Using UML-Based Feature Models and UML Collaboration Diagrams to Information Modelling for Web-Based Applications

Peter Dolog and Wolfgang Nejdl

L3S Research Center
University of Hannover
Expo Plaza 1, 30539 Hannover, Germany,
{dolog, nejdl}@l3s.de

Abstract. Web oriented software technology has provided access to information serving environments for a broad audience. This situation requires web-based software applications which satisfy increasing variety of requirements of the broad audience. Such variability can be found in requirements for information but also for environment which is serving the information. In this paper, we discuss a method which utilizes the UML-based feature modelling to support the need to model the variability. The information and environment configurations are modelled as common and variable features of application domain and environment concepts. Separation of feature models into application domain and environment allows us to select several configurations of environments to deliver particular information. The UML collaboration diagrams model collaborations between the application domain and environment concept and feature instances as an abstraction for presented information fragments in a web application.

Keywords: Feature modelling, information modelling, web-based application, UML collaboration diagrams

1 Introduction

Open environments such as Internet provide us with possibility to serve information to users with different goals, background, interests and so on. To build applications for such an environment requires to focus on diversity of features user of the applications can require in case of information provided and also in case of how the provided information is re-/presented.

Explicit information about a purpose of already authored content and the purpose of the information being described by the content can help requirements analyst, designer or author of the content to compare existing material with the diverse requirements of users and to say whether the content can be reused or new content should be provided. Current established modelling methods for web based applications do not provide sufficient mechanisms to express such information.

This paper explores an approach to information modelling which is based on the idea of information product line and is inspired by domain engineering approaches for building software systems like generative programming [6] or product line practices [23] where feature models play an important role. Our method is also based on the idea that an information is communicated usually using several concepts to a user. We use the UML collaboration diagrams to model such a design view where information features collaborate together to fulfill a main information goal. This extends our previous work on feature modelling for the domain engineering approach for hypermedia engineering (DEAHE) [7]. Our approach is based on:

- Two views on information: domain and environment;
- Feature models which model common and variable features of concepts and variation points in both views;
- Collaboration diagrams which refine and integrate feature models of both views.

Feature modelling allows us to capture common and variable features of concepts being covered by an application. Common and variable features reflect diverse requirements of different current and possibly future users of particular application for the information and also environment through which it is served. Collaborations between different features are determined by roles. The roles reflect the context in which information can be used and presented in an environment component.

The feature and collaboration models allow us to specify conditions in which features from two separate models can be used. The conditions determine which combinations of features provide us with meaningful information. Different configurations of features and roles represent possible web based applications which form an application family. Such an approach provides us with following advantages:

- The separation of the domain and information/environment models allows us to provide information about concepts in different environment and vice versa;
- Mandatory and optional features together with variation points in both models allow us to maintain information which reflect different successful implementations of web-based application;
- Instance roles and their collaborations allow us to maintain information how domain features collaborated in information serving environments installed at a customer site.

1.1 Motivating Scenario

To illustrate our approach, we refer to a scenario where a training company needs to be able to configure a training suite where, besides other courses, Java course is provided. The suite will be provided for a group of people with a background in electrical engineering — *tutorial*. They would like to gain an overview of the Java language and possibly ideas where they can use it.

On the other hand, the company needs to configure the training suite for a group of people from a software company which just acquired a project where Java language was chosen as a programming language — *detailed course*.

In addition, in both cases the training material has to be provided with different environments as it is needed to support mobile learners and also learners in offices.

The company's vision is that similar situation can happen in the future. Due to this they decide to document design models in a way which is suitable for the varying requirements.

As this situation shows, information about the same concepts has to be presented in a different way according to whether a detailed presentation is needed or just overview. In both cases the information is articulated by several concepts playing different roles. For example, to give an introduction to Java objects in a simple Java tutorial, object state and object behavior concepts are needed. This can be called the smallest meaningful set of concepts to present the Java object in any content configured for the Java lecture. Additional concepts might be suited to describe other aspects of the object state for example. Information modelling framework has to provide tools for modelling the concept configurations by means of mandatory and optional features of the concept and configuration dependencies between them.

As the information can be delivered by different environments, the domain concepts which are used to model information have to be separated form the environment concepts used to model the environment. On the other hand, interconnections between domain concepts and the environment concepts have to be modelled to document successful cases of training suite applications. The information modelling framework has to provide means for modelling such collaboration interconnections between several roles the concepts and features play in the environment.

1.2 Paper Structure

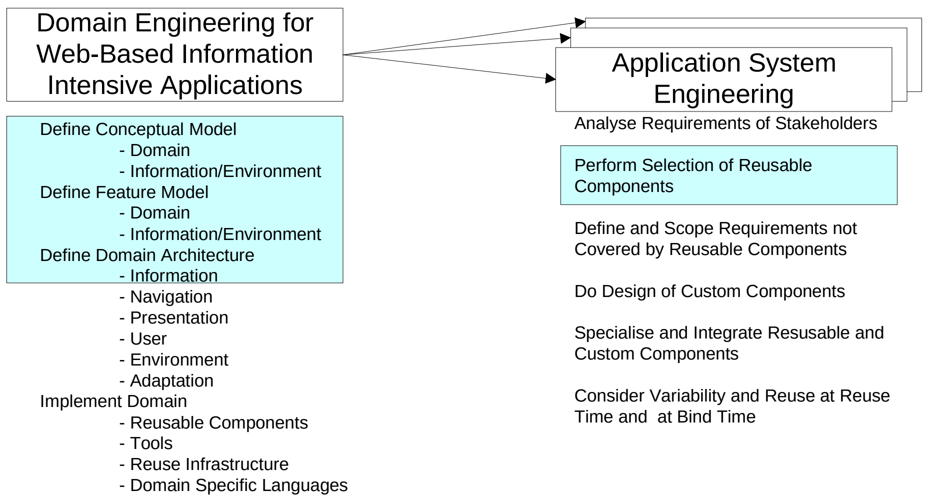
The rest of the paper is structured as follows. Section 2 provides a brief overview of an approach and its context in web-based application development. Section 3 describes UML-based feature modelling for information and environment. Section 4 discusses collaboration model and its refinements. A short summary of application and experiences with this approach is given in Sec. 5. Section 6 discusses our approach in the context of related work. The paper concludes with some remarks and proposals for further work (Sec. 7).

2 Overview of a Modelling Framework

Figure 1 depicts a framework for engineering web-based information intensive product lines [7]. This paper focuses mainly on the shaded areas in Fig. 1. In this paper we discuss a method where:

– *Domain and information conceptual models* are used to model concepts and their mutual relationships;

- *Domain and information/environment feature models* are used to maintain common and variable features of concepts and their dependencies as companies' experience in how different concepts can help in presenting another concept;
- *Collaboration models* refine the feature models and are used to express how features from domain and information/environment feature models collaborate together to achieve a main information goal.



Domain Engineering for Web-Based Information Intensive Applications
Define Conceptual Model
- Domain
- Information/Environment
Define Feature Model
- Domain
- Information/Environment
Define Domain Architecture
- Information
- Navigation
- Presentation
- User
- Environment
- Adaptation
Implement Domain
- Reusable Components
- Tools
- Reuse Infrastructure
- Domain Specific Languages
Application System Engineering
Analyse Requirements of Stakeholders
Perform Selection of Reusable Components
Define and Scope Requirements not Covered by Reusable Components
Do Design of Custom Components
Specialise and Integrate Resusable and Custom Components
Consider Variability and Reuse at Reuse Time and at Bind Time

Fig. 1. Domain engineering based approach for engineering web-based information intensive product lines

The purpose of the conceptual models is to document domain and environment vocabulary used in all other models. As the domain/content presented in a training suite in case of Java lecture from our example scenario, the domain conceptual model will refer to concepts from Java programming language. As the environment mentioned in the scenario is a course, the course structure and some other concepts will depict in the environment conceptual model.

The purpose of the feature models is to document presentation relations of all concepts from the conceptual models to other concepts in that model; i.e. which other concepts are used to articulate particular concept, e.g. on Java objects.

The purpose of the collaboration models is to connect the instances from the two models according to a selection performed by a designer. In addition, the purpose of the collaboration models is to model messages needed to be sent between the roles of those instances depicted in the models when an interaction with particular role instance was requested by a user.

The process of such information modelling can be summarized in the following steps:

1. Define a conceptual model for a domain and environment (e.g. concepts used to teach Java programming language served in a course as an environment);
2. Define a feature model for all concepts from the domain and environment conceptual models being used in the application;
3. Refine domain feature models into the domain collaboration model based on features selected for final application;
4. Refine the domain collaboration models
   (a) Introduce collaborations with information/environment features in place of selected collaboration links from the domain collaboration models;
   (b) Introduce messages being sent between the participants in the collaborations and constraints for determination which message applies in a specific situation;
5. Update conceptual and feature models of domain and environment if new concepts and/or features have been developed.

3 Commonalities and Variabilities in Domain and Environment

Conceptual Models. Let us recall our application scenario. The company has an experience with serving a Java tutorial. Information usually exposes one or more concepts from a domain where general conceptual models, taxonomies or ontologies can already exist. For example Java tutorial serves an information which belongs to computer science domain. There are several taxonomies which are used for example to classify computer science literature (ACM CCS[1]) or to describe a computing body of knowledge and curricula[2]. Companies use also their own conceptual models to communicate terminology used in their information systems.

Figure 2 depicts an excerpt of such a *domain conceptual model* modelled by the UML class diagram with basic object-oriented programming concepts (annotated by the `Concept` stereotype) and their mutual relationships. The figure expresses a company's general view on relationships between `Object`, `Class`, object's `State` and `Behaviour` using `Methods` and `Variables`.

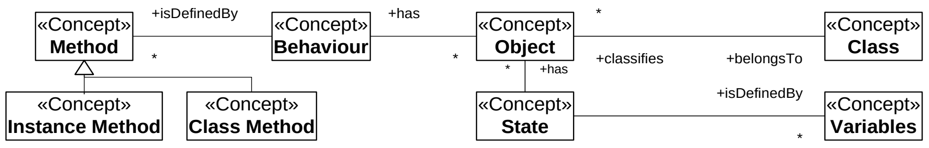


Fig. 2. An excerpt of conceptual application domain model

As the training suite is provided with several possible virtual environments, a company needs to communicate how the environments are structured. The example of such an environment suitable for the Java tutorial can be a virtual course. Concepts such as `Course`, `Lectures`, `Modules`, `Learning Object`, `Lecturer`, and

[1] http://www.acm.org/class/1998/
[2] http://www.computer.org/education/cc2001/

`Provider` would then appear in a similar UML class diagram for *environment conceptual model* (Fig. 3).

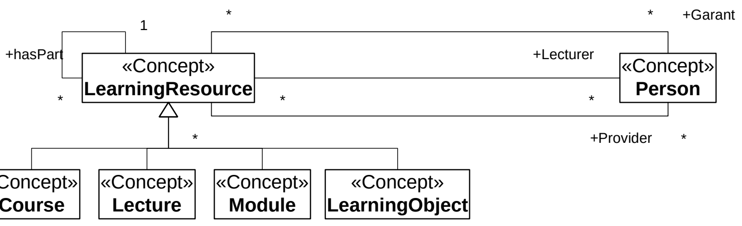


Fig. 3. An excerpt of conceptual environment/information model

Feature Models. According to our scenario, the company needs two different configurations of a content and several configurations of environment for two different audiences. Some features are common for both configurations and some vary.

As scenario pointed, the content is intended to be served through different environments which is selected according to requirements. Feature models have to be created for both views. The main elements in feature models are *concepts*, *features*, *variation points*, and *relationships* between them. A *concept* in feature models represents:

- *in a domain model* — an information, which is of the main purpose (main information goal) of the content which author had when authored the content,
- *in an environment/information model* — a main structural unit of a content in particular web-based application (different representations are modelled by different concepts).

A feature model has to be maintained for all concepts from conceptual model which are going to be depicted as main information entities in an environment. Figure 4 depicts an excerpt of such a feature model for the `Object` concept.

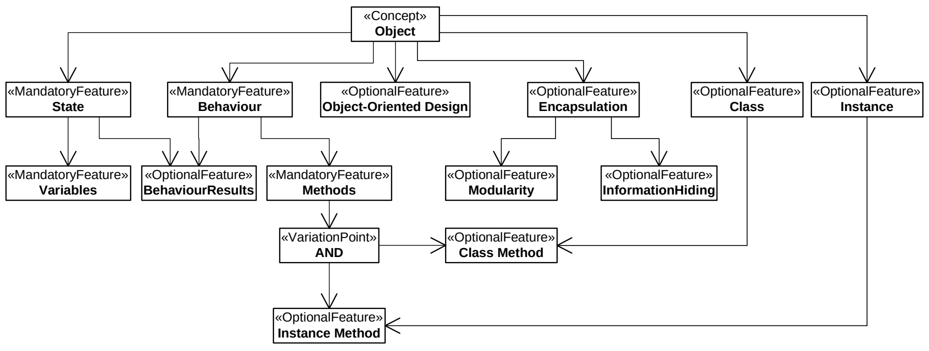


Fig. 4. An excerpt of *Object* feature model

Other concepts needed to communicate the main information goal represented by a concept in a feature model are selected and transformed to *features* of the concept. According to our scenario for example, the `Object` concept is usually described with the help of the concept of its `State` and `Behaviour`. Both appear as concepts in the conceptual model in Fig. 2. They appear as the `Object`-s features in Fig. 4.

All other concepts from the conceptual model (Fig. 2) have usually such feature models if they are communicated to learners as available in the application. For space limitation we do not show them here. Note also that the models depicted in our examples are not intended to provide a one and only solution, but just to exemplify how to create own feature models which report on best practices for information being served in web based applications.

The fact that there are some features which are common to all configurations and some vary has to be also reflected in the model. According to that we consider (in both, domain and information/environment models):

– *mandatory features* — form common or core features for all considered situations which will be covered in our applications (application family), and
– *optional features* — form variable features needed only in specific context.

In our example scenario from Fig. 4, the `State` and `Behaviour` are considered as mandatory features (annotated with `MandatoryFature` stereotype). The concepts annotated by `OptionalFeature` stereotype do not appear in all applications (e.g., `Object-Oriented Design`, `Encapsulation`, `Class`, and `Instance`).

Sometimes some features need to be presented together with other information features to provide sufficient explanations to understand presented information. Some other information features cannot be presented together because they could confuse a learner. In some cases the combination of features is not so relevant. For this purpose *variability relationships* have been introduced between features and they are usually denoted as *variation points* [22] or *variations* [9]. The variation point can define:

– *mutually exclusive variants,*
– *mutually required features, and*
– *mutually inclusive features.*

Figure 4 depicts a variation point shown for `Methods` mandatory feature. The model defines that the `Methods`have to be described also on `Instance Method` and `Class Method`.

Similarly, a feature model is needed for the information/environment concepts. Figure 5 depicts an excerpt of such a feature model of a virtual environment from our example scenario for the `Course` concept. Usually, information feature model for one virtual environment consists just of one feature model for the most general concept. The `Course` has to have a `Provider` and also a `Garant` (modelled by so called mandatory features). Then, according to requirements of a customer, the `Course` can consist of either `Lectures` where some of them can be encapsulated into thematic `Modules` or just from `Lectures` (this is reflected by the `OR` variation point of the `Course`). The `Lecture` can have a `Lecturer` provided who plays a role of a tutor when somebody needs to consult something related to the lecture. Both, `Lectures` and `Modules`, refer to learning objects.

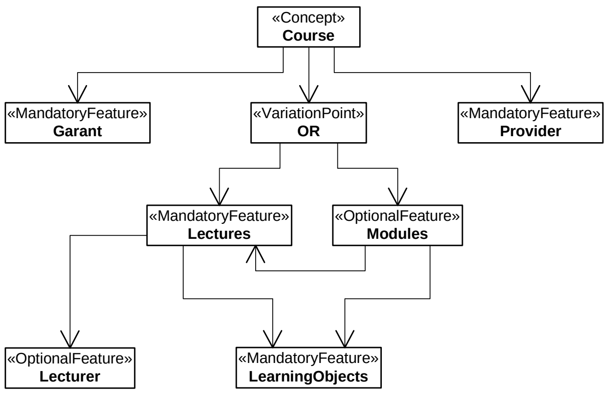


Fig. 5. An excerpt of *Course* feature model

4 Information Collaboration Modelling

In this section we will show just a collaboration model made according to a selection of features for the simple Java tutorial from our example scenario.

The Java tutorial provides a small lecture on introduction to the Java objects, which are introduced by it's state and behaviour. The features which are needed to communicate this introduction are depicted in Fig. 4. At runtime, the feature instances collaborate to create a content. The idea of collaboration is based on the notion of active learning objects which provide a defined interface to access their content and presentation. With this idea in mind, dependencies from the domain feature models can be transformed into collaboration links. The collaboration models are created as refinements of the feature models. The refinement consists of:

- Instantiating concepts and features from feature model;
- Identifying roles of the instances in collaborations;
- Transforming associations between features and concept in feature models into collaboration links.

Roles are used to model different purposes of a particular feature or concept in an environment/information component. We use the following notation:

$$O/R : P :: C$$

where O is a Classifier or Feature instance,
R is a Classifier or Feature role,
P (optional) is a Package name where Feature or Classifier belongs to, and
C is a Classifier or a Feature.

Roles terminology can form a complex structures. The UML class diagram can be employed to model such a structure [1]. This model can be used similarly to the domain and environment/information conceptual models; i.e. as a mean for communicating the roles terminology to be used in the web-based training suite.

Domain features collaborations. The first refinement of the feature models is refinement into domain features collaborations. The refinement is based on defining concept and feature *instances and roles* and *links* between them as instances of associations between the concept and features.

Instances and Roles. Figure 6 depicts an excerpt of a collaboration model of features from the `Object` feature model (the feature model depicted in Fig. 4). Roles of domain features used in the collaborations are *definition*, *example*, *exercise*, *description*, and so on. The `Definition` and `Example` roles played by the `Object` concept instances are used to model a situation where a term `JavaObject` is defined and then showed on the example. The term `JavaObject` is defined using the `JavaObjectState` and `JavaObjectBehaviour` definitions. The `Variable` definition is used define the state variables.

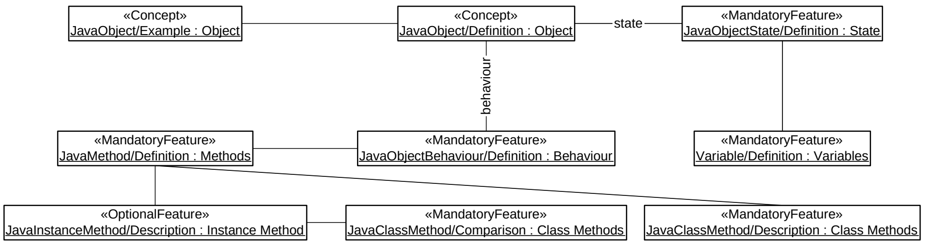

«Concept»
JavaObject/Example : Object
«Concept»
JavaObject/Definition : Object
state
«MandatoryFeature»
JavaObjectState/Definition : State
behaviour
«MandatoryFeature»
JavaMethod/Definition : Methods
«MandatoryFeature»
JavaObjectBehaviour/Definition : Behaviour
«MandatoryFeature»
Variable/Definition : Variables
«OptionalFeature»
JavaInstanceMethod/Description : Instance Method
«MandatoryFeature»
JavaClassMethod/Comparison : Class Methods
«MandatoryFeature»
JavaClassMethod/Description : Class Methods

Fig. 6. An excerpt of a collaboration model showing features from *Object* feature model

The `JavaObjectBehaviour` collaborates with the `JavaMethod` definition, as the behavior of an object is exposed by its methods. The `JavaInstanceMethod` and `JavaClassMethod` descriptions are provided as two alternatives for the Java methods (they play a description role in collaboration with the `JavaMethod` definition). In addition, a comparison of the class method and instance method is provided to improve an understanding of the difference between these kinds of methods. This is reflected by additional comparison role of the `JavaClassMethod` in collaboration with the `JavaInstanceMethod`.

Links. The collaboration links are defined mostly as straitforward mappings from associations in the feature model. Special attention has to be paid in case of several instances and roles of the domain features in the model. A designer has to decide which roles and instances will be linked together. In our example, two instances of the `Object` concept appear in the collaboration model. We created a link just between the `Definition` roles of the `JavaObject` and the `JavaObjectBehaviour`. The `Example` role of the `JavaObject` contributes just to the collaboration with `Definition` role of the `JavaObject` role in our example application, so the link to the `JavaObjectBehaviour` is not created. Similarly, the `AND` variation point in the feature model from Fig. 4 was resolved as several links between the participating feature instance roles.

Names of the links are suppressed in the model except of the `state` link between `JavaObject` and `JavaObjectState` and the `behaviour` link between `JavaObject` and `JavaObjectBehaviour`, which we will use for the purpose of description later.

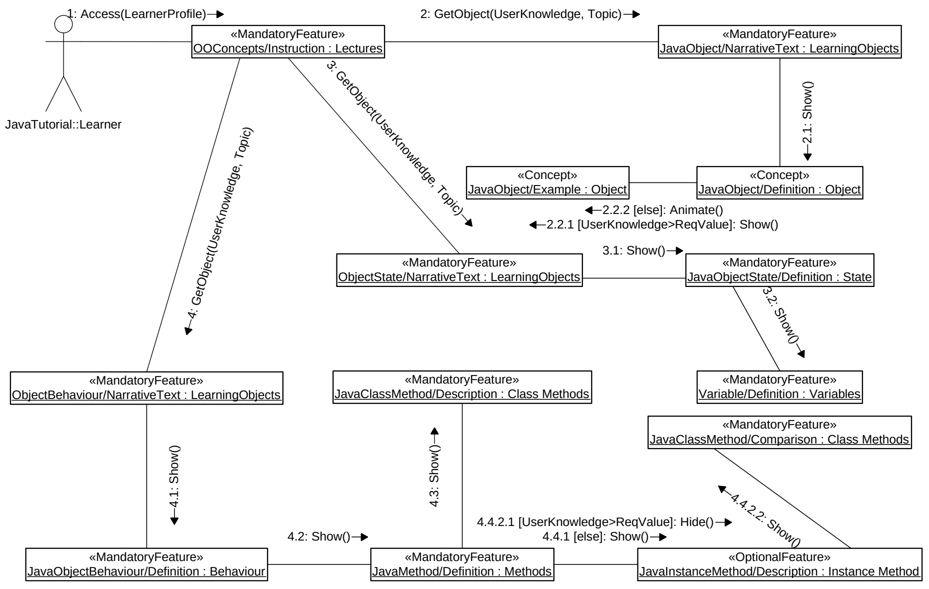

1: Access(LearnerProfile)
2: GetObject(UserKnowledge, Topic)
«MandatoryFeature» OOConcepts/Instruction : Lectures
«MandatoryFeature» JavaObject/NarrativeText : LearningObjects
JavaTutorial::Learner
3: GetObject(UserKnowledge, Topic)
2.1: Show()
«Concept» JavaObject/Example : Object
«Concept» JavaObject/Definition : Object
2.2.2 [else]: Animate()
2.2.1 [UserKnowledge>ReqValue]: Show()
4: GetObject(UserKnowledge, Topic)
3.1: Show()
«MandatoryFeature» ObjectState/NarrativeText : LearningObjects
«MandatoryFeature» JavaObjectState/Definition : State
3.2: Show()
«MandatoryFeature» ObjectBehaviour/NarrativeText : LearningObjects
«MandatoryFeature» JavaClassMethod/Description : Class Methods
«MandatoryFeature» Variable/Definition : Variables
«MandatoryFeature» JavaClassMethod/Comparison : Class Methods
4.1: Show()
4.3: Show()
4.4.2.2: Show()
4.4.2.1 [UserKnowledge>ReqValue]: Hide()
4.4.1 [else]: Show()
4.2: Show()
«MandatoryFeature» JavaObjectBehaviour/Definition : Behaviour
«MandatoryFeature» JavaMethod/Definition : Methods
«OptionalFeature» JavaInstanceMethod/Description : Instance Method

Fig. 7. An excerpt of a collaboration between feature from *Course* and *Object* feature model

Collaborations of domain features with information/environment features. Second refinement of the abstract feature models which follows the domain feature collaborations described above is the refinement into so called *information components*. By information components we mean accessible environment components which are accessed through their interfaces and are able to deliver concrete information about particular domain concepts.

This is achieved by linking the instances of the domain features to the instances of the environment/information features. The refinement extends above described collaboration models with:

- *Instance roles of environment/information features* to model mainly pedagogical style and purpose of the information component;
- *Links* between instance roles of environment/information features and domain features instance roles which replaces some of the links between domain feature roles (collaboration is being delegated to the information components);
- *Messages* being sent between environment/information feature roles and domain feature roles when a user performs an act of interaction.

Instance Roles of Environment/Information Features. Roles of information features in collaborations with domain features are usually *container*, *provider*, or more domain specific like *narrative text*, *simulation*, *problem statement*, *instruction*, and so on. Some of the roles are suitable just for higher level information objects like lecture or module. This is, for example, the *instruction* role which refers to the learning theory used to construct an information component. Similar concepts are used in the learning object metadata standard [12].

Figure 7 depicts an excerpt of such a collaboration model which links environment features to domain features. `NarativeText` role is introduced for the `JavaObject`, the `ObjectState`, and the `ObjectBehaviour` instances of `LearningObjects`. The learning objects can be accessed through the `OOConcepts` instruction which is an instance role of the `Lectures`. In a final training suite this lecture of Java tutorial collaborates with other lectures.

Additional Links — Collaboration Delegation. The three learning objects introduced in the Fig. 7 are linked to the corresponding domain features and/or concepts. The `JavaObject` is linked to the `Definition` role of the `JavaObject`. The `ObjectState` is linked to the `Definition` role of the `JavaObjectState`. The `Definition` role of the `JavaObjectBehaviour` is linked to the `ObjectBehaviour`. Note that these domain features are connected by the `state` and the `behaviour` links in the Fig. 6. These links are replaced by introduced environment feature instance roles and the collaborations are delegated to information feature roles accessible by a user (in this case the `OOConcept/Instruction`). Remaining links between domain features are derived from the Fig. 6. The collaboration links between environment/information features are created as instances of the associations in environment/information feature models. Similarly as in case of the domain features collaborations, collaboration links have to be resolved by a designer when several roles and instances of one feature or concept appear in the collaboration model and when variation point has to be transformed.

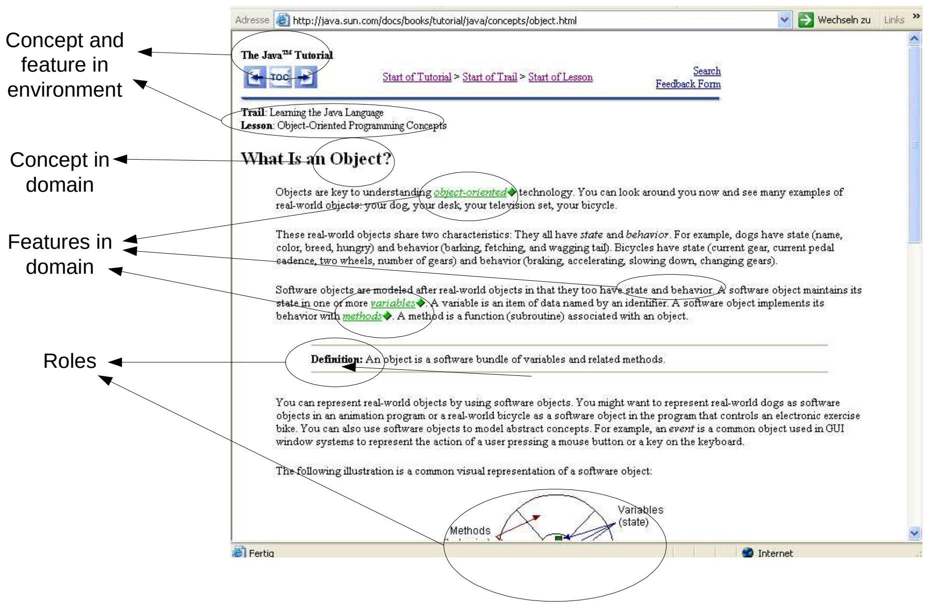
Messages. In addition, links are annotated by messages which are sent between the roles. User interaction generates the `Access(...)` message to the `OOConcepts` lecture. The `OOConcepts` lecture generates several `GetObject(...)` messages which are sent to the learning objects. Those learning objects request a content from the domain feature instance roles like `JavaObject` by sending for example the `Show()` messages. A content is propagated to the user as the result of interaction between the instance roles.

The `LearnerProfile` parameter of the `Access(...)` message is propagated by several `GetObject(...)` messages to domain features instance roles. Just fragments of the profile relevant to prerequisite competencies and/or topics of the learning objects are transferred by the `GetObject(...)` messages. These fragments are used to determine which presentation options are valid for a learner with particular level of knowledge, e.g. the two conditionally constrained messages at the link between the `Definition` and `Example` roles of the `JavaObject`. The `UserKnowledge` parameter is used to determine whether the example is just statically shown (`Show()`

message) or animated (`Animate()` message). The `ReqValue` is a predefined constant which sets which level of knowledge is required to switch between the presentation options. The conditionally constrained messages at the link between `JavaMethod` and `JavaInstanceMethod` can be interpreted similarly.

5 Applications and Experiences

Figure 8 depicts an example of the lesson or lecture from Java tutorial with some concepts used in our modelling approach.



http://java.sun.com/docs/books/tutorial/java/concepts/object.html

The Java™ Tutorial

Start of Tutorial > Start of Trail > Start of Lesson

Search
Feedback Form

Trail: Learning the Java Language
Lesson: Object-Oriented Programming Concepts

What Is an Object?

Objects are key to understanding *object-oriented* technology. You can look around you now and see many examples of real-world objects: your dog, your desk, your television set, your bicycle.

These real-world objects share two characteristics: They all have *state* and *behavior*. For example, dogs have state (name, color, breed, hungry) and behavior (barking, fetching, and wagging tail). Bicycles have state (current gear, current pedal cadence, two wheels, number of gears) and behavior (braking, accelerating, slowing down, changing gears).

Software objects are modeled after real-world objects in that they too have state and behavior. A software object maintains its state in one or more *variables*. A variable is an item of data named by an identifier. A software object implements its behavior with *methods*. A method is a function (subroutine) associated with an object.

Definition: An object is a software bundle of variables and related methods.

You can represent real-world objects by using software objects. You might want to represent real-world dogs as software objects in an animation program or a real-world bicycle as a software object in the program that controls an electronic exercise bike. You can also use software objects to model abstract concepts. For example, an *event* is a common object used in GUI window systems to represent the action of a user pressing a mouse button or a key on the keyboard.

The following illustration is a common visual representation of a software object:

Fig. 8. Modelling concepts of our approach highligted at a web-based Sun Java tutorial

The collaborations are presented as a web page showing content about several features we modeled in our models. There are two roles highlighted: the definition and example modeled for the Object concept in our feature model. Object-oriented design, behavior, state, methods and variables are similarly highlighted as the features from the domain. The lesson and course are also highlighted as environment features.

This information modelling approach resulted from several experiences with modelling and developing learning oriented web-based systems. First of all, it served as an underlying information modelling for the applications generated by the UML-Guide [8] where a very basic environment considered was environment of web pages.

In another EU/IST project — Elena[3] — we gained experiences in information modelling for information sources connected to a network and accessed through applications

[3] www.elena-project.org

making use of services such as recommendation, booking, and delivery. The information sources have been connected by Edutella [15], a p2p infrastructure for the semantic web. Resource Description Format (RDF) [13] was employed as the physical data model for metadata about information being served by the Edutella peers. The physical model for accessing the content varied. The metadata have been structured into two separated models, domain and information/environment. A subset of the ACM CCS system was used as concept ontology. Applications developed on top of the information provision infrastructure used defined interfaces encapsulating queries to application services and to the information provision infrastructure to provide tutorials, course, or learning resources through environments defined in the network.

6 Related Work

Recently, role oriented modelling was applied as a promising modelling approach for design and development of software applications [21,20] or databases [10]. The notion of role is used to identify different purposes of classes and objects in collaborations with different classes and objects available in a design framework. The importance of roles in modelling is apparent from the work on topic maps standard [16]. The concept of role is used in the topic map similarly to our approach: a topic plays particular role in association to other topics. Knowledge modelling community also reflects on notion of roles for example in [19]. There, the structural types can be associated to each other through roles the types play in the associations.

Feature modelling is another modelling technique which has been successfully applied in reuse oriented community [22,6,23,9]. The main idea is to explicitly represent information about different possible combinations of features a concept can have in software system for different purposes.

Our approach extended and integrated the ideas from the role oriented and feature oriented approaches for purposes of information modelling of information intensive web-based applications.

Tropos approach [4] relates to our work by considering variation of goals as a main concept which drives its requirements engineering phase. The goal oriented early requirements analysis can be suitable as a pre-step to feature modelling in our approach, as it is intended to provide a broader view for all interacting actors. Our approach also emphasizes on a conceptual model which is used as a vocabulary for the feature models documenting suitable configurations of information being provided in different environments. Refining technique of goal models into sequence diagrams in Tropos relates to our refinements from feature models into collaboration diagrams.

The information models in web applications modelling approaches usually reflect either an application domain point of view or environment point of view. OOHDM [18], W2000 [3], UWE [11], WebML [5], and UML extension for web engineering [14] model information from application domain perspective. ADM [2] concentrates on logical model of a web site where E-R model is used to specify data to be used in sites and page schema is used to specify pages. The work [17] relates to our approach by employed application domain.

Our approach introduces several new models, separating domain from environment modelling. Feature and collaboration models which are not considered in the mentioned approaches are used to represent additional design views needed in cases similar to our example scenario. This of course brings additional overhead arisen from creation and maintenance of those additional models. Therefore, our approach is beneficial when there is a knowledge or assumption that the models will help in other projects to reuse content and implementation from previous projects.

The approach introduced in this paper is highly complemental to those approaches introduced in recent web application design methods and can be integrated with them. The integration as pointed above have to be justified by the need to ease the future development projects.

7 Conclusions and Further Work

We discussed the UML-based feature modelling with the UML collaboration diagrams for information modelling for web-based applications in this paper. The core features of the approach are:

– Separation of domain and information/environment models;
– Making use of feature models in both design views to communicate best practices in information product line;
– Making use of the UML collaboration diagrams to configure a web-based information provision application with a specific virtual environment.

The models provide a designer of a web-based application with necessary information about which feature configurations make sense, what are important features to be considered and how their roles determine their use in provided environments.

In our further work we would like to investigate possibilities to automate some of the steps which now have to be performed by a designer. We also would like to further investigate how to improve a tool support for such information modelling described here. This comprises especially domain specific query language for more effective selection of features when building the collaboration models.

Acknowledgement. We would like to thank anonymous reviewers whose comments helped to improve the work presented in this paper. This work is partially supported by EU/IST ELENA project IST-2001-37264.

References

1. Heidrun Allert, Peter Dolog, Wolfgang Nejdl, Wolf Siberski, and Friedrich Steimann. Role-oriented models for hypermedia construction — conceptual modeling for the semantic web. Technical Report. Learninglab Lower Saxony, University of Hannover, February 2003.
2. Paolo Atzeni and Alessio Parente. Specification of web applications with adm-2. In Patrick van Bommel, editor, *Information Modelling for Internet Applications*, pages 127–143. Idea Group Publishing, 2002.

3. 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.
4. Jaelson Castro, Manuel Kolp, and John Mylopoulos. Towards requirements-driven information systems engineering: the Tropos project. *Information Systems*, 27(6):365–389, September 2002.
5. Stefano Ceri, Piero Fraternali, and Maristella Matera. Conceptual modeling of data-intensive web applications. *IEEE Internet Computing*, 6(4), August 2002.
6. Krysztof Czarnecki and Ulrich Eisenecker. *Generative Programing: Principles, Techniques, and Tools*. Addison Wesley, 2000.
7. Peter Dolog and Mária Bieliková. Towards variability modelling for reuse in hypermedia engineering. In Yannis Manolopoulos and Pavol Návrat, editors, *Proc. of Advances in Databases and Information Systems : 6th East European Conference, ADBIS 2002, Bratislava, Slovakia, September 8-11, 2002.*, vol. 2435 of *LNCS*, pages 388–400. Springer.
8. Peter Dolog and Wolfgang Nejdl. Using UML and XMI for generating adaptive navigation sequences in web-based systems. In Perdita Stevens, Jon Whittle, and Grady Booch, editors, *Proc. of UML 2003 - The Unified Modeling Language. Model Languages and Applications. 6th International Conference, San Francisco, CA, USA, October 2003,*, vol. 2863 of *LNCS*, pages 205–219. Springer.
9. 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.
10. Terry Halpin. *Information Modeling and Relational Databases: From Conceptual Analysis to Logical Design*. Morgan Kaufmann, 2001.
11. 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, vol. 1939 of *LNCS*. Springer.
12. IEEE Learning Technology Standards Committee. IEEE standard for learning object metadata (IEEE 1484.12.1–2002). http://ltsc.ieee.org/, July 2002.
13. O. Lassila and R. Swick. W3C Resource Description framework (RDF) Model and Syntax Specification, 2001. `http://www.w3.org/TR/REC-rdf-syntax/`.
14. David Lowe, Brian Henderson-Sellers, and Alice Gu. Web extensions to UML: Using the MVC triad. In S. Spaccapietra, S.T. March, and Y. Kambayashi, editors, *Proceedings of 21nd International Conference on Conceptual Modeling*, pages 105–119, Tampere, Finland, October 2002, vol. 2503 of *LNCS*. Springer.
15. W. Nejdl, B. Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmr, and T. Risch. EDUTELLA: a P2P Networking Infrastructure based on RDF. In *In Proc. of 11th World Wide Web Conference*, Hawaii, USA, May 2002.
16. Steven R. Newcomb, Sam Hunting, and Jan Algermissen. Reference model for ISO 13250 topic maps (RM4TM) v. 1.0, 2002. `http://www.isotopicmaps.org/rm4tm/`.
17. Joerg Pleumann and Stefan Haustein. A model-driven runtime environment for web applications. In Perdita Stevens, Jon Whittle, and Grady Booch, editors, *Proc. of UML 2003 - The Unified Modeling Language. Model Languages and Applications. 6th International Conference, San Francisco, CA, USA, October 2003,*, vol. 2863 of *LNCS*, pages 190–204. Springer.
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. John F. Sowa. *Knowledge Representation: Logical, Philosophical and Computational Foundations*. Brooks/Cole, 2000.

20. Friedrich Steimann. On the representation of roles in object-oriented and conceptual modelling. *Data and Knowledge Engineering*, 35(1):83–106, 2000.
21. Trygve Reenskau with P. Wold and O.A. Lehne. *Working with objects The OOram Software Engineering Method*. Manning/Prentice Hall, 1996.
22. 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`.
23. James V. Withey. Investment analysis of software assets for product lines, 1996. CMU/SEI-96-TR-010.