Defining a UML Profile for Web–based Educational Applications

Andreas Papasalouros
National Technical University of Athens
Software Engineering Laboratory
9 Heroon Polytechneiou, 15780, Zografou, Greece
andpapas@softlab.ntua.gr

Symeon Retalis
University of Piraeus
80 Karaoli & Dimitriou,
18534, Piraeus, Greece
retal@unipi.gr

Abstract

This paper presents a UML profile for web–based educational applications. The definition of the profile is provided by applying a certain formalism which is based on the metamodeling architecture of the UML language. An example is given throughout the paper for the illustration of an instantiated model based on the profile.

1. Introduction

The World Wide Web is becoming the basis of ubiquitous, learner–centered learning. Time and space independence and the ability for multiple representations of learning material due to enhanced multimedia presentation capabilities are among the characteristics of the Web that enforce its potential as a medium for education and training.

Web–based Educational Applications (WEA) are important components for web–based learning. By the term 'educational application' we mean properly structured and presented hypermedia material composed of educational resources and delivered through the web. The presentation of these resources is usually performed by specialized runtime environments, web applications or integrated systems for the support of learning through the web called Learning Management Systems. An trivial case of an educational application is an electronic book containing resources as web pages with educational content. The delivery of resource can be static or dynamic, that is, adapted to the user interaction with the application, though the latter case is not considered in this paper.

The development of WEA is a complex task. It involves people with different background such as software developers, web application experts, content developers, domain experts, instructional designers, etc. Furthermore, WEA are complex dynamic web–based applications with presentational, behavioral, architectural aspects. In order to effectively capture and specify the various aspects of a WEA, to document the decisions concerning various aspects of such applications, from the implementation of pedagogic and instructional design to subtle technical decisions of such applications and to facilitate the communication between the members of usually heterogeneous development teams there is a need for a design model. Experience from traditional software engineering has shown that the adoption use of such a model in the context of a systematic process for the development is beneficial for the quality of the product, namely educational applications and the efficient management of the resources, time and effort. Thus, a number of modeling approaches for web–based applications have been proposed, e.g. OOHDM [11], RMM [4], WebML [2]. Furthermore, specific models for educational applications exist. EML [5], which has evolved to the Learning Design specification [3] proposes a modeling formalism for educational environments. EML does not aim to support the design but rather to provide a conceptual framework for the description of educational settings as a set of "units of study". LMML [12] is another modeling language for educational content, which can be used for design in this domain.

This paper presents a formal definition of a modeling language for design models for WEA. This language is defined as a sub–set, i.e. a profile of the Unified Modeling Language (UML), a de facto standard for the modeling of software systems. It focuses on the formalism used for the definition of this new modeling language and not in the language itself, which is described elsewhere [10]. The purpose of the paper is to provide an overview and not a full specification of this language. The structure of the paper is as follows: In the next section a discussion of UML profiling and the approach adopted for defining this profile is presented. Next, the modeling language for WEA is described together with the definition of the profile for this language. The paper ends with a discussion of the consequences of formal modeling and some extensions of this work.

2. UML metamodels and profiles

The proposed modeling language is based on the UML. It is an extension of the UML by introducing new modeling elements and specified as a UML Profile. According to the UML specification a UML profile is “a coherent set of extensions, defined for specific purposes” [9]. More generally, profiling is the customization of a general purpose technology for a particular domain [6]. This is a process involving the elicitation of relevant only elements of the generic technology and the extension of the semantics of these elements towards the particular domain of application. In this way, the generic technology becomes more efficient and easy to apply.

The UML is a visual language. The definition of the language itself is based on a ‘Four–Layer Metamodel Architecture’ [9]. According to this approach, the definition of the language is structured in four layers: Meta–metamodel, metamodel, model and user objects. Each layer acts as a meta–language for the definition of the next layer. From another perspective, each layer is an *instance* of the previous one.

- The meta–metamodel layer provides the abstract constructs for the definition of the metamodel layer, e.g. MetaClasses, MetaAttributes, MetaOperations, etc. The definition of this layer is based on the Meta–Object Facility (MOF) [7] which is a specification for the definition and interchange of metamodels.

- The metamodel layer provides a language for specifying actual models. Examples of metamodel elements are Classes, Attributes, Operations, etc. Metamodel elements are represented as (meta) classes. The metamodel contains a set of UML class diagrams which contain metaclasses connected with association and generalisation relationships. Furthermore, for the definition of valid models additional rules and constraints must be provided. These rules and constraints are applied as “well–formedness” rules, which are also part of the UML metamodel. These rules are expressed in a Object Constraint Language (OCL) [13], which is also part of the UML specification, and, in few occasions, in natural language.

- The model layer contains instances of the modeling elements of the metamodel which are actually models of a particular application or domain, for example a class Course belongs to the model of a learning system and it is an instance of the Class element of the metamodel layer.

- Finally, the user objects layer defines specific information for the application domain into consideration, for example the particular state of a Course class instance.

Our purpose is to define a language for describing models for the application domain of web-based educational applications. The structure of these models and the concepts in use are designated by the application domain under consideration. As mentioned above, this language is defined by means of a UML profile. The establishment of a UML profile is provided by extending the metamodel layer in the four–layered architecture. This extension is provided by using the standard extension mechanisms of UML, namely stereotypes, tagged values and constraints. The basic extension mechanism, stereotypes, extends the existing UML elements, belonging to the metamodel layer, thus widening the vocabulary of the language. However, the mere definition of the UML extension mechanisms mentioned before does not provide a complete Profile, adequate for modeling a new domain by capturing both its concepts and their relationships in a consistent and meaningful manner. The approach that we follow for defining a profile for WEA is based on the UML Profile for CORBA Specification [8]. According to the latter, a profile consists of the following:

- The extension of a subset of the UML metamodel with new stereotypes and the assignment of these stereotypes to existing UML (metamodel) elements. This constitutes a “Virtual Metamodel”.

- The definition of additional of well–formedness rules, beyond these defined in the UML metamodel, for the new modeling elements. These rules impose additional constraints to the new elements so as to express the requirements of the particular domain. These rules refine and do not contradict with the rules that apply to their base metamodel elements, so as not to alter their semantics.

- The specification of the semantics of the model in natural language.

The well-formedness rules for the elements of the profile are defined as constraints expressed in the OCL and, in few occasions, in natural language. OCL is a language for the expression of constraints in UML model elements. As mentioned before, new constraints are defined for the new elements defined in the profile. The well–formedness rules expressed using OCL invariants. An invariant expression applies to a particular *context*. The context in which an invariant condition applies is denoted by the following expression which is in the start of every invariant:

```
context [Element Name] inv:
```

where `[Element Name]` is the name of the element under consideration to which the invariant applies. In order to define these rules proper OCL operations are needed. The operations used are OCL ‘built–in’ operations, operations defined in specification documents such as [9] and [8] or new operations defined for the purpose of defining the profile. These operations are presented next. The OCL expressions which define these operations are referring to the UML Relationships metamodel. The `aggregatedClasses` operation returns the set of classes (classifiers in the UML metamodel more generic terminology) that are connected with the class under consideration through composite aggregation associations:

```
aggregatedClasses: Set(Classifier)
aggregatedClasses =
 self.associations->
  select(a | a.connection->
   select (ae |
    ae.participant = self
     and
    ae.aggregation =
     #composite))->
  collect (ae |
   ae.connection->
    select(ae |
     ae.participant <> self)->
      collect(p |
       p.participant)).asSet
```

The `aggregateClasses` operation returns a set of Classes that are at the opposite side of aggregation associations, i.e. they contain the particular class:

```
aggregateClasses: Set(Classifier)
aggregateClasses =
 self.associations->
  select(a | a.connection->
   select (ae |
    ae.participant = self ))->
  collect (ae |
   ae.connection->
    select(ae |
     ae.participant <> self
      and
     ae.aggregation =
      #composite))->
       collect(p |
        p.participant)).asSet
```

3. Description of the profile for WEA

In the next sections the following sub-models are defined as UML Packages: Conceptual Model, Navigation Model and User Interface Model.

3.1. The Conceptual Model

The Conceptual Model captures design decisions during Instructional Design. These decisions are described as a hierarchy of learning activity and associated resources. An example of such a model is illustrated in Figure 1.

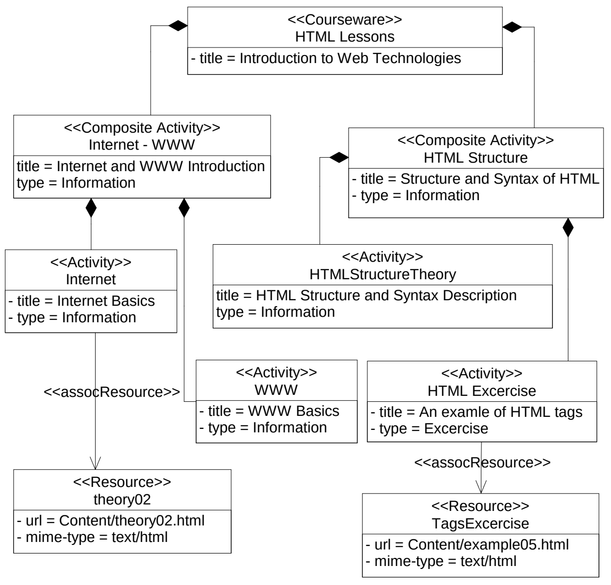


Figure 1. Example of a Conceptual Model

The virtual metamodel for the conceptual model is shown in Figure 2. In this figure the Conceptual Model is shown as a class with stereotype named `stereotype`. This (meta)class is connected with a dependency relationship with stereotype `baseElement` with a base class named `ModelManagement::Model`. This configuration denotes the fact that the element Conceptual Model ‘instantiates’ the element Model of the UML metamodel, i.e. it is a Model extended by stereotyping. The prefix `ModelManagement` states the fact that the metaclass Model belongs to the ModelManagement package of the UML metamodel. Figure 3 shows the elements of the Conceptual Model, namely `Activity`, `Composite Activity`, and `Resource`, which are stereotyped classes. `Abstract Activity` is a generic type of activity defined as an abstract metaclass, which means that `Abstract Activity` does not appear as a modeling element. The subclassing mechanism is used for organisation purposes. Rules and constraints applied to superclasses are inherited to the subclasses, unless overridden by new rules applied to the descendant classes. Composite activities contain other activities by connecting with them with aggregate associations. In this way, hierarchies of activities are defined in the Conceptual Model. A special type of composite activity with stereotype `Courseware` is the root in the activity hierarchy. In addition, atomic activities are related with one or more Resources with `AssocResource` associations (See Figure 4).

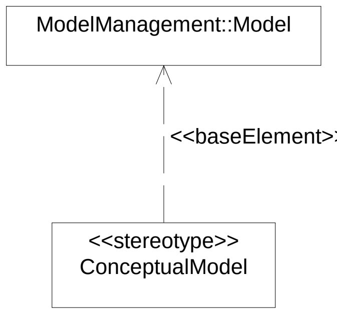


Figure 2. Conceptual Virtual Metamodel

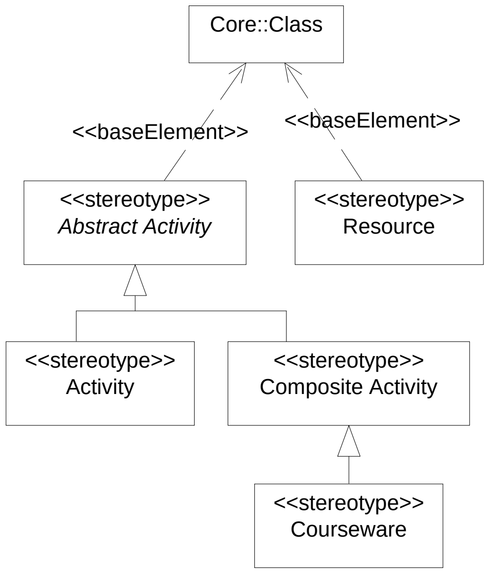


Figure 3. Conceptual Model Classes Virtual Metamodel

- The Conceptual Model contains only Activity and Resource classes

```
context ConceptualModel inv:
 self.ownedElement->forAll(e |
  e.oclIsKindOf(Class)
   implies (
e.isStereokinded("AbstractActivity")
 or
e.isStereokinded("Resource"))
```

- Abstract Activities have a `name` and a `type` attribute.

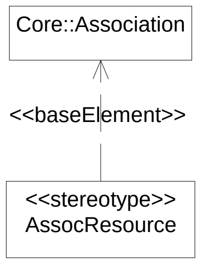


Figure 4. Conceptual Model Associations Virtual Metamodel

```
context AbstractActivity inv:
 self.features->
  select(a |
   a.oclIsKindOf(Attribute)
   and
   a.name = 'name').
    size = 1
 and
 self.features->
  select(a |
   a.oclIsKindOf(Attribute)
   and
   a.name='type').
    size = 1
```

- A Resource has a `mime-type` and a `url` attribute. The expression for this constraint is syntactically similar to the above and it is omitted.

- Composite Activities are associated (through aggregation associations) with activities, either atomic either composite.

```
context CompositeActivity inv:
 self.aggregatedClasses->forAll(c |
c.isStereokinded("AbstractActivity"))
```

- Atomic activities do not contain other classes through aggregation associations.

```
context Activity inv:
 self.aggregatedClasses.isEmpty
```

- A Courseware is not contained by any other classes through aggregation associations.

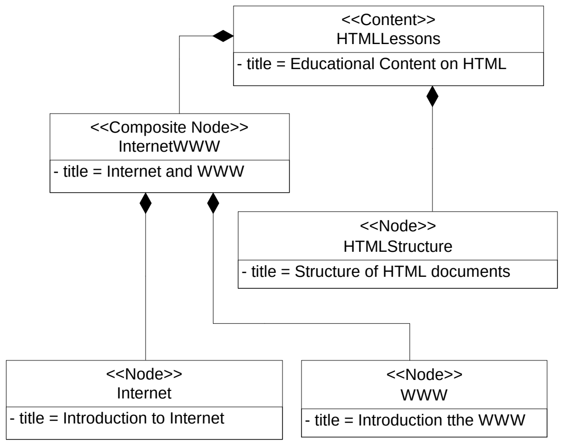
```
context Courseware inv:
 self.aggregateClasses.isEmpty
```

- `AssocResource` associations connect one `Resource` stereotyped class with a `Concept` stereotyped class.

```
context AssocResource inv:
 self.connection.size = 2
 and
 self.connection.participant->
  exists(c |
   c.isStereokinded("Activity"))
 and
  exists(c |
  c.isStereokinded("Resource"))
```

3.2. Navigation Model

The Navigation Model captures the design decisions concerning the definition of the actual web pages of the learning content, as well as the links that facilitate the navigation through these pages.



<<Content>>
HTMLLessons
- title = Educational Content on HTML
<<Composite Node>>
InternetWWW
- title = Internet and WWW
<<Node>>
HTMLStructure
- title = Structure of HTML documents
<<Node>>
Internet
- title = Introduction to Internet
<<Node>>
WWW
- title = Introduction tthe WWW

Figure 5. Example of a Navigation Structure Model

The navigation model is separated with two sub-models: The Navigation Structure Model deals with the definition of the actual web pages, the links that facilitate navigation as well as the definition of the actual content of these pages. The Navigation Behavior Model defines the dynamic, adaptive behavior of the WEA. In this version of the profile deals with static applications only so the Behavior Model is not considered. The Navigation Structure Model defines the (static) navigation structure of a WEA by means of the following elements: `Content`, which is a top-level container, `Composite Node`, which is a composite elements containing other elements, such as a chapter in an electronic book, `Nodes`, which are the actual web pages. Again, `Abstract Node` is an abstraction of a composite or atomic Node. The structure of the nodes is hierarchical, as the Conceptual Model. In addition, `Link` elements can connect nodes of the navigation model. Note that these links are associative links that define arbitrary navigation between nodes of the educational hypertext. Besides these, implicit structural links exist. These links denote the transitions to the previous or next node in the traversal of the hierarchical structure of nodes or the selection of a certain node. The presentation of these links is supported by the run–time system which delivers the educational application. Although both conceptual and navigational model diagrams are hierarchical, the corresponding graphs are not necessarily isomorphic. Nodes are realisations in the navigation space of one or more learning activities. Thus, one node can be related to one or more activities. Figure 5 illustrates an example of a Navigation Structure Model, which is an instantiation of the above virtual metamodels. The trace relationships between elements of this diagram and the Conceptual Model are not depicted, since they are obvious from the names of the elements.

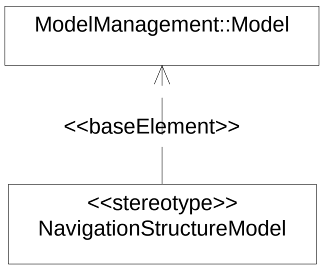


Figure 6. Conceptual Virtual Metamodel

The following constraints are applied to elements of the Navigation Structure Model:

- As mentioned above, the structure of nodes is hierarchical, as is the Conceptual Model. Thus, similar constraints apply to its elements as regards to aggregation. A `Composite Node` stereotyped class is connected through aggregation associations with `Abstract Node` classes, `Content` class has no aggregate classes, i.e. it is not contained by any other node, and, finally, atomic nodes do not contain other nodes. The OCL expressions for these rules are omitted for the sake of brevity.

- The Navigation Model contains only `Abstract Node` classes.

- A `Link` stereotyped association connects Nodes. A link represents a unidirectional link from a source to one or more destinations. It denotes the direction of

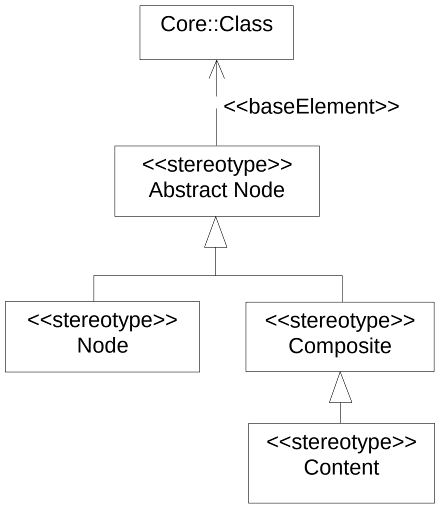


Figure 7. Navigation Model Classes Virtual Metamodel

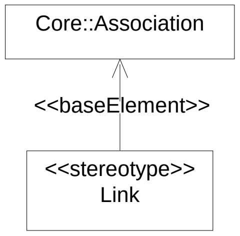


Figure 8. Navigation Model Associations Virtual Metamodel

a link in the following way: In the UML metamodel an association contains two or more association ends. These ends have a navigability attribute. Navigability is graphically depicted with an arrow at the navigable end of the association. The navigability of the association end, that is the direction of the arrow in the visual presentation of the association, denotes the direction of the link. Thus, there is exactly one end which is not navigable, which corresponds to the “source” of the link and one or more navigable ends, which correspond to the “destination” of the link.

```
context Link inv:
 self.connection.participant->
  forAll( p |
   p.isStereokinded("Node"))
 and
 self.connecions->
  select(ae |
     ae.isNavigable = false).
      size = 1
```

- `Abstract Node` elements have an attribute named `title`.
- An `Abstract Node` is related to one or more `Abstract Activity` elements of the Activity Model with a Trace dependency. This is a standard UML element, a special kind of relationship which depicts dependencies model elements between different models. This constraint is provided in natural language only.

3.3. User Interface Model

The User Interface Model deals with the presentation aspects of the elements defined in the Navigation Model i.e. their layout and style. In particular, nodes of the Navigation Model are associated with an element of the User Interface model. Multiple navigation elements can be associated with the same presentation specification, thus promoting uniformity and maintenability of the user interface. The User Interface Model elements have their counterparts in corresponding elements of web technology specifications, namely HTML and Cascading StyleSheets (CSS) elements. The basic element in this model is named `Template`. A template contains the following stereotyped UML classes with aggregation associations: `Html`, that represents HTML elements or aggregations of HTML elements and `Css` that represent Cascading Style Sheet classes. Attributes of these classes and their values are attributes and values of CSS elements. Elements defined in the User Interface model are related with Trace dependencies to particular nodes of the Navigation Model thus assigning specific presentation attributes to these nodes, as well to their children in the navigation structure hierarchy. No metamodel diagrams are shown, since they are similar to those of the Conceptual and Navigation Structure models. Figure 9 shows an example of a User Interface Model for the HTML educational application under consideration. The trace relationship between the template and the root element of the navigational model means that all nodes (pages) share the same template.

The following constraints are applied to the User Interface Model elements:

- The Presentation Model contains `Template`, `Css` and `Html` classes.
- `Template` classes contain, through composite aggregations, `Css` and `Html` elements.
- `Template` classes are connected with Abstract Nodes of the Navigation Model through `Trace` relationships.

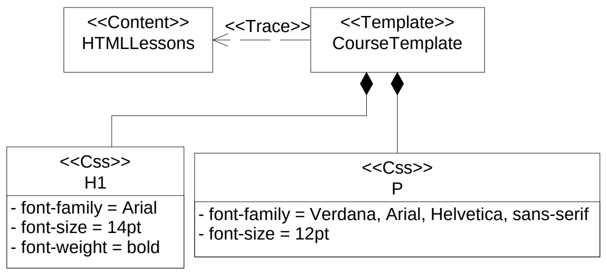


Figure 9. Example of a User Interface Model

4. Discussion

The definition of a Design Model can facilitate the process of developing software projects, regardless of the domain of the application. This facilitation is even more important in fields where the people involved in the development process come from different backgrounds so there is an increased need for a means of communicating design decisions. However, this improvement is often confuted by the lack of formalism in the definition of such models. This lack of formalism has certain negative aspects:

- Poorly defined models, which are based on the intuition of the designers rather than in predefined 'rules'.
- It is impossible to automate the authoring of models by means of specific CASE tools.
- It is impossible to automate the process of automatic code generation based on the models created (forward engineering).

Rigor definitions of modeling languages alleviates the aforementioned problems. A definition of a modeling language is usually provided by means of a metamodel, i.e. a model of the set of all valid models in the language. Such a MOF-based metamodel for web application design is proposed in [1]. Furthermore, EML and LMML presented earlier, are defined by means of a metamodel, i.e. a normal UML model that describes the modeling primitives of the language. Besides modeling primitives, EML and LMML provide an XML definition and provide appropriate schemas, as an additional formalism. Conversely, our approach is based on the extension mechanisms of UML for the creation of a profile, thus it does not define a new metamodel but rather refines the existing metamodel of the language using a specific formalism.

We are in the process of extending the definition of the profile so as to include dynamic features of WEA. Furthermore, this profile was defined based on version 1.5 of the UML and OCL. We intend to update it by using the version 2.0 of these languages.

5. Acknowledgements

This work was partially supported by the "TELL: Towards effective network supported collaborative learning" project which is partly sponsored by the European Commission under e-learning program (ref num: 2003-4721 TELL).

References

[1] L. Baresi, F. Garzotto, L. Mainetti, and P. Paolini. Metamodeling techniques meet web application design tools. In R.-D. Kutsche and H. Weber, editors, *FASE*, volume 2306 of *Lecture Notes in Computer Science*, pages 294–307. Springer, 2002.
[2] S. Ceri, P. Fraternali, and A. Bongio. Web Modeling Language (WebML): a modeling language for designing web sites. *Computer Networks*, 33(1-6):137–157, 2000.
[3] IMS. *IMS Learning Design Information Model, Version 1.0 Final Specification*, Jan. 2003.
[4] T. Isakowitz, E. A. Stohr, and P. Balasubramanian. RMM: A methodology for structured hypermedia design. *Communications of the ACM*, 38(8):34–44, Aug. 1995.
[5] R. Koper. *Modeling Units of Study from a Pedagogical Perspective: The Pedagogical Meta-Model Behind EML*, 2001.
[6] R. Malveau and T. J. Mowbray. *Software Architect Bootcamp*. Prentice Hall PTR, Upper Saddle River, NJ, 2001.
[7] OMG. *Meta Object Facility (MOF) Specification, Version 1.4*, Apr. 2002.
[8] OMG. *UML Profile for CORBA, Version 1.0*, Apr. 2002.
[9] OMG. *Unified Modeling Language Specification, Version 1.5*, Mar. 2003.
[10] S. Retalis and A. Papasalouros. Designing and automatically generating educational adaptive hypermedia applications. *Educational Technology & Society*, to appear.
[11] D. Schwabe and G. Rossi. The object-oriented hypermedia design model. *Communications of the ACM*, 38(8):45–46, 1995.
[12] C. Süß and B. Freitag. Metamodeling for web-based teachware management. In P. P. Chen, D. W. Embley, J. Kouloumdjian, S. W. Liddle, and J. F. Roddick, editors, *Advances in Conceptual Modeling: ER '99 Workshops on Evolution and Change in Data Management, Reverse Engineering in Information Systems, and the World Wide Web and Conceptual Modeling, Paris, France, November 15-18, 1999, Proceedings*, volume 1727 of *Lecture Notes in Computer Science*, pages 360–373. Springer, 1999.
[13] J. Warmer and A. Kleppe. *The Object Constraint Language: Precise Modeling with UML*. Addison Wesley, Reading, MA, 1999.