Navigation Modelling in Adaptive Hypermedia*

Peter Dolog and Mária Bieliková

Department of Computer Science and Engineering
Slovak University of Technology
Ilkovičova 3, 812 19 Bratislava, Slovakia,
{dolog, bielikova}@dcs.elf.stuba.sk,
http://www.dcs.elf.stuba.sk/~{dologp,bielik}

Abstract. In this paper we reflect the need for modelling in a systematic production of adaptive hypermedia applications. Proposed approach is based on the Unified Modelling Language (UML). State diagrams are used to model possible paths through hypertext. The user model expressed by a class diagram determines structural and behavioural features, which are used for specification of adaptations in states and transitions contained in state diagrams.

1 Introduction

One of the main goals of any adaptive hypermedia application is to increase user efficiency measured either in the time spent searching for information or the amount of information absorbed by the user. Another important issue is to aid developers of such systems which are going to be more and more complex. The increased complexity of hypermedia applications raises the need to employ modelling in hypermedia development process.

The modelling of a hypermedia is extensively studied only in past two decades. Models help us understand developed system by simplifying some of details. Adaptation of navigation, together with a user model, should be addressed in hypermedia application modelling. The goal of this paper is to present an approach to modelling adaptive navigation. The Unified Modelling Language, namely state diagrams together with sequence and class diagrams are employed for these purposes.

2 Process of Adaptive Navigation Modelling

A navigation model represents 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.

Basic modelling technique for navigation modelling in our approach is a state diagram, which enables to model dynamic character of navigation. We proposed

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

five basic steps of navigation modelling process: identifying basic interaction scheme, identifying states, identifying transitions, identifying events, 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).

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 [12]. The user model is derived from user roles. Structure of the user model follows the well known Adaptive Hypermedia Application Model (AHAM) [6]. 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.) can be incorporated when it is needed [9]. The user model contains operations for reading the current state of user characteristics and for updating user characteristics. Environment or context data [2] are carried during mapping to navigation model likewise data in the user model.

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.

II. Identifying states. States in a navigation model fulfil the role of information chunks [8]. They can be grouped into superstates. The states are created from an information model. There are two possibilities of mapping: (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 synchronisation 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 generalisation/specialisation tree. An event can be mapped to more than one transition.

V. Mapping user model elements to state diagram. The adaptive behaviour is modelled by an introduction of features of user model classes into state diagrams. Accessible attributes of user model classes are mapped to guards conditions. They 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 Modelling of Techniques for Adaptive Hypermedia

Several efficient techniques for adaptive hypermedia were proposed [3]. We selected some of these techniques for presentation of capabilities of the proposed approach. The examples cover both link-level and content-level adaptation.

Fig. 1 depicts part of an adaptive navigation model, which was created according to the approach proposed in this paper. The example figures the model of a lecture on functional programming (FP). This lecture consists of four topics: `Functional Programming`, `Programming Schemes`, `Examples of Linear Lists Processing` and `Examples of Non-Linear Lists Processing`. The aim of such lecture is to exercise programming of basic list processing functions in the Common Lisp language. First, some introduction is needed. This is carried out by `Introduction to FP`, `Computation in FP`, and `Introduction to LISP` fragments. Next, the introduction to Programming schemes (`Programming Schemes` state) is performed. It is represented by `Introduction` substate and simple categories of `Linear Lists Processing` and `Non-Linear Lists Processing`.

Adaptation rules are involved in transition labels or as internal transitions of states. Events handle user interaction or internal system events. Conditions and actions are taken from the user model.

Conditional text is modelled by *Entry* internal transition of a state. It is followed by a condition, which determines whether the fragment is displayed or not. In the Fig. 1, conditional text is represented by the `Introduction to LISP`

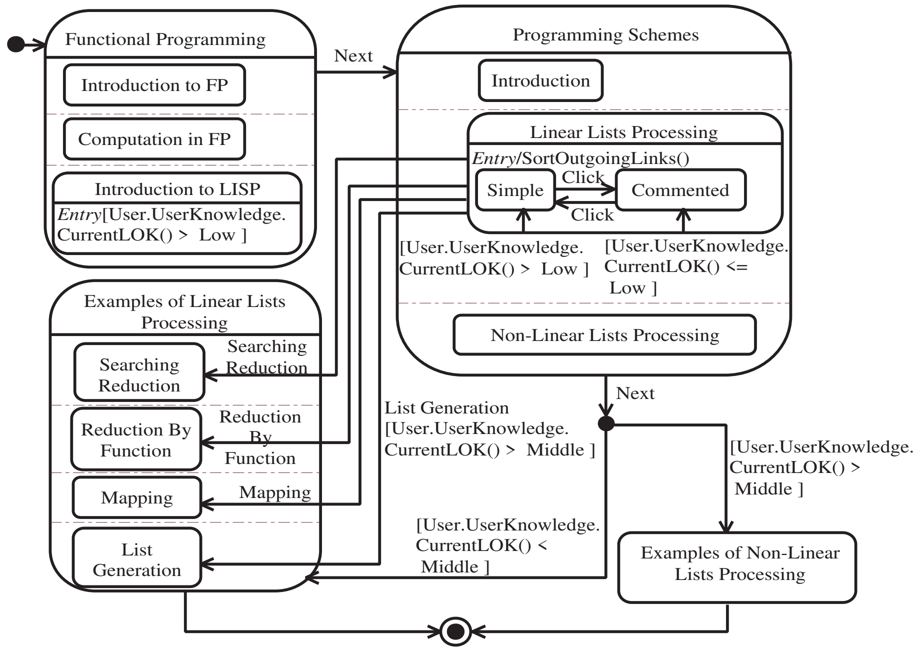

Fig. 1. A part of an adaptive navigation model.

state. There is the condition, which examines the level of current user knowledge. The fragment is displayed only if the level of knowledge is greater than `Low`.

Stretchtext can be modelled by two approaches. The first is to represent stretched and unstretched text as two *different alternative states* with transition between them. The transition has associated `Click` event. The second approach is to model stretchtext with *two parallel states*, where one state is conditionally constrained. It means that the state, which is conditionally constrained is presented only if the condition is satisfied. Condition is mostly based on a history (usage data). In the Fig. 1, stretchtext is represented in the `Linear List Processing` state containing `Simple` and `Commented` version of introduction. The text is displayed unstretched if current user level of knowledge is greater than `Low`. Otherwise, `Commented` version is displayed. These two alternatives can be switched by clicking (`Click` event).

Commented text is modelled similarly. The difference between commented text and stretchtext is that stretchtext can be clicked and thus unstretched.

Alternative pages or fragments are modelled by conditional transition split. The decision symbol (junction) represents modelling element for alternation. Another possibility is to employ the diamond symbol. In the Fig. 1, there are two alternatives when the next event occurs. If current level of user knowledge is less than `Middle` only the `Examples of Linear Lists Processing` state is displayed. Otherwise, `Examples of Non-Linear Lists Processing` as more challenging task is provided by adaptive hypermedia application.

Direct guidance can be modelled by assigning the `next` or `back` event to particular transition. Another approach is to use directly state diagram as a guide, which is interpreted and displayed as a map and the current position and allowed links are sufficiently indicated (for example like marking in petri nets).

Hiding of a link is modelled by guard condition of a transition. When the condition is not satisfied, link is not enabled (not displayed). The transition in Fig. 1 labelled by `List Generation` event is an example of such a link. The link is displayed (transition is allowed) only if current user level of knowledge is greater than `Middle`.

Link sorting is an operation. It is obviously performed when a user reached particular fragment, where the links have to be sorted. Such operation can be modelled as the `Entry` action of particular state. Fig. 1 depicts `Linear List Processing` state, which represents such fragment. The `SortOutgoingLinks()` action is invoked when the user reached this state.

The state/transition diagram can be used also as a model of *adaptive map*. Special manipulation function can be provided, which interprets meta-model and rules of states, and according to them displays particular part of the model to the user.

4 Related work and conclusions

In this paper we described an approach to adaptive hypermedia modelling. Application of a state/transition 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 such as in OOHDM [14], UHDM [12] or WebML [4] (for a review of others, see for example [7]).

Provided guidelines for user modelling and the integration of a user model and navigation model is another advantage. User modelling is supported by UHDM [12], WebML [4] and W2000 [1] but independently from navigation. According to [10] the adaptation specification can be involved as slice or slice relationship condition. But slices and their relationships do not satisfactorily deal with interactions. The authors do not explicitly discuss relationships between user model and presentation or application model. In [13] the graph formalism is employed for modelling paths. Adaptation is specified as text composition templates with linguistic rules.

HMBS/M [5] and χTrellis [11] are based on behavioural techniques. However, the former only allows to map states to class instances. Both approaches do not emphasize on adaptation and user modelling.

Our approach can be used for modelling known techniques of adaptive navigation and presentation. Our further research is oriented to extension of proposed approach to support implementation modelling.

References

[1] Luciano Baresi, Franca Garzotto, and Paolo Paolini. Extending UML for modeling web applications. In *Proc. of 34th Anual Hawaii International Conference on System Sciences (HICSS'34)*, Maui, Hawai, January 2001. IEEE Press.

[2] Mária Bieliková. Adaptive presentation of evolving information using XML. In T. Okamoto, R. Hartley, Kinshuk, and J.P. Klus, editors, *Proc. of IEEE International Conference of Advanced Learning Technologies (ICALT'2001)*, pages 193–196, Madison, USA, August 2001. IEEE Press.

[3] Peter Brusilovsky. Methods and techniques of adaptive hypermedia. *User Modeling and User-Adapted Interaction*, 6(2-3):87–129, 1996.

[4] Stefano Ceri, Piero Fraternali, and Aldo Bongio. Web Modeling Language (WebML): a modeling language for designing web sites. *Computer Networks and ISDN Systems*, 33(1–6):137–157, June 2000.

[5] Marcia Regina de Carvalho, Maria Cristina Ferreira de Oliveira, and Paulo Cesar Masiero. HMBS/M - an object oriented method for hypermedia design. In *Proc. of Brazilian Symposium on Multimedia and Hypermedia Systems (SBMIDIA'99)*, pages 43–62, Goiânia, June 1999.

[6] Paul De Bra, Geert-Jan Houben, and Hongjing Wu. AHAM: A dexter-based reference model for adaptive hypermedia. In K. Tochtermann, J. Westbomke, U.K. Wiil, and J. Leggett, editors, *Proc. of ACM Conference on Hypertext and Hypermedia*, pages 147–156, Darmstadt, Germany, February 1999.

[7] Peter Dolog. Modelling in hypermedia development, August 2001. Technical Report (A Written Part of PhD Examination). Department of Computer Science and Engineering, Slovak University of Technology.

[8] Peter Dolog and Mária Bieliková. Modelling browsing semantics in hypertexts using UML. In J. Zendulka, editor, *Proc. of ISM'2001 - Information Systems Modelling*, pages 181–188, Hradec nad Moravicí, Czech Republic, May 2001.

[9] Peter Dolog and Mária Bieliková. Hypermedia modelling using UML. In *Proc. of ISM'2002 - Information Systems Modelling*, Rožnov pod Radhoštěm, Czech Republic, April 2002.

[10] Flavius Frasincar, Geert Jan Houben, and Richard Vdovjak. A RMM-based methodology for hypermedia presentation design. In A. Caplinskas and J. Eder, editors, *Proc. of ADBIS 2001 - Advances in Databases and Information Systems*, pages 323–337, Vilnius, Lithuania, September 2001. Springer, LNCS 2151.

[11] Richard Furuta and P. David Stotts. A formally-defined hypertextual basis for integrating task and information, 1994. Tech. Report TAMU-HRL 94-007.

[12] Nora Koch. Software engineering for adaptive hypermedia systems? In Paul De Bra, editor, *Proc. of Third Workshop on Adaptive Hypertext and Hypermedia, 8th International Conference on User Modeling*, July 2001.

[13] Daniela Petrelli, Daniele Baggio, and Giovanni Pezzulo. Adaptive hypertext design environments: Putting principles into practise. In *Proc. of International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH'2000)*, pages 202–213, Trento, Italy, August 2000. Springer, LNCS 1892.

[14] Daniel Schwabe and Gustavo 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, October 1998.