Towards Variability Modelling for Reuse in Hypermedia Engineering*

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 discuss variability modelling for hypermedia applications. Inspired by domain engineering, we propose a domain engineering based method for hypermedia development. Since several adaptive hypermedia become more and more popular to incorporate different information views for different audience or environments, we believe that it is important to move variability capturing to modelling phases. Several established modelling views of hypermedia application are discussed from the variability point of view. We also explain modelling techniques by means of examples for the application domain view, the navigation view, the presentation view and discuss importance of the user/environment view for parametrisation of components.

1 Introduction

Hypermedia applications are very popular since the Internet became their main platform for implementation. Wide acceptance of the hypermedia applications for delivering huge amount of information with complex structure through the Internet raised the need for studying methods of hypermedia applications construction. Hypermedia construction moved from authoring to development of complex systems. Several methods for hypermedia design appear, for example Hypertext Design Model (HDM) [10], Object Oriented Hypermedia Design Method (OOHDM) [18], Relationship Management Methodology (RMM) [14] or UML-based Hypermedia Design Method (UHDM) [13].

These methods systematise development and provide techniques for modelling hypermedia applications. Since hypermedia are now incorporated also in data intensive applications (e.g., web-based information systems) or applications characterised by frequent change of some aspects, automation of some steps significantly improves maintainability. The example of such approach is RMM-Based Methodology for Hypermedia Presentation Design [9].

We address different point of view to design automation — application construction from reusable components. Hypermedia applications constitute a good

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

platform for information serving and adaptation to different audience and environment. Applications such as tutoring systems, e-courses or digital libraries can be seen as an *application family.* Several components of such applications can be reused, e.g., subset of information, navigation styles, or presentation objects. Let's take an example of an e-course for Introduction to Software Engineering. Some concepts presented in this e-course but in different context and depth can be served in an e-course of Software Systems Architectures available on the web, or the course, recorded onto CD ROM as a training material. It means that at least conceptual model together with information associated can be reused. Such reuse is addressed in *domain engineering.* The domain engineering based software development inspired us to research possibilities of employing such approach in hypermedia development. The result of this research is an approach to variability modelling for hypermedia presented in this paper.

The rest of the paper is structured as follows. Section 2 describes background of domain engineering as a base for our research. Section 3 is devoted to review of current approaches to hypermedia development. In section 4 we summarise proposed approach and relate our approach to domain engineering and application engineering. Variability as the main idea behind feature modelling and a key component of domain engineering is discussed in section 5. We discuss where variability features in hypermedia can be found and also propose solutions to variability modelling in application, navigation and presentation design. Finally, we give conclusions and proposal for further work.

2 Background — Domain Engineering vs. Application Engineering

Domain engineering concentrates on providing reusable solutions for families of systems. According to [15], the domain engineering is a systematic way of identifying a domain model, commonality and variability, potentially reusable assets, and an architecture to enable their reuse. The idea behind this approach to reuse is that the reuse of components between applications occurs in one or more application domains. The components created during domain engineering activities are reused during subsequent application system engineering phase. Several approaches to domain engineering for software systems appear, for example Model Based System Engineering [21] of SEI, which was later replaced by Framework for Software Product Line Practice [22]. Generative programming [5] and multi-paradigm design for AspectJ [20] also address domain engineering.

Domain engineering process consists of three main activities [5]:

- *domain analysis* defines a set of reusable requirements for the systems in the domain;
- *domain design* establishes a common architecture for the systems in the domain;
- *domain implementation* implements reusable components, domain-specific languages, generators, and a reuse infrastructure.

The result of the domain analysis is a domain model. This model represents common and variable properties of the systems in the domain and relationships between them. Domain analysis starts with a selection of the domain being analysed. Concepts from the domain and their mutual relationships are analysed and modelled. The domain concepts in the model may represent a domain vocabulary. Each concept is then extended by its common and variable features and dependencies between them. This is the key concept of domain engineering. Variable features determine the configuration space for the systems family.

Domain design and domain implementation are closely related and sometimes are presented as one phase. The domain design produces generic abstract architecture for the family of systems according to the well known architectural patterns (layered, model-view controller, etc.). Domain implementation implements the architecture by appropriate technology for particular environment. Figure 1 summarizes software development based on domain engineering.

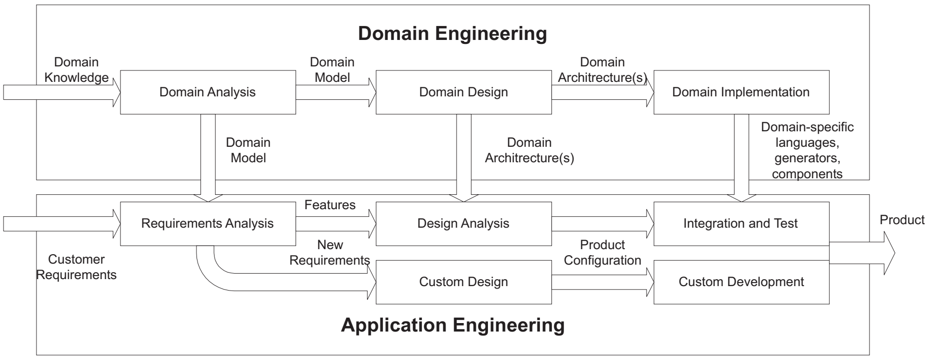

Fig. 1. Software development based on domain engineering according to [21].

3 Current View on Hypermedia Engineering

Likewise earlier works on software systems development, current hypermedia development methods conform single system development approach, i.e. the focus is put on the development of single systems rather than reusable models for classes of systems. Most methods are described in terms of "what to model" question. It means that development process chains activities related to application domain, navigation and presentation modelling together with requirements capture and implementation. OOHDM [18], W2000 [1] or UHDM [13] are examples of such methods for engineering single hypermedia application.

Introducing adaptive hypermedia raises the need for studying variability of presentation and navigation. UHDM and RMM-Based Methodology for Hypermedia Presentation Design [9] are examples of such methods where adaptation is considered already during the modelling phase. RMM based extension addresses similar adaptation and user modelling techniques like Adaptive Hypermedia Application Model (AHAM) [3]. AHAM provides modelling primitives for a user and its level of knowledge, preferences and goals description and also the rule based language for specification of adaptation rules and actions.

User requirements are mostly captured by use cases [18, 1] or their extension — user interaction diagrams [12]. The concepts and their mutual relationships are derived from user requirements and modelled by a class diagram [18, 1, 13] or an entity-relationship diagram [9]. The navigation model usually consists of two schemas: navigation class schema and navigation context schema. Several mechanisms are employed for modelling navigation such as views [18, 13], nodes and links, collections [1], states and transitions [8], and slices and their relationships [9]. Abstract interface design is modelled for example by abstract data views [18], presentation blocks and relationships [9], pages and framesets [13, 1].

Hypermedia and especially web-based applications are often accessed by several types of users. Information provided is presented also using several environments. Moreover, these environments require different implementation modelling languages (e.g., WAP, HTML). These observations raised the need for studying reuse for families of hypermedia applications.

Current methods (oriented to single system development) provide modelling techniques, which allow variability modelling partially at the conceptual level by means of generalisation/specialisation relationship or `xor` constraints between aggregation or association relationships. However, the specialisation relationship is rather implementation oriented technique for variability modelling. Moreover, specialisation explicitly prescribes that the particular concept instance may belong only to one specialized subconcept. Similarly can be seen the `xor` constraint. The variability at the attribute level cannot be modelled by techniques employed in mentioned methods. Another technique where variability can be partially captured is views modelling. The views are mostly described by an object query language. It means that at the navigation or presentation level only several concepts or their attributes appear in one context. These techniques are rather configuration oriented than variability description oriented.

4 Towards Domain Engineering Approach for Hypermedia

We proposed a framework for incorporating established hypermedia application modelling aspects into activities of domain engineering. This inclusion is summarised in Figure 2. Domain analysis for hypermedia application involves real world domain conceptual and feature model and representation of conceptual and feature model.

Domain design for hypermedia application involves information modelling, navigation modelling, presentation modelling, user modelling and environment modelling. It also incorporates mappings between these models. The mappings

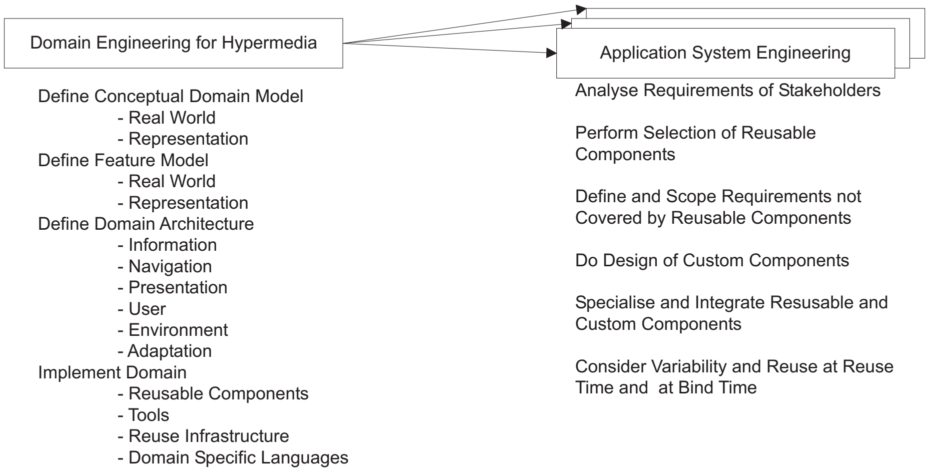

Fig. 2. Domain engineering based approach for hypermedia.

involve specification of rules and determination of variable feature mappings between developed models.

Domain implementation includes construction of parameterised implementation components with their mutual dependencies. Parameterisation of implementation components can be realised for example in the form of HTML templates, active server templates, WAP templates or components in other implementation languages. Domain implementation should also incorporate domain specific languages such as query languages or languages for selecting and integrating components for particular application from the application family. All of the mentioned models and their parameterisation are transformed into these implementation components.

Application engineering as a subsequent activity following domain engineering activities is out of scope of this paper. The hypermedia application is built according to the requirements, specific for particular application. Similarly to software systems, the requirements are split into those, which

- can be satisfied by the components from the application family framework created during the domain engineering process,
- should be satisfied by a custom development.

Results of the framework generation and custom development are integrated into the final product.

5 Variability Modelling for Hypermedia

Variability is considered from two points of view. The first point of view is variability at the system level. It means existence of versions of system's components and system releases. Version control in hypermedia domain was studied at the document level in several works [16, 2, 19]. All mentioned works provide a model of versions and a model of configuration, which defines how the versions contribute into the final configuration. The approach described in [2] stresses importance of families at the level of versions.

The second point of view is the variability at the application level. The variability is considered in several modelling aspects of hypermedia application [7]. The most significant aspects in hypermedia application modelling are application domain, navigation and presentation. The variability modelling is important in all these aspects.

Application domain model should be mapped into the content. Several concepts from application domain model can be expressed by different contents. Moreover, the same content can be represented by different media and this content can evolve in time. The content can be also presented in different formats, e.g. as a book, lecture, or an article. Also overall access to the content can be managed through different patterns such as digital library, e-course (virtual university), on-line help, etc.

Hypermedia application can be used by different types of users. Each user group may require different information to browse, different composition of presented information (local navigation), and different order and interconnections between information chunks (global navigation). Different navigation styles can be determined also by the target environment where the information is served to a user.

Similarly, different audience may require different appearance of information chunks, different layout and different presentation of organisation of read information. Target environment can also restrict possibilities to presentation. Thus, it is important to capture also this kind of variability.

5.1 Application Domain

We consider two views of application domain for hypermedia. The first is conceptual model of the "real world" and the second is representation of the real world concepts. By representation of the real world view we mean appearance of the concepts in the hypermedia application. The hypermedia application serves information by its content. The content is somehow structured. For example, concepts may be described in the paragraph of a section in a paper or book, or can be described as a part of e-lecture.

Most of current methods reflect only one conceptual view in the hypermedia application — the representation view. A little attention is paid to real world concepts organisation as the vehicle for information modelling. It is very important to consider both views when we think about a reuse because the components of both views can be reused separately.

We model concepts for both views using the same techniques. Since the UML is de facto standard not only for software modelling, we employ this language for application domain modelling. A concept is modelled by the class, which is stereotyped by the ≪`Concept`≫ stereotype. Concepts can be connected by one of association, generalisation/specialisation, or aggregation relationships.

The example of simple conceptual model for representation view is depicted in Figure 3. We take the e-course and its e-lecture as the representation domain. Figure 3 depicts `Course`, `Lecture`, `Student` and `Lecturer` concepts. Depicted associations are decorated by stars, which represent “many” cardinality.

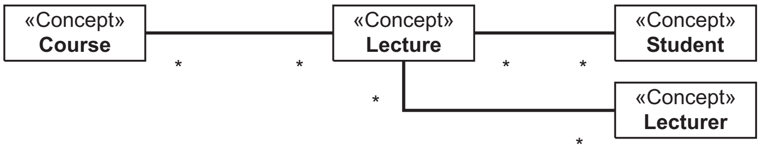

Fig. 3. Conceptual model of course representation.

Figure 4 depicts example of the real world view — the functional programming domain basic concepts. The basic concepts of this domain are `Functional Program`, `Function`, and `Expression`. These basic concepts should be described in any lecture on functional programming.

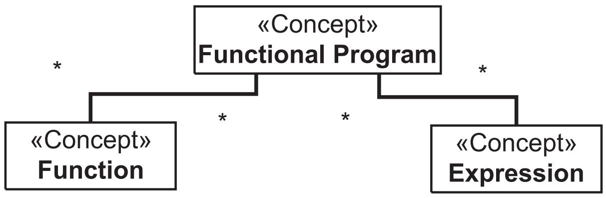

Fig. 4. Conceptual model of the Functional programming domain.

Concepts can have common and variable features in both (representation and real world) views. We consider mandatory features or optional features of the concept. Mandatory features must be incorporated in the domain model of any application from the application family. On the other hand, optional features need not to be incorporated in concepts specified in particular application.

Variability relationships must also be considered. Features can stand as:

– mutually exclusive variants,
– mutually required features,
– features, whose optional number can be chosen.

A feature, which has defined such variability relationships for its subfeatures, is denoted as a *variation point* [21] or *variation* [11].

Feature diagrams are the key part of the feature modelling in methods such as MBSE [21] or generative programming [5]. UML is employed for example in Reuse-Driven Software Engineering Business (RSEB) [15, 11]. Variability points are considered in the use case view and the component view in [15]. The variability modelling in the conceptual view is discussed in [11]. In this work the extension for features, their attributes and variation points is defined. The attributes can be suppressed in the model. However, the definition of extension for feature and variability modelling has some inconsistencies. First, a feature is defined as a stereotype of the class element. Since the class is used also for modelling concepts, it seems that the concept feature should be defined as the stereotype of another element. Second, attributes of the feature need not to be distinguished because they can be represented as subfeatures of the feature.

To avoid above mentioned problems we have proposed the following solution. We define the ≪`ConceptualFeature`≫ stereotype for the UML abstract element `Feature`. The ≪`ConceptualFeature`≫ stereotype inherits properties and relationships from its predecessor. The `Feature` element is associated to the `Class` element by the aggregation relationship at the metamodel level. If features are inserted into a class at the model level, they are mandatory for the class. A class can have features in the model but features cannot have their features.

We need to express whether a feature belongs to a concept or another feature at the model level. The feature model of particular concept can form a complex hierarchy. For this purpose we define the constraint, which restricts aggregation relationship at the metamodel level for ≪`ConceptualFeature`≫ and allows to define association between the concept, its features and their subfeatures.

We defined the special purpose stereotypes (named ≪`MandatoryFeature`≫, ≪`OptionalFeature`≫) of ≪`ConceptualFeature`≫ stereotype for modelling mandatory and optional features. Features are associated to concepts or features by association relationships. We also defined the ≪`VariationPoint`≫ stereotype of the `Element` UML metaclass with the `VariationKind` attribute to enable handling variation points. The variation kind can be `XOR`, `OR` or `AND`.

Figure 5 depicts a part of the feature model for the `Lecture` concept. `Lecture` must have at least the `Lecturer` and `Parts` features (they are depicted as mandatory features in the figure). The `Name` and `Description` are optional. Their subfeatures can be interpreted similarly. The figure depicts also `OR` variation point of the `Picture` feature. It prescribes that this feature must appear at least with one of the `Text`, `Audio`, or `Video` features. For space limitation we show here only a part of the `Lecture` concept feature model. Each concept from the conceptual model can be described similarly.

Figure 6 depicts a part of the `Functional Program` concept feature description. Functional program as a real world domain concept for a lecture should be somehow described (`Definition` mandatory feature). The computation of functional program should also be explained in e-lecture (`Computation` mandatory feature). The composition of functional program is also important (`Program Definition` mandatory feature). These features must be described in any lecture on functional programming. Remaining features are optional.

Computation of a functional program is based on expression evaluation (`Form` optional feature). `OR` variation point of the `Computation` feature prescribes that at least one subfeature must be described in any e-lecture on functional programming. The meaning of rest features can be directly derived from the figure. The reminder of the concepts from discussed domain can be described similarly.

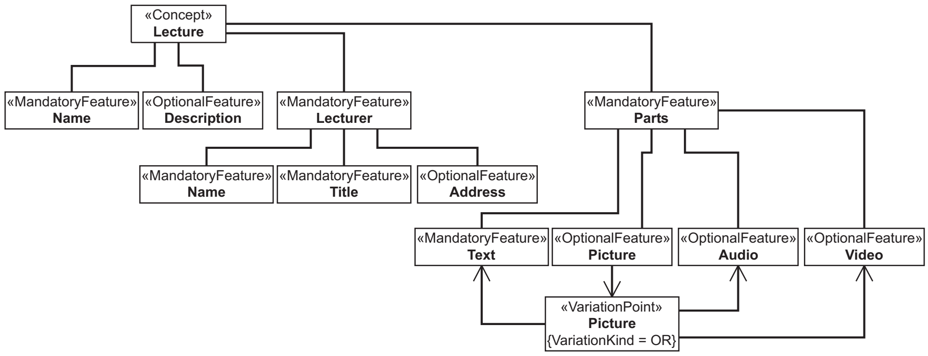

Fig. 5. Feature model of the Lecture concept.

Whereas two views on an application domain are defined, the mapping between these views should be defined. Mapping can be different for different applications from the application family. This is subject of the domain design process. The mapping represents the realisation of concepts and their attributes in the content representation. It can be specified by transformation specifications, view mechanism, instantiation etc.

5.2 Navigation

Navigation design is a counterpart of the component identification in the domain design for software application families. The navigation design defines groupings of information chunks into higher level contexts, which are interconnected by links. Links can be derived from semantics relationships between the concepts and/or features, or from the composition and variability relationships defined in the feature models.

For navigation specification we employ the UML state diagrams. Features or concepts are transformed into states (ordinary or composite). Variability is handled by two approaches. The former is structural, i.e. the composite state is decomposed into parallel or alternative states [6]. The latter is the specification of the adaptation rules related to the transitions between states, or the specification of an entry, exit or internal rules of states [8].

The features which are members of an `AND` variation point and mandatory features are transformed directly into parallel states. The features of a `XOR` variation point are transformed into alternative states. The features of an `OR` variation point and optional features are transformed into parallel states with an entry parameter determining whether the states will be considered in a generation of an application or not. This is denoted as *reuse at use time*.

Second type of reuse is *reuse at bind time*. The components are incorporated into the implementation but their usage is determined according to the rules, which are evaluated at bind time. These dynamic rules can be composed according to a user or an environment model.

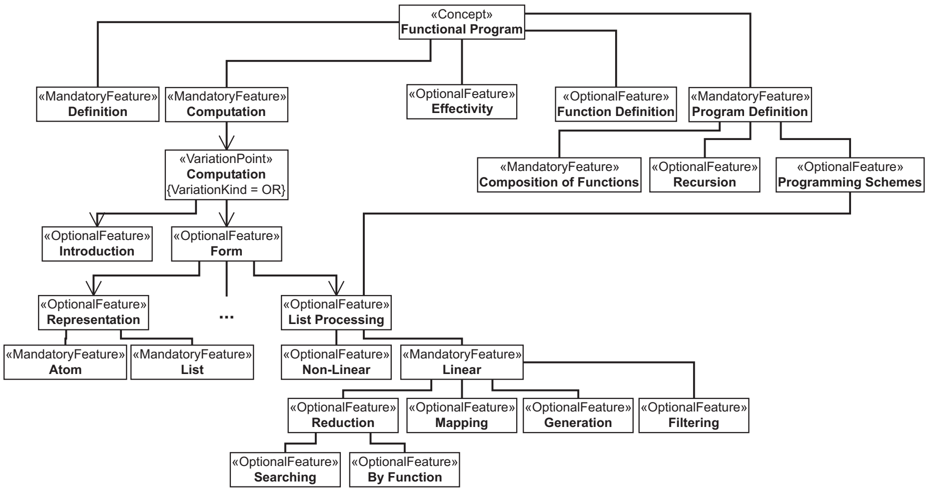

Fig. 6. Feature model of the Functional Program concept.

The example of the state-diagram navigation model is depicted in Fig. 7. The diagram contains selected transformed features from Fig. 6. This model can be seen as a model for one application from the application family. It means that the selection process of reusable components was executed.

As it is obvious, the `Definition` feature of the `Functional Program` concept is depicted as the `Introduction to FP` state. Similarly the `Computation` feature is mapped to the `Computation in FP` state and the `Introduction` feature is mapped to the `Introduction to LISP` state. Other states in the figure is derived from fetures similarly. State diagram in the Fig. 7 also contains rules, which reference mostly user characteristics.

5.3 Presentation

Presentation design also belongs to domain design activities. Presentation design aims at the specification of presentation objects. These abstract presentation objects have their appearance features and also are associated with spatial relationships. The relationship can also have features associated. Such a feature is a closer specification of the shifting or relationship role. The role can be left, right, above, bottom, in front, or behind.

Variability is expressed similarly to the domain analysis model. A presentation object can have its features, which are variable or common with respect to the variability relationships between them. The features of the spatial relationship can also be considered as variation points.

Presentation objects are mapped to navigation and/or conceptual design elements. Mapping is performed at reuse time or at bind time according to the specified rules. The mapping at reuse time is performed by choosing a particular

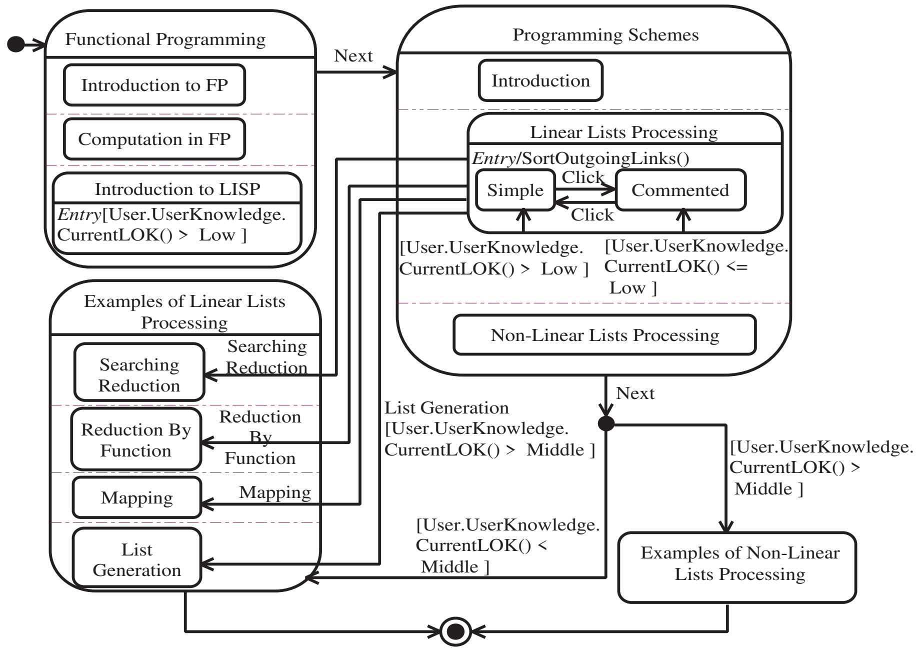

Fig. 7. Navigation model of functional programming e-lecture with adaptation rules.

set of the presentation features and its mapping to navigation or presentation objects. The mapping at bind time is performed by choosing a set of variable presentation features with association of rule when particular feature is actual. State transition graph is refined by presentation states of particular state, which is associated to concept or its features.

Presentation or navigation can be adapted or parameterised according to user characteristics or environment characteristics [4]. Preferences, goals, level of knowledge are examples of user characteristics. Environment characteristics are elements, which are provided by implementation environment for an implementation of high level design components. User and environment concepts have also associated variable and common features. These features can also be constrained by variability relationships.

Features of a user or environment model can be used in rules for parameterisation of presentation and navigation models or for specification of transformation to the implementation. We introduced such mapping in [8]. Some mappings can be derived directly from the application domain model where variable features are determined according to different requirements of stakeholders.

6 Conclusions and Further Work

In this paper we discussed domain engineering based approach for reuse in hypermedia development. Proposed approach is the combination of lessons learned from implementation of systems and synthesis of these experiences with principles of domain engineering. A part of this approach was employed in the *Con4U* [17] project for development of the web based conference review system. The result of this project was a framework, which allowed us to release two applications: the first for conference paper reviews and the second for student project reviews. Particular application was created by a configuration and an instantiation of this framework.

Proposed approach extends classical notion of single hypermedia system development to the application family engineering. We defined which activities belong to established domain engineering high level activities. We proposed separation of two conceptualy different aspects in application domain engineering for hypermedia and discussed advantages of such separation in design from reuse point of view. The contribution to the modelling is the revised definition of the UML extension for variability modelling at the conceptual level. We showed how variability can be considered in application domain, navigation and presentation modelling for hypermedia.

There is a need to define domain specific languages for manipulating with reusable components for hypermedia. The investigation and development of generators for specific environments, which consider parametrisation is also needed. In spite of practical evaluation in the mentioned Con4U case study and e-lecture in described examples, more case studies or practical realisations are needed.

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] M. Bieliková and P. Návrat. Modelling versioned hypertext documents. In B. Magnusson, editor, *System Configuration Management, ECOOP'98 SCM-8 Symposium*, pages 188–197, Brussels, Belgium, July 1998. Springer-Verlag, LNCS 1439.

[3] 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.

[4] Peter Brusilovsky. Adaptive hypermedia. *User Modeling and User-Adapted Interaction*, 11(1-2):87–100, 2001.

[5] Krysztof Czarnecki and Ulrich Eisenecker. *Generative Programing: Principles, Techniques, and Tools.* Addison Wesley, 2000.

[6] 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.

[7] Peter Dolog and Mária Bieliková. Hypermedia modelling using UML. In Petr Hanáček, editor, *Proc. of ISM'2002 - Information Systems Modelling*, pages 79–86, Rožnov pod Radhoštěm, Czech Republic, April 2002.

[8] Peter Dolog and Mária Bieliková. Navigation modelling in adaptive hypermedia. In Paul De Bra, Peter Brusilovsky, and Ricardo Conejo, editors, *Proc. of 2nd Conference on Adaptive Hypermedia and Adaptive Web-based Systems*, Malaga, Spain, May 2002. Springer LNCS 2347.

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

[10] Franca Garzotto and Paolo Paolini. HDM — a model-based approach to hypertext application design. *ACM Transactions on Information Systems*, 11(1):1–26, January 1993.

[11] Martin L. Griss, John Favaro, and Massimo d' Alessandro. Integrating feature modeling with the rseb. In P. Devanbu and J. Poulin, editors, *Proc. of 5th International Conference on Software Reuse*, pages 76–85, Victoria, Canada, June 1998. IEEE Computer Society Press.

[12] N. Guell, Daniel Schwabe, and Patricia Vilain. Modelling interaction and navigation in web applications. In *Proc. of the WWW and Conceptual Modeling Workshop*, Salt Lake City, 2000. Springer LNCS.

[13] Rolf Hennicker and Nora Koch. A UML-based methodology for hypermedia design. In S. Stuart A. Evans and B. Selic, editors, *Proc. of UML 2000 Conference*, York, England, October 2000. Springer LNCS 1939.

[14] Tomás Isakowitz, A. Kamis, and M. Koufaris. Extending the capabilities of RMM: Russians dolls and hypertext. In *Proceedings of the 30th Annual Hawaii International Conference on System Sciences*, January 1997.

[15] Ivar Jacobson, Martin Griss, and Patrik Jonsson. *Software Reuse: Architecture, Process and Organization for Business Success.* ACM Press, 1997.

[16] Theodor Holm Nelson. Xanalogical Structure, Needed Now More than Ever: Parallel Documents, Deep Links to Content, Deep Versioning and Deep Re-Use . Available at: http://www.sfc.keio.ac.jp/ ted/XUsurvey/xuDation.html. Accessed on March 1, 2002.

[17] Richard Richter, Róbert Trebula, Peter Lopeň, Ján Zázrivec, and Peter Kósa. Con4U: Project and paper review system, May 2002. Team Project supervised by Peter Dolog. Dept. of Computer Sci. and Eng., Slovak University of Technology.

[18] 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.

[19] Luiz Fernando G. Soares, Rogério F. Rodrigues, and Débora C. Muchaluat Saade. Modeling, authoring and formatting hypermedia documents in the HyperProp system. *ACM Multimedia System Journal*, 8(2):118–134, March 2000.

[20] Valentino Vranić. AspectJ paradigm model: A basis for multi-paradigm design for AspectJ. In Jan Bosch, editor, *Proc. of Third International Conference on Generative and Component-Based Software Engineering (GCSE 2001)*, pages 48–57, Erfurt, Germany, September 2001. Springer LNCS 2186.

[21] James V. Withey. Implementing model based software engineering in your organization: An approach to domain engineering, 1994. CMU/SEI-94-TR-01, see also `http://www.sei.cmu.edu/mbse/index.html`.

[22] James V. Withey. Investment analysis of software assets for product lines, 1996. CMU/SEI-96-TR-010.