the *State Machines* subpackage (it is a special case of state machines). The *Collaborations* subpackage enables us to model collaboration of the model elements in time or by invoking call or event. Thus, the *Collaborations* subpackage is also related to the *State machines* subpackage. On the other hand the *Use Cases* subpackage enables us to model the services, which the system will provide to a user and the relationship between services and the user.

All of the mentioned techniques could be used to model the browsing semantics but in different point of view. The use cases may be used in the phase of requirement definition for the hypertext document. Examples of using this type of diagrams are given in [10].

The sequence or collaboration diagrams (the *Collaborations* subpackage) could be used to express the interaction of a user with the hypertext. However these diagrams provide only techniques for modelling single roles, which are interconnected by arcs with meaning of a message. This is sufficient only for modelling simple hypertexts or for modelling the fraction of a complex system. It is not suitable for specifying the browsing semantics for large and complex domains. Example of such diagram is depicted in Fig. 1.

As you can see in Fig. 1, the user ("*student*" role) interacts with the hypertext presenting a university course. The student enters the course ("*Enter Course*" link). To be able to enter the course, the student should have some prerequisite knowledge. The system generates a test ("*Generate Test*" link) to establish the student's level of knowledge. The student has to complete a test ("*Filling Test*" link). However, this activity could be complex and could be composed from several other subactivities, which generate another events or signals. If we enclosed all mentioned activities into one diagram, the result would be unreadable. The same problem arises with the "*Browsing*" link. The complexity is marked by the «*user activity*» stereotype. A particular user activity could be specified separately by the activity diagram. There are also "*Open Theme*" and "*Generate Test*" links, which are interconnected with *xor* constraint with the following meaning: if a user passes the test, he can enter the course, if the user fails, new test is generated.

The *State Machines* subpackage contains several mechanisms for expressing requirements stated in Section 3. For expressing the current state of a hypertext, the *State Machines* subpackage provides metaclass *StateVertex* with its submetaclasses (e.g., State, Pseudostate). The *Transition* metaclass serves for expressing the traversal mechanism in the hypertext. The *Transition* metaclass could be forked or joined with one of the stereotypes of

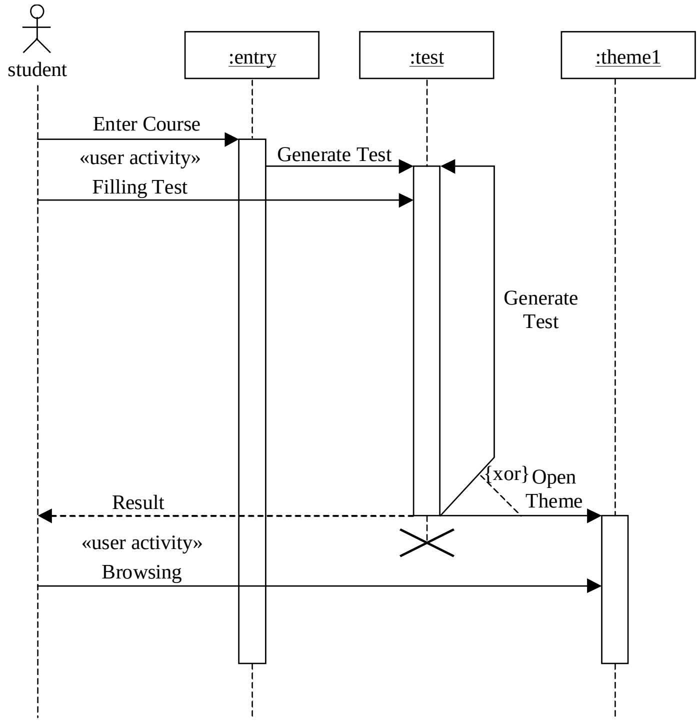

Figure 1. Example of sequence diagram as interaction between user and hypertext.

the *Pseudostate* metaclass. The *CompositeState* metaclass represents the state, which could be decomposed into several atomic or composite states. The transition could have associated a condition. The *Guard* metaclass is provided by the UML for these purposes. The package contains the construct for specification the history (the *deepHistory* or the *shallowHistory* stereotypes of the *Pseudostate* metaclass). The complete description of semantics with Well-Formedness Rules could be find in [12].

The state machine has to have defined events or types of events, which are capable to fire transitions. The UML provides the events for the O-O systems modelling. The events for the hypertext navigation are slightly different (comparing for example with *SignalEvent* or *CallEvent* defined in the UML). Thus we should extend *Events* part of the *State Machines* subpackage. One approach for extending semantics for an event is to add the stereotype for the *SignalEvent* metaclass. The signal event is raised by a send action. Clicking a link means that the web browser generates the send action, which causes the signal to the web server for getting appropriate content to the web browser. Because the send action is not directly stated but generated by clicking the link, we sign the event as a new stereotype to distinguish it from the signal event caused by the behavioural feature of a metaclass instance. We can sign this stereotype as «*GetContentEvent*».

Another possibility is to add a new submetaclass for the *Event* metaclass. The diagram with added metaclass (with the same name as the *GetContentEvent* stereotype) is illustrated in Fig. 2. For simplicity, some elements are omitted. The full diagram without our extension can be found in [12].

UML semantics is extended with the *HyperLink* metaclass, which characterises the hyperlink that caused the *GetContent* event. This represents rather structural semantics and thus discussing it is out of the scope of this paper. We prefer the adding metaclass approach, because the semantics is described better with adding the metaclasses with associations to the metamodel.

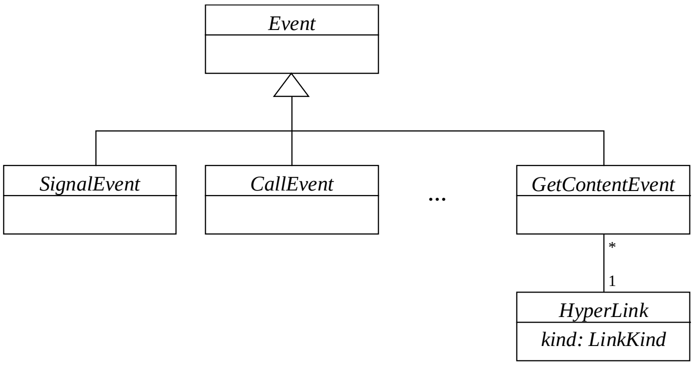

Figure 2. Events in the *State Machines* subpackage in the UML.

5 Case Study – a University Course

In this section we give an example of browsing semantics modelling in hypertext using statechart diagrams. The case study is based on a university course web site (the Knowledge-based systems master course at the Slovak University of Technology).

The first step in development of a statechart is to define a set of hypertext document states. The states are represented by information chunks contained in a hypertext and presented to a user. In our example, we consider only a few states for simplicity. There are the *Caution*, *Content*, *Tasks*, *Conditions*, and *References* states. As can be seen in Fig. 3, they belong to the composite state called *KBS*. There is also the *How to read* state.

The initial (pseudo) state is shown as a small filled circle with no incoming arcs. The *Caution* state represents the part of course web page, where the caution for students is stated. The caution could contain for example a warning that the page does not contain all information needed for passing an exam. The *Content* state is a concurrent state with the *Caution* state. Its main purpose is to summarise the content of the whole web site. The *Tasks* state represents information about assignments for the current term. The *Conditions* state represents information about conditions for passing the course. The *References* state represents the literature or references to relevant sources on the Internet. Finally, the *How to read* state represents instructions about a way of reading and navigating in this site. The internal final (pseudo) state in the *KBS* state is shown as a small circle surrounding a filled circle.

The second step in a statechart development is to determine possible transitions in the system. A transition is shown as a directed arc. The model contains labelled and non-labelled transitions. Labelled transitions represent named links, which occur in the content of a particular state. Clicking the next link in our example causes transition to the internal final

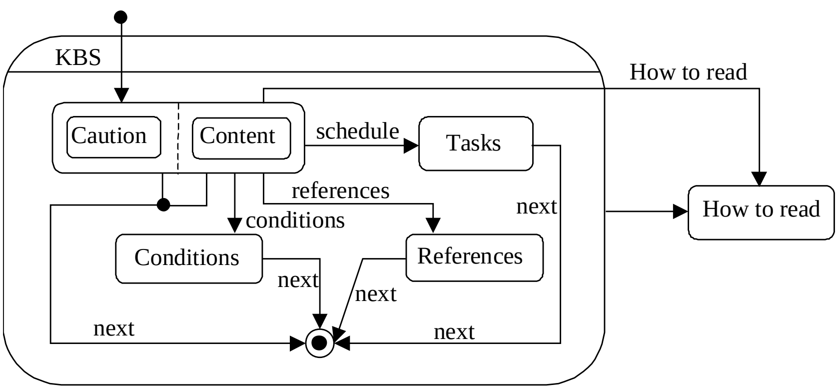

Figure 3. Statechart diagram of the Knowledge-based systems course.

state. The previous link in our example is implicit. It means that we can go back to the previous state through the same path as we came into the current state.

A non-labelled transition is used for example for modelling a transition from the composite state (if a final state within the composite state is reached). The example of a non-labelled transition is the transition, which interconnects the composite state *KBS*, and the *How to read* state. The *How to read* state can be reached from an arbitrary state contained in the composite state *KBS*. However, the transition is allowed only in the case of reaching the final internal state. This step of a statechart development could be structured into subactivities for determining internal transitions, transitions from the initial and into the final state and transitions for interconnecting the composite states. The next subactivity is determining whether a transition is labelled or non-labelled.

The *Content* part of the web site usually contains more links. For simplicity, we consider only links to explicitly modelled internal substates of the *KBS* state and one external link to the *How to read* state.

The use of the state machine view of a hypertext enables stepwise refinement of the model. Fig. 4 depicts the high-level statechart diagram of our example (see Fig. 3). The composite state is marked with hidden decomposition indicator. The stubbed transition is also depicted. It represents the transition to additional state, which is not visible in the diagram.

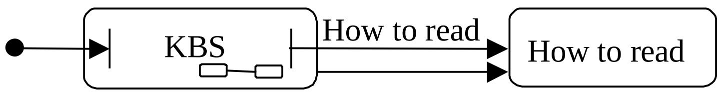

Figure 4. The high-level statechart diagram.

The example of web page according to the proposed statechart is given in Fig. 5 (the content is written in the Slovak language). Note that the model does not specify the graphical appearance of the hypertext document.

6 Conclusions

In this paper we proposed the techniques, which can be taken from the UML and applied to modelling browsing semantics in a hypertext. We extended the *State Machine* subpackage with the mechanism for expressing the events, which cause the transition between states in the *State Machine* subpackage (the *GetContentEvent* metaclass). Proposed extension enables also expressing elements that provide specific events when manipulated (association with the *HyperLink* metaclass). To illustrate proposed extension we provided simple example of a university course. We presented possible model abstraction by hiding details of substates and showing only high level states and transitions. It is clear that the method for hypertext

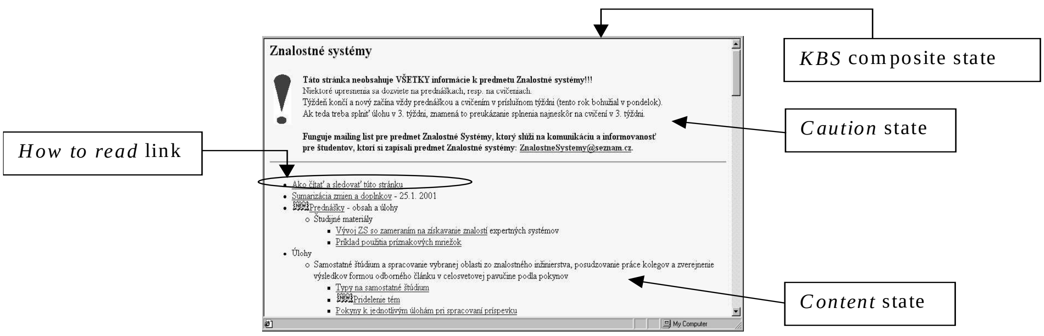

Figure 5. Knowledge-based systems course web page.

authoring should determine techniques for modelling of all mentioned views of the system – structural and behavioural. Existing approaches use different formalisms for modelling. Our aim is to unify the techniques similarly as in software engineering.

Described approach to hypertext modelling enables specification of adaptivity characteristics of the hypertext using for example guards for transitions in a statechart. Open question is using the synchronisation features of the UML for specification of hypermedia systems. Our future work will concentrate on extension of current authoring tools with the ability to enclose the proposed techniques and thus simplify the process of hypermedia constructing. Adding the visualisation of the model into a presentation can also improve a user orientation in the hypertext.

This work was partially supported by Slovak Science Grant Agency, grant No. G1/7611/20.

References

1. Barbosa, S., Schwabe, D.: *Navigation modeling in hypermedia applications*. Tech. report, Rio de Janeiro, PUC-Rio, Departamento de Informática, 1994.
2. Bichler, M., Nusser, S.: Developing structured WWW-sites with W3DT. In *Proc. of WebNet'96 World Conference of the Web Society*, H. Maurer (Ed.), San Francisco, CA 1996, pp. 7-12.
3. Bieliková, M., Návrat, P.: Modelling Versioned Hypertext Documents. In *Proc. of System Configuration Modelling Symposium*, B. Magnusson (Ed.), Springer, Brussels 1998, pp.188-197.
4. De Bra, P., Houben, G.J., Kornatzky, Y.: A Formal Approach to Analyzing the Browsing Semantics of Hypertext. In *Proc. of CSN-94 Conference*, 1994, pp. 78-89.
5. Dolog, P.: Aspects of Hypermedia Modelling. In *Proc. of 3rd Conf. on Electrical Engineering and Information Technology for PhD Students*, Bratislava, September 2000.
6. Frolich, P., Henze, N., Nejdl, W.: Meta Modeling for Hypermedia Design. In *Proc. of the 2nd IEEE Metadata Conference*, Silver Spring, Maryland USA, September 1997.
7. Furuta, R., Stotts, P.D.: *Trellis: A Formally-defined Hypertextual Basis for Integrating Task and Information*. Tech. Report TAMU-HRL 94-007, 1994.
8. Garg, P.K.: Abstraction Mechanisms in Hypertext. *Communications of the ACM*, Vol. 31, No. 7, 862-870, July 1988.
9. Garzotto, F., Paolini, P., Schwabe, D.: HDM – A Model-Based Approach to Hypertext Application Design. *ACM Transactions on Information Systems*, Vol. 11, No. 1, 1-26, January 1993.
10. Guell, N., Schwabe, D., Vilain, P.: Modeling Interaction and Navigation in Web Applications. In *Proc. of the WWW and Conceptual Modeling Workshop*, Springer, Salt Lake City 2000.
11. Na, J.C., Furuta, R.: Context-Aware Hypermedia in Dynamically-Changing Environment Supported by a High-Level Petri Net. In *Proc. of 11th ACM Conf. on Hypertext and Hypermedia*, San Antonio, Texas, USA, June 2000.
12. Object Management Group, Inc.: *OMG Unified Modeling Language Specification*. Version 1.3, OMG, 1034p., March 2000.
13. Paulo, F.B., Turine, M.A., de Oliviera, M.C., Masiero, P.C.: XHMBS: A formal model to support hypermedia specification. In *Proc. of HyperText 98*, Pittsburgh 1998, pp. 161-170.
14. Schwabe, D., Rossi, G.: The Object-Oriented Hypermedia Design Model. *Communication of the ACM*, Vol. 38, No. 8, 45-46, August 1995.
15. Stotts, P.D., Furuta, R.: Petri-Net-Based Hypertext: Document Structure with Browsing Semantics. *ACM Transactions on Information Systems*, Vol. 7, No. 1, 3-29, January 1989.
16. Tompa, F.WM.: A Data Model for Flexible Hypertext Systems. *ACM Transactions on Information Systems*, Vol.7, No. 1, 85-100, January 1989.