Modelling Adaptivity with Aspects*

Hubert Baumeister[1], Alexander Knapp[1], Nora Koch[1,2], and Gefei Zhang[1]

[1] Ludwig-Maximilians-Universität München
{baumeist, knapp, kochn, zhangg}@pst.ifi.lmu.de
[2] F.A.S.T. GmbH, Germany
koch@fast.de

Abstract. Modelling adaptive Web applications is a difficult and complex task. Usually, the development of general system functionality and context adaptation is intertwined. However, adaptivity is a cross-cutting concern of an adaptive Web application, and thus is naturally viewed as an aspect. Using aspect-oriented modelling techniques from the very beginning in the design of adaptive Web applications we achieve a systematic separation of general system functionality and context adaptation. We show the benefits of this approach by making navigation adaptive.

1 Introduction

Adaptive Web applications are an alternative to the traditional "one-size-fits-all" approach in the development of Web systems. An adaptive Web application provides more appropriate pages to the user by being aware of user or context properties. Modelling adaptive Web applications is a difficult task because general system functionality aspects and adaptation aspects are tightly interwoven. We propose to view adaptivity as a cross-cutting concern and thus to use aspect-oriented modelling techniques to the modelling of adaptive Web applications. The advantages of using aspects when modelling adaptation is the removal of redundant modelling information, the increase in maintainability of models, and the better modularity of designs by grouping interrelated facets [9]. In particular, aspects make explicit where and how adaptivity interacts with the functional features of the Web application. Building less redundant models has the additional advantage of bug reduction in an implementation based on these models.

We demonstrate how aspect-oriented modelling techniques for adaptivity can be used to specify common types of adaptive navigation. In particular, we present aspects for adaptive link hiding, adaptive link annotation and adaptive link generation [5]. This demonstration is done in the context of the UML-based Web Engineering (UWE [13]) method and the SmexWeb (Student Modelled Exercising on the Web [1]) framework for adaptive Web-based systems which has been developed at the University of Munich.[1]

* This research has been partially sponsored by the EC 5th Framework project AGILE (IST-2001-32747) and Deutsche Forschungsgemeinschaft (DFG) within the project MAEWA (WI 841/7-1)

[1] http://smexweb.pst.informatik.uni-muenchen.de

The remainder of this paper is structured as follows: We first provide an overview of UWE and of adaptivity in Web applications. Next, we present our approach to modelling adaptivity with aspects. As a running example, we use a SmexWeb instance that implements a lesson on the topic of EBNF (Extended Backus-Naur Form). We conclude with a discussion of related work and some remarks on future research.

2 UML-Based Web Engineering

Separate modelling of Web application concerns is a main feature of UML-based Web Engineering (UWE) as well as of other Web engineering methods. Thus, different models are built for each point of view: the content, the navigation structure, the business processes, and the presentation. The distinguishing feature of UWE is its UML compliance [15] since UWE is defined in the form of a UML profile and an extension of the UML metamodel (for more details, see [12,13]).

In UWE, the content of Web applications is modelled in a conceptual model where the classes of the objects that will be used in the Web application are represented by instances of ≪conceptual class≫ which is a subclass of the UML Class. Relationships between contents are modelled by UML associations between conceptual classes.

The navigation model is based on the conceptual model and represents the navigation paths of the Web application being modelled. A ≪navigation class≫ represents a navigable node in the Web application and is associated to a conceptual class containing the information of the node. Navigation paths are represented by associations: An association between two navigation nodes represents a direct link between them. Additional navigation nodes are access primitives used to reach multiple navigation nodes (≪index≫ and ≪guided tour≫) or a selection of items (≪query≫). Alternative navigation paths are modelled by ≪menu≫s.

A navigation model can be enriched by the results of the process modelling which deals with the business process logic of a Web application and takes place in the process model. The presentation model is used to sketch the layout of the Web pages associated to the navigation nodes.

Figure 1 shows a simplified navigation model of our SmexWeb example [1]: an EBNF lesson, in which a grammar for constructing mountains is developed. After an introductory session, the user can select among three alternatives: “recognising mountains” or “building mountains”, or to solve directly. First the user interactively solves the exercise with the support of the system. Subsequently, the user can apply his EBNF knowledge to solve a similar exercise without support.

3 Adaptivity

Adaptive Web applications allow for personalisation and contextualisation of Web systems. An adaptive Web application provides more appropriate pages to the user by being aware of user or context properties. User properties are characteristics such as tasks, knowledge, background, preferences or user’s interests. Context properties are those related to the environment and not to the users themselves comprising both, user location (place and time) and user platform (hardware, software, network bandwidth). These

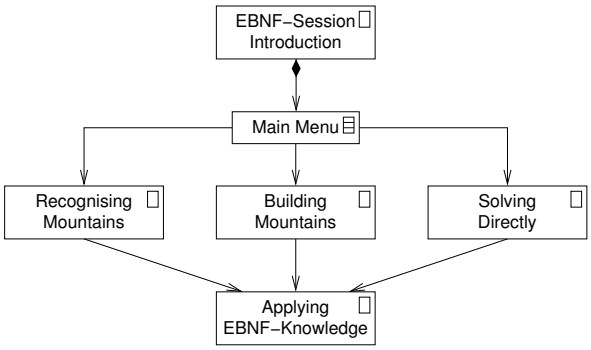


Fig. 1. SmexWeb: Navigation model for the EBNF lesson

properties are kept in a user model or context model, which is continuously updated based on the observation the system makes of the user behaviour or the environment, or on modifications of the user or context profile explicitly performed by the user.

We distinguish three levels of adaptation [12]: content adaptation, link or navigation adaptation and presentation adaptation. Others, like Brusilovsky [5], distinguish only between content and link level adaptation, where content adaptation refers as well to changes in the layout as to differences in the contents. Adaptive content comprises text adaptation and multimedia adaptation, whereas well known techniques for text adaptation are inserting/removing/altering of fragments, stretchtext and dimming fragments. Techniques to implement adaptive presentation are modality (selection, e.g., between written text or audio), multi-language (text translation into another language) and layout variations (e.g., resizing of images and ordering of text fragments or multimedia elements). Adaptive navigation support is achieved by adaptive link ordering, adaptive link hiding/removing/disabling, adaptive link annotation and adaptive link generation. The direct guidance and map adaptation techniques proposed by Brusilovsky [4,5] can be implemented by the adaptation link techniques mentioned above: ordering, annotation, hiding and generation; thus we limit the description to those techniques:

- *adaptive link ordering* is the technique of sorting a group of links belonging to a particular navigation node. The criteria used for sorting are given by the current values of the user model: e.g., the closer to the top of the list, the more relevant the link is. Sorted links are only applicable to non-contextual links and are useful in information retrieval applications, but they can disorient the user as the link list may change each time the user enters the page.
- *adaptive link annotation* consists of the augmentation of the links with textual comments or graphical icons, which provide the user with additional information about the current state of the nodes behind the annotated links. Link annotation is a helpful and frequently used technique, also used for user-independent annotation.
- *adaptive link hiding* can be easily implemented by removing, disabling or hiding of links. All three techniques reduce the cognitive-overload of the user with the advantage of more stable nodes when links are added incrementally.

– *adaptive link generation* is a runtime feature for adding new anchors for links to a node based on the current status of the user model. The typical implementation are user model dependent shortcuts.

In our SmexWeb example we use the techniques of adaptive link annotation, adaptive link ordering and adaptive link hiding for link adaptation. Depending on the acquired knowledge, the user skills and the estimated cognitive abilities of the user—contained in the current user model (see below)—the system offers different links, sorted in a different way and differently annotated with emoticons.

The current user model for the EBNF course comprises three sub-models: domain, navigation and individual model. Values of the domain model represent the learner's knowledge about the topic of the course. The most important attribute of the navigation model captures the learner's navigation behaviour. The individual model represents learning preferences, for instance with brief or extended explanations, more abstract or more pragmatic descriptions. The initial values of all sub-models are assigned on basis of the answers to the initial questionnaire the user has to go through before starting the lesson. From there on, values of the user model will change dynamically according to the user's behaviour while navigating or solving an exercise. For further details about the user modelling techniques implemented in SmexWeb see [12,1].

4 Modelling Adaptivity with Aspects

Adaptivity often cross-cuts the main functionalities of a Web application, e.g., counting the visits to a link and presenting the link differently depending on the number of visits are orthogonal to the navigation structure. It is not desirable to model such cross-cutting features in the principal model for each affected model element separately, since this would lead to redundancy and make the models error-prone. Instead, the techniques of aspect-oriented modelling [9] should be used.

Aspect-oriented modelling is a maturing modelling paradigm which aims at *quantification* and *obliviousness* by introducing a new construct: *aspect*. By quantification model elements of a cross-cutting feature are selected and comprised in an aspect and thus redundancy can be reduced; obliviousness means that the principal model does not need to be aware of the existence of the aspects and provides thus a better separation of concerns.

An aspect consists of a *pointcut* part and an *advice* part. It is a (graphical) statement saying that additionally to the features specified in the principal model, each model element selected by the pointcut also has the features specified by the advice. In other words, a complete description including both general system functionality and additional, cross-cutting features of the quantified model elements is given by the composition of the principal model and the aspect. The process of composition is called *weaving*. Building on a simple extension of the UML by aspects, we illustrate the use of aspect-oriented modelling for modelling adaptivity in our running example SmexWeb.

4.1 Aspects in the Unified Modeling Language

The UML [15] does not show genuine support for aspect-oriented modelling. In fact, several proposals have been made for integrating aspect orientation with the object-

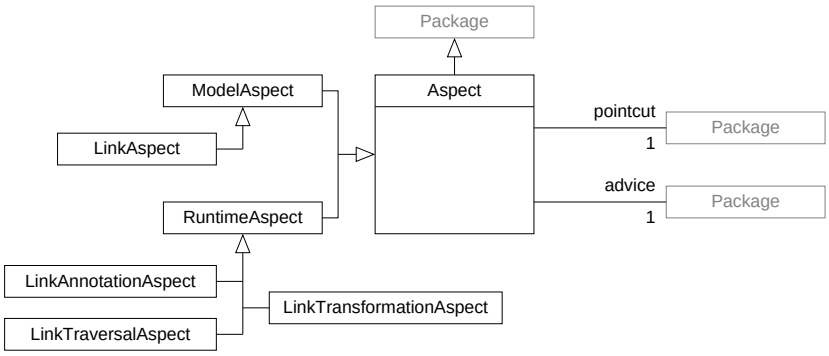


Fig. 2. Extension of UML metamodel

oriented paradigm followed by the UML, ranging from representing the programming language features of AspectJ in the UML [16] to integrating aspects in UML 2.0 as components [3], for an overview see, e.g., [9].

We restrict ourselves to a rather lightweight extension of the UML that merely composes the main ingredients of aspect-oriented modelling into a subclass (stereotype ≪aspect≫) of the UML metaclass Package: pointcut and advice; see Fig. 2. The pointcut package comprises (references to) all model elements on whose occurrence the advice package is to be applied. Both packages may contain constraints that either detail the application condition or the effect of an aspect. The semantics of applying an advice on a pointcut depends on whether an aspect is to be woven statically (≪model aspect≫) at the model level or dynamically (≪runtime aspect≫) at runtime. The different kinds of aspects we use for modelling navigation adaptation are discussed in the subsequent sections.

4.2 Extension of the UWE Metamodel

In order to capture adaptive link ordering, link annotation, and link hiding, we extend the UWE metamodel for designing navigation structures of Web applications by a NavigationAnnotation metaclass, see Fig. 3. In navigation structure models, navigation annotations thus can be attached to any navigation link. The details of how to represent the annotation, ordering or hiding in a specific Web application can thus be deferred to the choice of the designer.

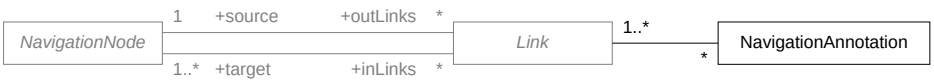
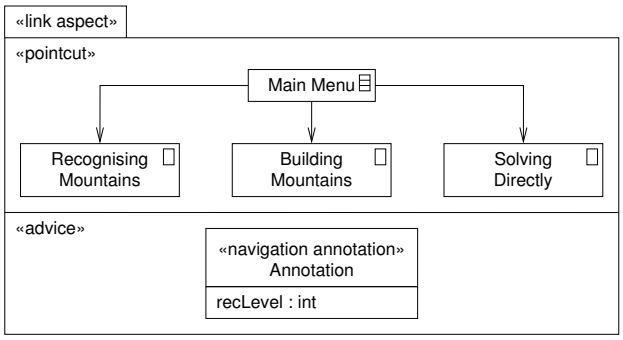
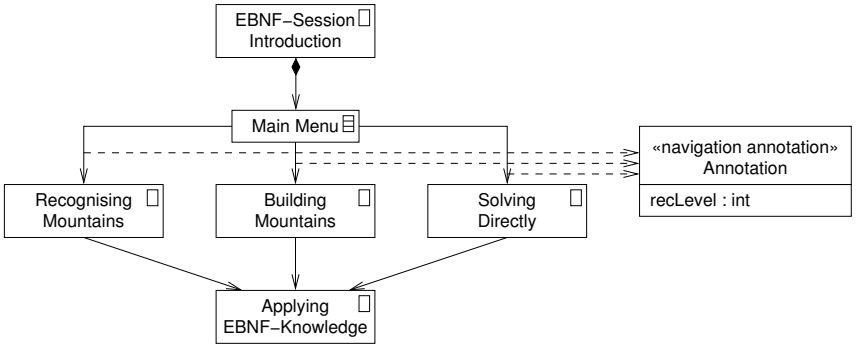


Fig. 3. Extended UWE metamodel (fragment)



(a) Annotation model aspect



(b) Weaving result

Fig. 4. Adding annotations to the SmexWeb navigation structure

4.3 Model Aspects

Adaptive navigation behaviour of Web applications has to rely on parameterisable features of the underlying navigation model. Thus, the integration of adaptability into UWE navigation structures is most simply achieved by adding annotations to navigation links: these annotations reflect the adaptations to the user model incurred by observed user behaviour [12]. Such an extension, in particular, has to take place on the navigation model level. On the one hand, each navigation link that shall be subject to adaptation has to be marked; on the other, the marking may not be desired to be uniform, but to be restricted to a certain part of the navigation model.

In Fig. 4(a), we show how the partial introduction of annotations can be achieved by using a model aspect for the SmexWeb navigation structure (cf. Fig. 1). This aspect separates the annotation feature from the navigational behaviour and documents the adaptability possibilities in a dedicated place. Each model aspect is woven statically with the principal model at design time. The pointcut of the ≪link aspect≫ describes those parts of the navigation structure model which are to be made amenable to adaptation. The advice part of the aspect specifies that the class Annotation, which is an instance of

NavigationAnnotation, has to be added to all the links (hence the name ≪link aspect≫) present in the pointcut. The attribute recLevel of the annotation represents the recommendation level of a link. The result of the weaving is shown in Fig. 4(b).

4.4 Runtime Aspects

The difference between a run time aspect and a model time aspect is that the effect of weaving the aspect with the navigation model is based on information only available at runtime. This includes information about which link is being traversed and the state of the user model. In addition, a run time aspect may change the runtime environment.

There are three types of run time aspects, link annotation aspects, link traversal aspects, and link transformation aspects (cf. Fig. 2). A ≪link annotation aspect≫ is used for adaptation of the link's annotation attributes depending, for example, on the experience of the user. A ≪link traversal aspect≫ allows us to model the adaptation of the user and navigation model when a link is traversed, e.g., to count how often a certain link is followed and, in combination with a ≪link annotation aspect≫, to increase the link's priority, if followed often. With a ≪link transformation aspect≫ new navigation links which were not available in the original navigation model can be introduced and existing links removed. For example it can be modelled that a direct navigation link is added when the system discovers that the user navigates to a navigation node quite often.

Link Annotation/Traversal Aspect Both, link annotation and link traversal aspects, have a similar structure. In both cases the pointcut is a navigation diagram and the advice is an OCL constraint. The difference is the type of OCL constraint and how weaving is performed. With a link annotation aspect the constraint is an invariant, and, for all links which are instances of the links in the navigation diagram of the pointcut, the constraint is required to hold. In contrast, with a link traversal aspect, the constraint is a postcondition constraint, and, whenever a link instance of one of the links of the pointcut is being traversed, the postcondition has to hold after the traversal of the link.

In the advice of a ≪link annotation aspect≫ we refer to the current session by using thisSession, assuming that thisSession is an instance of class Session which has at least an association to a class User representing the current user of this session (cf. Fig. 5). In addition, we assume that the variable link refers to an instance of a link in the navigation diagram defined in the pointcut. This makes it possible for the constraint to navigate to the current user and the links in the navigation diagram.

The semantics of a ≪link annotation aspect≫ is that at runtime, for all links that are instances of links in the navigation diagram defined in the pointcut of that aspect, the

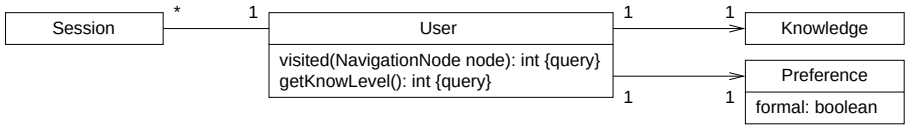


Fig. 5. Relationship between sessions and users

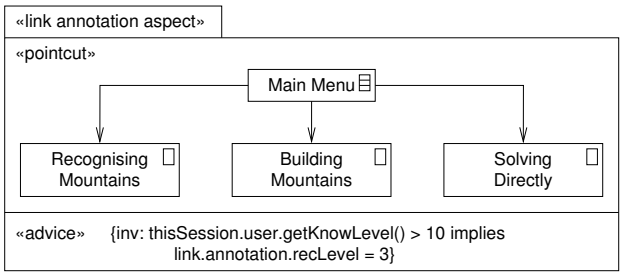


Fig. 6. Example of the use of a ≪link annotation aspect≫

instance diagram has to satisfy the constraint of the advice. Note that we are modelling the behaviour of aspects but not how this behaviour is implemented. Thus the implementation of the aspect has to ensure that the constraint is satisfied.

In the example, cf. Fig. 6, the constraint of the advice ensures that the attribute recLevel of the annotation associated with a navigation link from the navigation diagram is set to 3 when the current user has a knowledge level greater than 10. When a link from the navigation diagram is displayed, the value of recLevel can be used, for example, to display a smiley for a high recommendation level, or a frowny for a low recommendation level.

In the case of a ≪link traversal aspect≫, an advice is an OCL postcondition constraint in the context of an instance of the navigation diagram given by the pointcut of the aspect. As in an advice for a ≪link annotation aspect≫, the constraint may refer to the current session by thisSession. The result of weaving a ≪link traversal aspect≫ into the navigation model is that when a link corresponding to the navigation diagram of the pointcut is traversed, the constraint of the advice is true w.r.t. the states of the system before and after the traversal of the link. Similar to the ≪link annotation aspect≫, the postcondition has to be ensured by the implementor of the aspect after the link has been traversed. Using an aspect-oriented language, the postcondition can be implemented by an `after` advice where the pointcut is the method in the runtime environment that traverses the link. In the example, cf. Fig. 7, the constraint of the advice ensures that the traversal of links in the navigation diagram by the current user is counted.

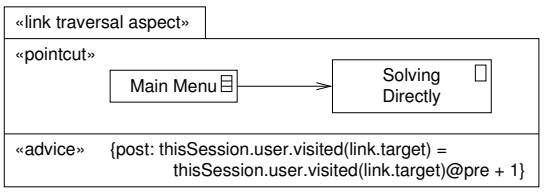


Fig. 7. Example of the use of a ≪link traversal aspect≫

Link Transformation Aspect Adaptive link generation and removal can also be modelled using aspects in a modularised way. In the SmexWeb example (cf. 1), a shortcut from Main Menu to Apply EBNF-Knowledge should be added to the navigation model after the user has solved the mountain grammar exercise directly. The pointcut of a ≪link transformation aspect≫ has the form of a navigation model, consisting of the navigation nodes, between whom extra links are to be added or removed and which can use OCL constraints specifying when the advice should be applied. The advice is another navigation model, consisting of the classes contained in the pointcut and the new navigation structure between them. Figure 8 shows an aspect where a link from Main Menu to Apply EBNF-Knowledge should be introduced when the user has visited node Solving Directly at least once.

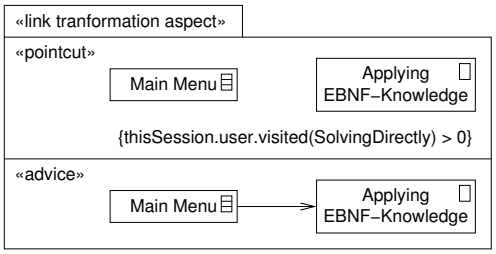


Fig. 8. Example of a ≪link transformation aspect≫

5 Related Work

Most of the currently existing methodologies tackle the modelling of adaptive Web applications by defining a rule or a filter for each point in the application where adaptation applies. As far as we know, only some of them view adaptivity as a cross-cutting feature and none of them uses aspects for modelling adaptivity.

The Hera methodology [10] provides an RMM-based notation for the representation of slices and links, which are the basic model elements of the application model. An application model is used to represent the navigation structure and presentation aspects of the Web application. Adaptation is on the one hand modelled explicitly, e.g., specifying the possible choices with links and alternative sub-slices. On the other hand, Hera constructs a rule-based adaptation model, but does not offer a visual representation of these rules. Rules are allowed to be composed recursively; Hera, however, assumes a confluent and terminating rule set.

The OO-H approach [11] proposes the use of personalisation rules to adaptivity. These rules are associated to the navigation links of the navigation model. This means that if a navigation node requires adaptation, this will be performed at runtime by execution of these rules. When a node is reachable by several links, for each link the

corresponding filter has to be defined. This introduces additional redundancy, which opens the door for bugs, like forgetting a required filter for adaptation on a link.

In WSDM [7], an adaptation specification language is defined that allows designers to specify at the level of the navigation model which adaptations of the navigation structure can be performed at runtime. Although a visual representation of the rules is missing, rules are defined orthogonally to the navigation functionality as designers are allowed to define rules on one single element (node, link) and on group of elements. Another approach is OOHDM [6], which separates adaptation from navigation by adding a wrapper class for each navigation node which requires adaptation.

The use of aspect-oriented modelling has been recognised as a general means to improve the modularity of software models [9]. Our pragmatic approach to the aspect-oriented design of adaptive Web applications takes up some aspect techniques, but is rather geared towards the application of aspect-oriented modelling. In particular, Stein et al. [17] propose a more elaborate graphical notation for selecting model elements. It supports the application of wildcards for pattern matching model element names. This notation can be used to extend our approach in specifying pointcuts. Furthermore, Straw et al. [18] have given directives such as `add`, `remove`, and `override` for composing class diagrams. The directives can be used to describe how to weave two class diagrams together and thus the weaving process in our approach may be described by their composition directives. Theme/UML [8] uses templates to model aspects. Its main focus is a generic extension of the behaviour of classes where classes and their operations are bound to the formal parameters of the templates. In our approach aspects are used to describe additional concerns of links and associations.

6 Conclusions and Future Work

We have demonstrated the use of aspects for modelling adaptive Web applications in the UWE method by separating the navigation model from the adaptation model. A link aspect introduces navigation annotations to particular links for link reordering, annotating, or hiding due to user behaviour. This link aspect is applied at model time and thus captures cross-cutting model information in a dedicated place. During runtime, link annotation and link traversal aspects record information of the user behaviour and accordingly adapt the annotations or the user model for further use in the presentation layer. Additionally, the navigation structure becomes adaptable by link transformation aspects, that allow the designer to have fine-grained control on when links are added or removed.

We expect that contents adaptation and presentation adaptation can also be described by aspect-oriented modelling techniques, and we plan to investigate this topic in more detail. In particular, the effect of adding and modifying annotations on the navigation level to the presentation can again be described by aspects. We have restricted ourselves to a rather lightweight approach to integrating aspects into the UML and UWE in that we used the built-in extension facilities provided by the UML to define UML profiles. A more elaborate pointcut and advice description language is certainly desirable and subject to future research. For tool support, we plan to integrate adaptivity aspects in the open-source Web application modelling tool ArgoUWE.

References

1. Florian Albrecht, Nora Koch, and Thomas Tiller. SmexWeb: An Adaptive Web-based Hypermedia Teaching System. *J. Interactive Learning Research*, 11(3–4):367–388, 2000.
2. Thomas Baar, Alfred Strohmeier, Ana Moreira, and Stephen J. Mellor, editors. *Proc. 7th Int. Conf. Unified Modeling Language (UML'04)*, volume 3273 of *Lect. Notes Comp. Sci.* Springer, Berlin, 2004.
3. Eduardo Barra, Gonzalo Génova, and Juan Llorens. An Approach to Aspect Modelling with UML 2.0. In *Proc. 5th Wsh. Aspect-Oriented Modeling (AOM'04)*, Lisboa, 2004.
4. Peter Brusilovsky. Methods and Techniques of Adaptive Hypermedia. *User Model. User-Adapt. Interact.*, 6(2–3):87–129, 1996.
5. Peter Brusilovsky. Adaptive Hypermedia. *User Model. User-Adapt. Interact.*, 11:87–110, 2001.
6. Juan Cappi, Gustavo Rossi, Andres Fortier, and Daniel Schwabe. Seamless Personalization of E-commerce Applications. In Hiroshi Arisawa, Yahiko Kambayashi, Vijay Kumar, Heinrich C. Mayr, and Ingrid Hunt, editors, *Proc. Wsh. Conceptual Modeling in E-Commerce (eCOMO'01)*, volume 2465 of *Lect. Notes Comp. Sci.*, pages 457–470. Springer, Berlin, 2003.
7. Sven Casteleyn, Olga De Troyer, and Saar Brockmans. Design Time Support for Adaptive Behavior in Web Sites. In *Proc. 18th ACM Symp. Applied Computing*, pages 1222–1228. ACM Press, 2003.
8. Siobhán Clarke. Extending Standard UML with Model Composition Semantics. *Sci. Comp. Prog.*, 44(1):71–100, 2002.
9. Robert E. Filman, Tzilla Elrad, Siobhán Clarke, and Mehmet Aksit, editors. *Aspect-Oriented Software Development*. Addison-Wesley, 2004.
10. Flavius Frasincar, Geert-Jan Houben, and Richard Vdovjak. Specification Framework for Engineering Adaptive Web Applications. In *Proc. 11th Int. Conf. World Wide Web (WWW'02), Web Engineering Track*, Honolulu, 2002.
11. Irene Garrigós, Jaime Gómez, and Cristina Cachero. Modelling Dynamic Personalization in Web Applications. In Lovelle et al. [14], pages 472–475.
12. Nora Koch. *Software Engineering for Adaptive Hypermedia Systems: Reference Model, Modeling Techniques and Development Process*. PhD thesis, Ludwig-Maximilians-Universität München, 2001.
13. Nora Koch and Andreas Kraus. Towards a Common Metamodel for the Development of Web Applications. In Lovelle et al. [14], pages 497–506.
14. Juan Manuel Cueva Lovelle, Bernardo Martín González Rodríguez, Luis Joyanes Aguilar, José Emilio Labra Gayo, and María del Puerto Paule Ruíz, editors. *Proc. 3rd Int. Conf. Web Engineering (ICWE'03)*, volume 2722 of *Lect. Notes Comp. Sci.* Springer, Berlin, 2003.
15. Object Management Group. Unified Modeling Language Specification, Version 2.0 (Superstructure). Revised final adopted draft, OMG, 2004. `http://www.omg.org/cgi-bin/doc?ptc/2004-10-02`.
16. Dominik Stein, Stefan Hanenberg, and Rainer Unland. An UML-based Aspect-Oriented Design Notation For AspectJ. In *Proc. 1st Int. Conf. Aspect-Oriented Software Development*, pages 106–112. ACM, 2002.
17. Dominik Stein, Stefan Hanenberg, and Rainer Unland. Query Models. In Baar et al. [2], pages 98–112.
18. Greg Straw, Geri Georg, Eunjee Song, Sudipto Ghosh, Robert France, and James M. Bieman. Model Composition Directives. In Baar et al. [2], pages 84–97.