Hamburg University of Technology

”Model-Driven Development of a Collaborative Web Application”

Master’s Thesis
by Rizki Nugraha Pratama ICS / 31461
Software, Technology & Systems Group (STS) - TUHH

Supervisor:
Prof. Ralf Möller
Prof. Friedrich Mayer-Lindenberg
Adviser:
Miguel Garcia

May 2007

Abstract

The project aims at building up know-how on the upcoming “model compilers”, i.e. software engineering tools that generate (most of) a software application taking high-level, yet precise, specifications of behavior as input. In order to validate the approach, the development of a collaborative web application is taken as case study. The web application is developed with a model-driven tool called WebRatio which is based on WebML (Web Modelling Language). While developing the web application this project aims to explore the capabilities of WebML and WebRatio in designing a collaborative web application, and discover areas for improvement. It is found that the current specification of WebML has limited expressiveness for access control policies, which is evident in an environment of fine-grained sub-roles typically existing in social networking applications. An improvement to WebML is proposed in the form of adding authorization constraints to WebML units in the hypertext model using the WebML-OQL query language. This approach follows a similar approach taken in the UML-based SecureUML (another domain-specific language also exhibiting a model compiler). In order to incorporate the proposed authorization constraints into WebML, the WebML metamodel is extended with additional language constructs. The concrete syntax of WebML is also augmented to support authorization constraints, both in the graphical and textual representations. The addition of authorization constraints enables developers to specify arbitrary levels and constellation of sub-roles according to the business logic of the application.

1 Introduction

The project aims at building up know-how on the upcoming “model compilers”, i.e. software engineering tools that generate (most of) a software application taking high-level, yet precise, specifications of behavior as input. Model compilers targeting the platform Java Enterprise Edition (Java EE, defined in JSR-220) are very popular and include: DASL [Gol05], SecureUML [BDW06], and WebML [Cer06].

In order to validate the approach, the development of a collaborative web application is taken as case study, which will allow drawing comparisons with established development processes followed by practitioners. A collaborative web application is a data-intensive social networking application, with which users can communicate, exchange data and form virtual social groups. The target audience for this collaboration platform is an alumni network, as for example the graduates of a university. The web application is developed with a model-driven tool which uses WebML (Web Modeling Language) [CFBBCM03], a conceptual language designed for building data-intensive web applications that originated in academia. WebML provides graphical yet formal specifications in the framework of a complete design process. The tool supporting such language is WebRatio [WR]. While developing the web application this project aims to explore the capabilities of WebML and WebRatio in designing a web application, and discover areas for improvement. An extension or enhancement is then proposed to improve WebML in the discussed areas.

The report is organized as follows. Chapter 2 gives an overview of the technologies used by this project. The third chapter describes a case study in model-driven development of a collaborative web application. Chapter 4 describes the areas of improvements of WebML. The chapter afterwards discusses a proposed extension to WebML in order to improve it in the areas discussed. In Chapter 6, an overview of other work related to this project is given. The last chapter concludes the report and summarizes the achievements of this project.

2 Fundamentals

This chapter gives all necessary definitions and describes all technical terms and concepts used in the project.

2.1 Model-driven development

In [KW05] six Modelling Maturity Levels (MMLs) in software development are described. MML 0 denotes no software specification, only source code, and MML 5 means that the software is specified purely by models. Model-driven software development is targeted to reach the level 4 (MML 4) of this definition.

At MML 4, the specification of the software is described in one or more models. The models are precise enough to have direct link to the actual source code. Changes are done to the models and they are directly reflected in the regenerated source code. This in effect keeps the models up to date with the actual source code and allows agile, incremental development.

2.2 J2EE

The Java 2 Platform, Enterprise Edition (J2EE) Specification [J2EE] is designed as an extension to the Java 2 Platform, Standard Edition (J2SE) Specification [J2SE] to incorporate the needs of software applications deployed in enterprises. Such applications typically require among others platform-independent portability, high availability, scalability, reusability and modularity [JS05]. The architecture of the J2EE Specification is shown in Fig. 1.

The J2EE architecture uses a four-tier approach, consisting of `Client`, `Web`, `Business`, and `Enterprise Information` tiers. The `Client` Tier provides support for a large number of client types and allows access to other server-based tiers. The `Web` Tier and the `Business` Tier, collectively known as the `Middle` Tier, provide a set of services to help the rapid deployment of enterprise applications. The `Enterprise Information` Tier comprises the databases, ERP applications and file systems.

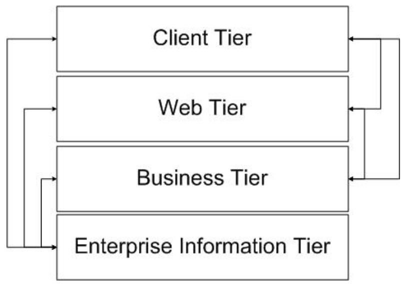


Fig. 1. J2EE Architecture

2.3 OCL

The Object Constraint Language (OCL) is a textual modeling language based on a mathematical (logical) syntax for object-oriented models. OCL adds information involving among others the constraints of objects which otherwise cannot be expressed within UML diagrams. [WK03] Its standard is published by the Object Management Group (OMG) as an add-on to UML. The current version, OCL 2.0 conforms to UML 2.0 [OCL2].

Examples of OCL can be shown in the following simple scenario: in a university, a `Student` can participate in one or more `Classes`. The UML class diagram representing this scenario is shown in Fig. 2. The UML diagram itself is not enough to describe some constraints that apply for such a scenario. These constraints may include:

- A `Student` must have a minimum total of 30 `ECTSpoints` from all the `Classes` he or she participates in
- A `Class` can only be attended by a maximum of 20 `Students`.

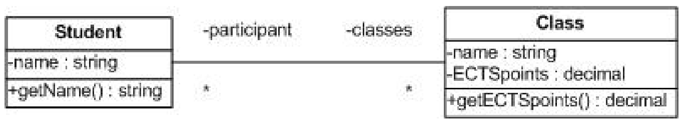


Fig. 2. Student-Classes scenario

OCL can be used to define those constraints, as shown in the example Fig. 3. They are described in the form of invariants, or conditions that are persistent and must be fulfilled throughout the lifetime of the objects. Invariants may also state rules for associated objects. Besides stating invariants, OCL can also be used to describe: initial values, derivation rules, query operations, and definitions of new attributes, operations, as well as pre- and post-conditions for operations [WK03].

```
/* A Student must have a minimum total of 30 ECTS points from all
Classes participated */
context: Student
inv minPoints: classes.ECTSpoints->sum() >= 30
/* A Class may only be attended by a maximum of 20 Students */
context: Class
inv maxParticipants: self.participants-> size() <= 20
```

Fig. 3. OCL Expressions

Formatted: Bullets and Numbering

2.4 RBAC

Role-based access control (RBAC) is described in [FK92]. It is argued that RBAC fulfills the access control requirements of commercial and industry applications better than the current standard at that time, the Discretionary Access Control (DAC). While DAC works by giving rights to individual users to grant and revoke access privileges to objects assigned to them, RBAC defines access rights according to the roles an individual user take in an organization. In commercial and industry applications, information belongs to the organization and functions are performed by individuals belonging to roles, rather than to single users. A role is viewed as a set of operations or transactions that is performed by a group of individuals. Since commercial and industry organizations typically consist of functions or hierarchical lines performing exclusive tasks, RBAC is more appropriate for this purpose.

A formal definition of RBAC consisting of four declarations and three basic rules [FK92] is shown in Fig. 4. The first three declarations are as follows: for each subject, the active role is the one that the subject is currently using; each subject may be assigned to one or more roles; and each role may be granted performing one or more transactions. The fourth declaration states that the predicate $exec(s,t)$ is true if subject s can execute transaction t at the current time, otherwise it is false.

The first basic rule states that a subject can execute a transaction only if the subject has been assigned a role. While authorization and identification processes such as login do not constitute a transaction, for others a subject must have an active role. The second rule requires that the active role assigned to a subject must belong to its authorized roles. Both rules combined warrant that users can only take roles for which they are authorized. The third rule mentions that a subject can only execute a transaction only if the transaction is authorized for the subject's active role. All three rules in combination guarantee that a subject can only execute transactions for which they are authorized.

```
Declarations:
AR(s:subject) = {the active role for subject s}
RA(s:subject) = {authorized roles for subject s}
TA({r:role}) = {transactions authorized for role r}
exec(s:subject, t:tran) = true iff subject s can execute transaction t
Rules:
```

1. Role assignment: $\forall s: subject, t: tran(exec(s,t)) \Rightarrow AR(s) \neq 0$
2. Role authorization: $\forall s: subject(AR(s) \subseteq RA(s))$
3. Transaction authorization: $\forall s: subject, t: tran(exec(s,t) \Rightarrow t \in TA(RA(s)))$

Fig. 4. Formal definition of RBAC

Formatted: Bullets and Numbering

2.5 SecureUML

SecureUML is a modeling language designed for model-driven development of secure distributed systems. SecureUML defines a set of vocabulary relevant to access control to annotate UML-based models. The vocabulary is based on the RBAC model, thus specifying roles, role permissions and user-role assignments, and in addition to the RBAC model it also specifies authorization constraints [LBD02]. The authorization constraints include information relevant for access control such as the state of a protected resource, current system time and parameter values, which are not described in the UML diagram. The authorization constraints are expressed in OCL and annotated into an extended UML diagram as shown in the SecureUML metamodel of Fig. 5.

The SecureUML metamodel introduces RBAC concepts to the UML metamodel, represented as metamodel types `User`, `Role`, and `Permission`. Some elements represented with UML can be given the status protected objects. The `ResourceSet` type is additionally introduced to represent a user-defined set of model elements used to define permissions or authorization constraints.

A `Permission` is a relation which connects a `Role` to a `ModelElement` or a `ResourceSet`. `Permissions` are classified by `ActionType` elements, which in turn define the semantics of a `Permission`. Every `ActionType` represents a class of security-relevant operations on a certain protected resource. The set of `ActionTypes` available can be freely specified using `ResourceType` elements. A `ResourceType` defines all the `ActionTypes` available for a specific metamodel type, whose name is identified in the `baseClass` attribute of the `ResourceType`.

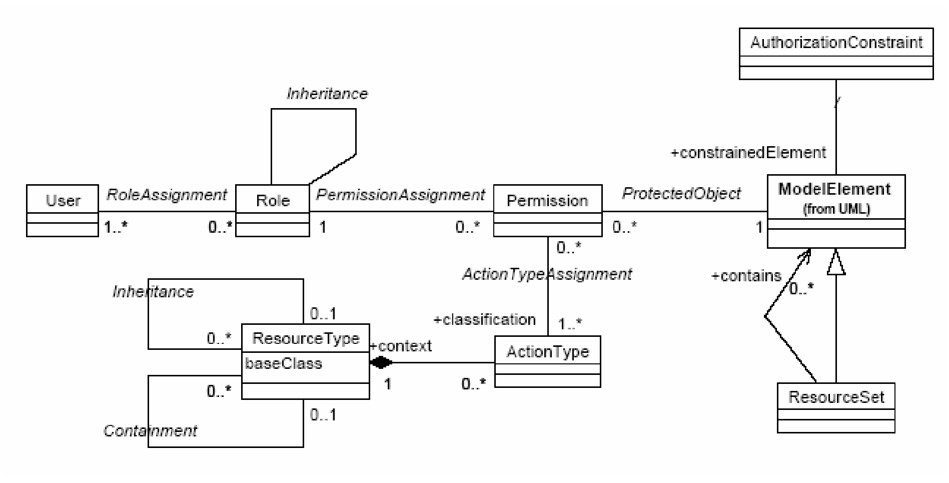


Fig. 5. SecureUML Metamodel

An `AuthorizationConstraint` expresses a precondition imposed on every call to an operation of a particular resource. The preconditions usually depend on the dynamic state of the resource, the current call, or the environment. `AuthorizationConstraint` is derived from the core UML type `Constraint`, and can be attached directly to a model element or indirectly via a `Permission`.

2.6 WebML

The Web Modeling Language (WebML) is a modeling language which describes the composition, navigation, and content of hypertext applications. The language is geared towards providing a high-level specification of web applications, which can then be transformed into executable code by means of a CASE tool. The specifications are textual and can be visualized in diagrams, drawing from the concepts of Entity Relationship Diagrams and UML.

The following models are defined in WebML: the data model, hypertext model, and content management model [CFBBCM03]. These models correspond to elements of the WebML development model which is based on Boehm's spiral model [Boe88], shown in Fig. 6. The self-explanatory diagram shows the incremental spirals of development leading to the completion of the application development. With respect to this diagram, the data model conforms to the `Data Design` element while the hypertext model and the content management model are used in the `Hypertext Design` element.

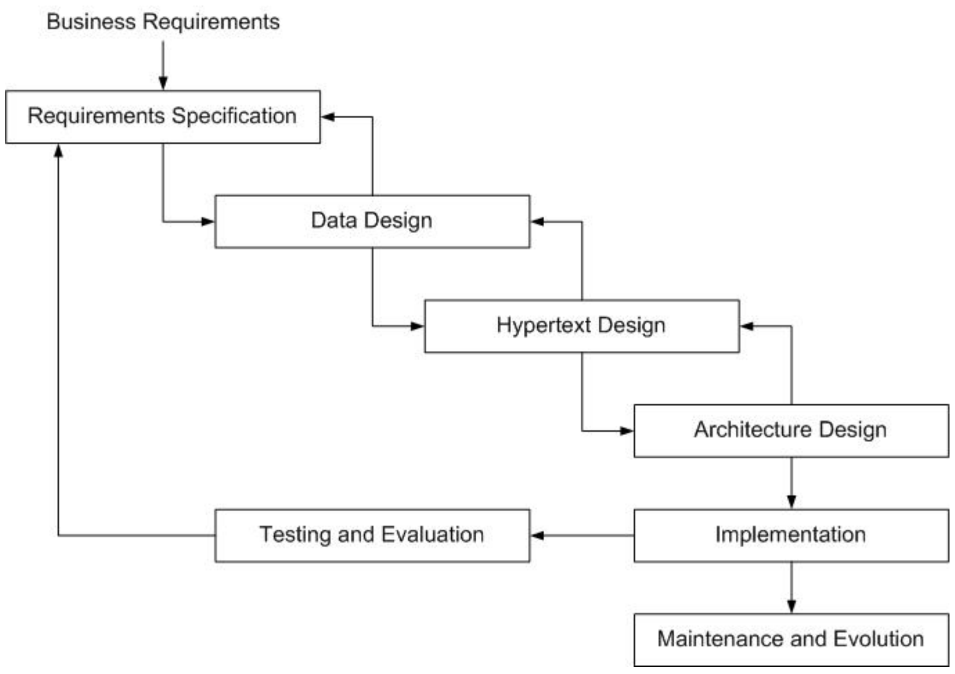


Fig. 6. The WebML Development Model

The data model consists of `entities`, which are containers of data elements, and `relationships`, which describe semantic relations between elements. `Entities` may have properties called `attributes` which are associated to a data type and may inherit another (parent) `entity`'s properties through generalization. `Relationships` can be constrained by means of cardinalities. Derivation of `attributes` is made possible by exploiting `relationship` traversal and manipulation of existing `attributes`, and the derivation rules are expressed in a syntax drawing from OCL, called WebML Object Query Language (WebML-OQL). The model draws from and as a result is compatible to Entity Relationship Diagrams and UML Class diagrams. An example of a data model is shown in Fig. 7.

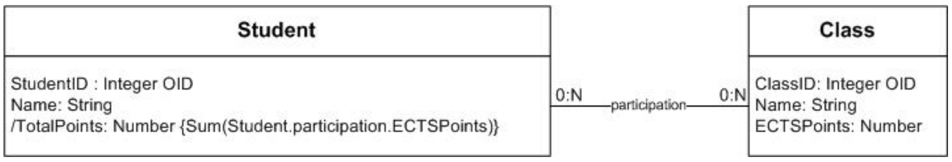


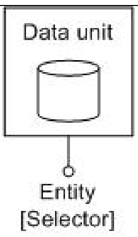
Fig. 7. WebML Data Model example

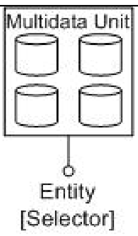
The data model is based on the scenario of Student-Classes mentioned in Section 2.3. There are two entities, `Student` and `Class`, linked by a relationship named `Participation` which effectively represents two relationships, `StudentToClass` and `ClassToStudent`. The cardinality of both relationships represented are `0:N`, which means that a `Student` can participate in multiple classes or no classes at all, and a `Class` can be attended by zero or more `Students`. The `Class` entity has three attributes, namely `ClassID`, `ClassName` and `ECTSPoints`, while the `Student` entity has attributes named `StudentID` and `Name` as well as a derived attribute called `TotalPoints`. Each attribute is typed according to the WebML primitive types [WR-WMLg] shown after the colon in the name. The derivation rule for `TotalPoints` is the sum of all `ECTSPoints` of the `Classes` the `Student` is attending, whose link is represented by the `Participation` relationship, as shown by the WebML-OQL expression in curly brackets. WebML-OQL is elaborated in Section 2.7.

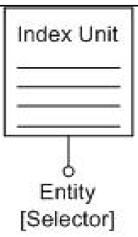
The hypertext model describes the composition and navigation of pages within a hypertext application. The core elements of the hypertext model are: `units`, which are atomic elements of information that can be published; `pages`, which are containers of `units`; and `links`, which are navigational paths connecting `pages` and `units`. `Pages` with similar purpose may be grouped into `areas`, while a wider set of coherent `pages` and `areas` serving to a well-defined set of requirements such as a the needs of a specific group of users are grouped into a `site view`. `Units` display data drawn from a data source described in an `entity` in the Data Model, and the computation of data source can be constrained using `selectors`.

Several types of `units` are supported in WebML: `data units`, which display information of a single object; `multi-data units`, which display information of a set of objects; `index units`, which lists the descriptive properties of a set of objects; `scroller units`, which enable browsing through an ordered collection of objects and give direct access to the previous, next, first and last object; and `entry units`, which enable input of a set of parameter to the hypertext application. Two varieties of `index units` are also specified in WebML: `hierarchical index units`, which allow nesting of indexes; and `multi-choice index units`, which enable selection of multiple index entries. Each `unit` that displays content may use `Selectors` in order to filter the object(s) displayed. Table 1 lists the WebML elements with their graphical and textual notations.

Element Diagram	Sample Textual Notation	Description

	`DataUnit Student`
	`(source Student;`
	`selector LastName=”Pratama”;`
	`attributes FirstName, LastName,`
	`StudentID)`
		Displays information of a single object of an entity. May use `Selectors` to filter the displayed object.

	`MultidataUnit MultiStudent`
	`(source Student;`
	`selector Age=”27”;`
	`attributes FirstName, LastName,`
	`StudentID;`
	`orderBy StudentID)`
		Displays information of multiple objects of an entity. May be ordered by `Attributes` and use `Selectors`.

	`IndexUnit StudentList`
	`(source Student;`
	`selector Age=”27”;`
	`attributes LastName;`
	`orderBy LastName)`
		Lists objects of an entity according to the selected `Attributes`. May use `Selectors` and `orderBy` clause.

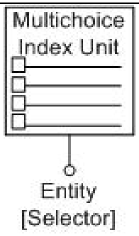
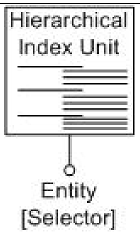
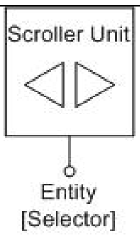
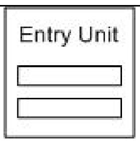
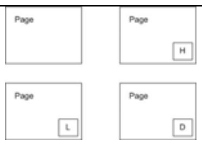
	`IndexUnit StudentList multi-choice`
	`(source Student;`
	`selector Age=”27”;`
	`attributes LastName;`
	`orderBy LastName)`
		Enables selection of multiple objects of an entity through checkboxes. May use `Selectors` and `orderBy` clause.

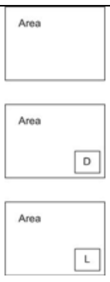
Table 1. Elements in the WebML Hypertext Model

	`IndexUnit StudentList hierarchical`
	`(source Student;`
	`selector Class="06";`
	`attributes LastName; orderBy LastName`
	`  NEST Class`
	`  selector StudentToClass;`
	`  attributes Name;`
	`  orderBy Name)`
		Enables indefinite nested indexing of linked entities. May use `Selectors` and each an `orderBy` clause.

	`ScrollerUnit StudentScroll`
	`(source Student;`
	`selector Age="27";`
	`blockFactor 1;`
	`orderBy LastName)`
		Enables scrolling through object list of an entity. `BlockFactor` determines the number of units scrolled by one click.

	`EntryUnit StudentInput`
	`(fields`
	`FirstNameField String;`
	`LastNameField String;`
	`StudentIDField Integer)`
		Enables user input. `Fields` can be specified according to WebML primitives.

	`Page StudentDetail default`
	`(units StudentList, StudentDetail)`
		Container of units. May have `home`, `landmark`, or `default` keyword.

	`Area StudentAdministration landmark`
	`(pages StudentDetail, InputStudent)`
		Container of pages. May have `landmark` or `default` keyword.

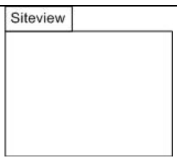
	`Siteview Student`
	`(areas StudentAdministration, Scheduling;`
	`pages StudentHome)`
		Container of areas and pages.

Table 1 (continued). Elements in the WebML Hypertext Model

The navigation of the hypertext model involves `links`, `link parameters` and `link selectors`. `Links` are navigational paths that connect `units` and `pages`. `Link parameters` are information passed along in links. `Link selectors` are unit `selectors` with reference to a `link parameter`.

Links are grouped into `normal`, `automatic` and `transport` links. `Normal links` are initiated by user interaction; `automatic links` do not require user interaction and automatically directs navigation to the destination page or unit; `transport links` do not direct navigation but merely passes link parameters. Links may connect unit to unit, unit to page, page to unit or page to page. Links between pages which do not necessarily contain parameters are called `non-contextual links`, while links between units which pass on parameters required for computing are called `contextual links`. Table 2 lists the types of link in WebML along with their graphical and textual notations.

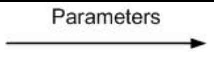
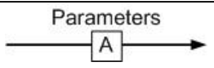
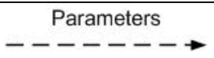
Link Diagram	Sample Textual Notation	Description
	`Link StudentListToStudent (from StudentList to Student; parameters StudentID:{OID})`	Normal link which is activated by user navigation. May pass parameters.
	`Link StudentListToStudent automatic (from StudentList to Student; parameters StudentID:{OID})`	Link which automatically directs navigation. May pass parameters.
	`Link StudentListToStudent transport (from StudentList to Student; parameters StudentID:{OID})`	Link which passes parameters but does not direct navigation.

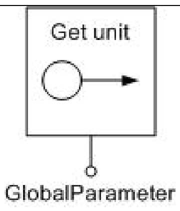
Table 2. Links in the WebML Hypertext Model

The notion of user sessions is supported by means of `global parameters`, which are persistent parameters stored during a user's session in the application, valid within the scope of a site view. Access to the global parameters is enabled by a `get` unit, which retrieves the value stored, and a `set` unit, which stores the selected value into the global parameter. Table 3 lists the WebML global parameter unit and its access units along with their graphical and textual notations.

An example of a WebML Hypertext model is shown in Fig. 8. The model consists of one site view, named `Student`. The `Student` site view consists only of one page named `StudentHome` which has the `home` keyword, meaning that users navigating to this site view are directed to the `StudentHome` home page. The `StudentHome` page contains two units, the StudentList index unit and the Student data unit. Both units have the source entity Student (not shown). The StudentList index unit displays all Students whose age is 27, as described by the selector expression. If a user clicks on one of the Student entries in the index list, the link StudentListToStudent is followed. This normal link propagates along a parameter named SelectedID whose value is taken from the selected entry's StudentID. The Student data unit has a selector which takes the propagated parameter and matches it with the StudentID in the database. This hypertext model enables users to view a list of Students whose age are 27, and select a particular Student in the list to view his or her detailed data.

Unit Diagram	Sample Textual Notation	Description

(none)	`globalParameter CurrentUser`
	`(type OID;`
	`entity User)`
	
	`globalParameter CurrentUserName`
	`(type string;`
	`InitialValue „Pratama")`
		Global parameter declaration consists of user-defined name, type, and default initial value.

	`getUnit getUser`
	`(parameter CurrentUser)`
		Retrieves value stored in the indicated global parameter.

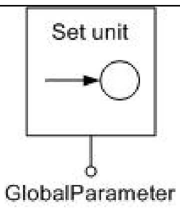
	`setUnit setUser`
	`(parameter CurrentUser)`
		Stores value to the indicated global parameter.

Table 3. WebML global parameter unit and its access units

The content management model is an extension to the hypertext model which enables handling of data supplied by user input to the hypertext application. The model is based on operations, which are processes involving manipulation of data initiated through a navigational link, and outgoing links from the result of an operation which can either be successful (OK) or failed (KO).

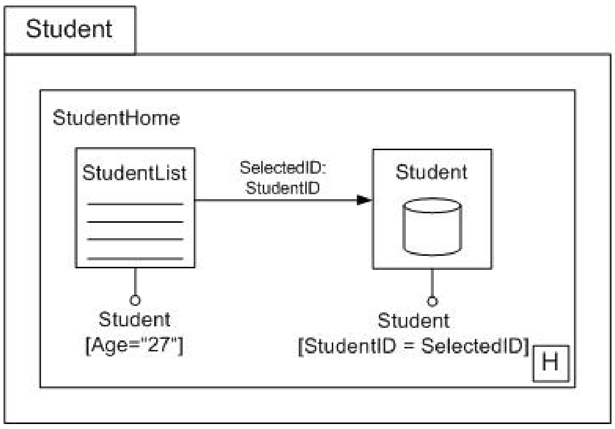


Fig. 8. WebML Hypertext Model example

The operations are modeled by `operation units`, describing either a database manipulation process or a generic process involving external services, and the outgoing links are modeled by `OK-` and `KO-links` originating from operation units. The `OK-link` is followed if the operation performed by the operation unit is successful; else the `KO-link` is followed. The introduction of operation units implies categorizing the units described in the hypertext models as `content units`, meaning units that display content, in order to distinguish between them. Table 4 lists the operation units and outgoing links of the WebML Content Management Model.

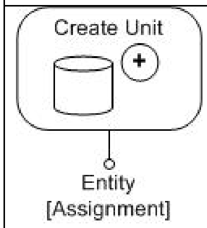
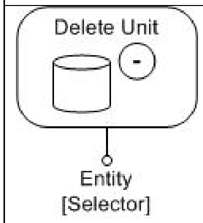
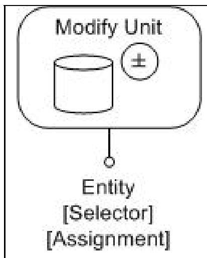
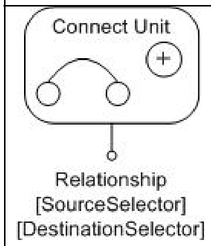
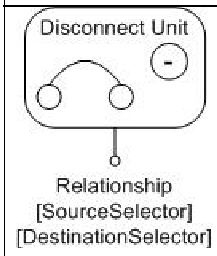
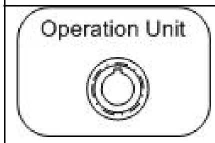
Unit Diagram	Sample Textual Notation	Description
	`CreateUnit CreateStudent` `(source Student;` `FirstName:="Rizki", LastName:="Pratama")`	Creates an instance of the sourced entity. `Assignment` to the new instance's attributes may be directly defined or taken from incoming link parameters.
	`DeleteUnit DeleteStudent` `(source Student;` `selector LastName="Pratama")`	Deletes an instance of the sourced entity. `Selectors` may be directly defined or taken from incoming link parameters.

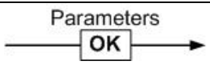
Table 4. WebML operation units and outgoing links

	`ModifyUnit ModifyStudent`
	`(source Student;`
	`selector LastName="Pratama";`
	`Age:="26")`
		Modifies an instance of the sourced entity. `Selectors` and `Assignments` may be directly defined or taken from incoming link parameters.

 Connect Unit
+
Relationship
[SourceSelector]
[DestinationSelector]
	`ConnectUnit AssignClass`
	`(source StudentToClass;`
	`[Student.OID="1"];`
	`[Class.OID="5"])`
		Creates a `Relationship` instance based on the source. `Selectors` may be directly defined or taken from incoming link parameters.

 Disconnect Unit
-
Relationship
[SourceSelector]
[DestinationSelector]
	`DisconnectUnit CancelClass`
	`(source StudentToClass;`
	`[Student.OID="1"];`
	`[Class.OID="5"])`
		Deletes an instance of a `Relationship`. `Selectors` may be directly defined or taken from incoming link parameters.

	`External ChargeCreditCard`
	`(parameters Amount:="100.00",`
	`Number="4990")`
		Invokes a generic operation, which may be a service hosted by another system.

	`OKLink CreateStudentOKLink`
	`(from CreateStudent to ConnectToClass;`
	`parameters StdID:StudentID)`
		Followed upon successful execution of the originating operation.

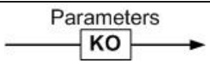
	`KOLink CreateStudentKOLink`
	`(from CreateStudent to ErrorPage)`
		Followed upon unsuccessful execution of the originating operation.

Table 4 (continued). WebML operation units and outgoing links

Predefined operations for commonly occurring processes in hypertext applications exist in WebML; these include `data manipulation`, `access control` and `sending` `email` operations. Data manipulation operations consist of: `create` operation, which initiates the creation of an entity's instance using parameters supplied; the `delete` operation, which removes the instance of an entity; the `modify` operation, which modifies the parameters of an entity's instance; the connect operation, which creates an instance of a relationship; the `disconnect` operation, which deletes an instance of a relationship. Access control operations include the `login` operation, which directs access to a specific site view in a multiple site-view application according to business requirements based on user credentials supplied, and the `logout` operation, which returns the user to a non-protected site view. More on the implementation of access control in WebML is elaborated in Chapter 5. The `send mail` operation enables sending an SMTP-based electronic message. Table 5 lists the WebML access control and send mail units.

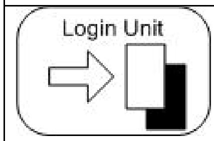
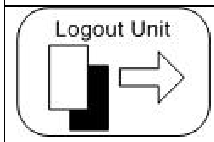
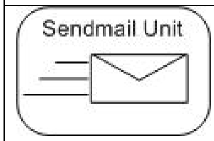
Unit Diagram	Sample Textual Notation	Description
	`login LoginOperation (parameters UserName:=UName, Password:=Pwd)`	Matches the supplied credentials to the `User/Module` tables.
	`logout LogoutOperation`	Directs user to a public site view from a protected site view..
	`sendmail SendMailOperation (parameters Sender:=Sender, Recipients:=Recipients, Subject:=Title, Body:= Text, Attachments:=Attachments)`	Sends an SMTP-based message using the supplied parameters.

Table 5. WebML access control and send mail units

Several WebML operation units representing a sequence of coherent processes may be grouped into transactions. By grouping into transactions the atomicity of the whole sequence of processes is ensured, meaning that a failure in any of the component operation units does not affect the state of the database.

A hypertext model incorporating content management units is depicted in Fig. 9. This example extends the hypertext model described in Fig. 8. The `Student` site view now contains an additional page, the `ErrorPage` page. The `StudentHome` page also contains a third unit, the `CreateStudent` entry unit. By filling the form and hitting the `submit` button on the entry unit, users initiate the `CreateStudentTransaction` transaction. This atomic transaction begins with a `CreateStudent` create unit, which takes the parameters `FName` and `LName` from the entry fields, creates a new `Student` instance and stores it in the database. Upon successful instance creation the OK-link is followed, which leads to the `ConnectToClass` connect unit, the second operation in the transaction. The `ConnectToClass` connects the newly created `Student` instance to a `Class` instance also given as a parameter carried from the entry unit. If the execution of any of the two operations fails, the respective KO-link is followed, thus directing the user to the `ErrorPage`. Upon successful execution of the connect operation the user is directed back to the `StudentHome` page.

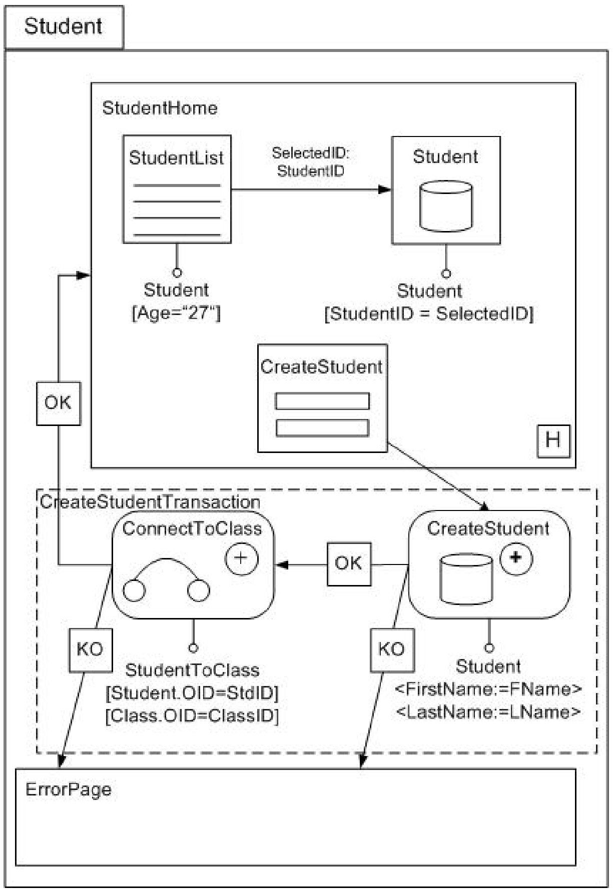


Fig. 9. WebML Hypertext Model with content management example

The data model and hypertext model enable a working, high-level specification of a read-only hypertext application, such as a plain website consisting of a navigable set of pages. For hypertext applications involving user interaction, the content management model is required to handle input, processing of parameters, manipulation of the database, and interaction with external (Web) services.

2.7 WebML-OQL

WebML-Objective Query Language (WebML-OQL) [WR-WMLg] is a query language derived from OCL used in WebML. The language is not defined in the core version of WebML [CFBBCM03] but it is first introduced by WebRatio [WR-WMLg]. WebML-OQL specifies derivation query for entities, relationships, and attributes. The complete syntax of WebML-OQL can be found in Appendix A.

This language is used in several occasions in WebML models: in the data model, WebML-OQL is used to define the derivation rules of an entity's derived attribute; in the hypertext model, WebML-OQL is the language used to write selector constraints and to express validation rule constraints in entry unit fields. An example of WebML-OQL expressions for each type of derivation query is shown in Fig.10.

```
/* EntityQuery: deriving a Student with the last name containing the
string "Prata" */
      Student AS s WHERE s.LastName contains "Prata"

/* AttributeQuery: deriving a Student's total ECTSPoints from all
Classes participated (see Fig. 7.) */
      Sum (Student.StudentToClass.ECTSPoints)

/* RelationshipQuery: deriving classmates of a Student with the same
age */
      Self.StudentToClass.ClassToStudent AS mates WHERE Self.Age =
      mates.Age
/* RelationshipQuery: deriving a User's favourite articles */
      Self TO Article AS A WHERE A.categoryName in
      Self.UserToPreference.Name
```

Fig. 10. Example WebML-OQL expressions.

The `EntityQuery` enables selecting particular instances of an entity using the specified conditions, in this example to select `Students` with the last name containing the word “Prata”. This type of query is used in the hypertext model. The `AttributeQuery` is used to derive attributes of an entity in the data model, as shown in the previous example in Section 2.6 Fig. 7. The `RelationshipQuery` derives relationships in two ways: the first example shows concatenation of relationships, shown as `StudentToClass` and `ClassToStudent`, which enables selecting all `Students` having the same age; and the second example determines pairs of related object by evaluating a condition, shown as the entities `User` and `Article` with the condition that the category of the `Article` is contained in the `User`’s preferred categories in the `Preference` entity, linked by the relationship `UserToPreference`.

2.8 WebRatio

WebRatio is a CASE tool which is based on and extends WebML. The tool enables automatic generation of executable code and synchronization with the database from a set of input consisting of WebML data model, hypertext model and an additionally introduced presentation model.

The data model used in WebRatio includes all the entity relationship concept of WebML data model. The tool enables mapping and synchronization of the data model to a database system, which means changes to the data model are reflected in changes of the database tables and records. The databases supported by the current version are among others the proprietary Oracle [Orcl], MS SQL [MSSQL], MS Access [MSA] and DB2 [DB2] as well as the open source PostGreSQL [PGSQL] and MySQL [MySQL].

The hypertext model in WebRatio uses the concepts specified in WebML and introduces extending units and pages. Extensions to content units include: `sortable index units`, which allow index entries to be sorted according to one of their properties; `event calendar units`, which allow display and browsing of entries in a calendar-based browser; and field types for the entry unit, which include: `long-text field`, `date field`, `Boolean field`, and `selection field`.

Extensions to operation units include: `bulk create units`, which allow creating multiple instances in a single step; `bulk modify units`, which allow simultaneous modification of multiple instances of an entity; `change group units`, which enable users to change groups or access levels on the fly, allowing them to switch to different site views without needing to log out of the current site view; an improvement over `send mail unit` which enable among other sending messages with multi-part attachments and including link parameters in the messages; `selector units`, which enable selection of an instance of an entity according to selector constraints; `math units`, returning mathematical values based on (link parameter) operands and operator specified; and `time units`, returning system timestamps as link parameters.

WebRatio also introduces the concept of navigation units to WebML, which direct the path of navigation in case of conditional paths. These include the "`is not null`" units, which return `KO` if the supplied link parameter has a value of null and otherwise `OK`; and `switch units`, which apply the classic switch-case statement logic based on link parameter input.

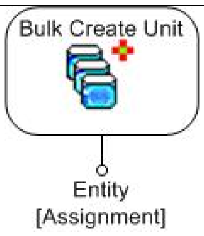
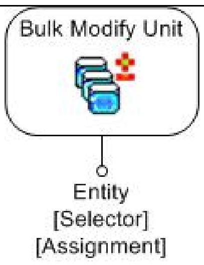
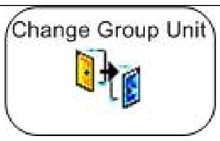
Unit Diagram	Description
	Creates multiple instances of an entity according to the specified `Assignment`.
	Modifies multiple instances of an entity according to the specified `Assignment` and `Selector`.
	Enables switching user `Groups` on the fly, allowing `Users` with multiple `Group` identities (e.g. an administrator) to switch to different site views without needing to log out.

Table 6. Additional units introduced in WebRatio

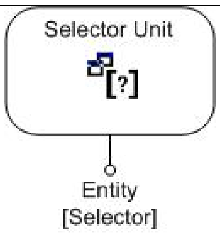
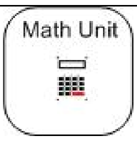
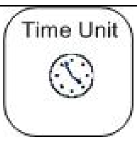
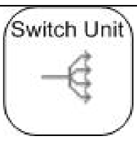
Power Mail Unit	An improvement over the `SendMail` unit which enables inclusion of link parameters as part of subject and body entries, file attachments, and multi-part messaging.
	Enables selection of instances of an entity using the specified `Selector` without publishing the content. May be used inside pages or outside for directing navigation.
	Parses mathematical expressions, either statically or dynamically using input parameters. Propagates the result as output parameter. May be used inside pages as content unit or outside pages as operation units.
	Specifies current system time, date, or timestamp. Can be used as content or operation units, therefore both inside and outside pages.
Is Not Null Unit	Checks the value of its input parameter. If the value is null, the KO link is followed, else the OK link.
	Checks the value of its input parameter to a set of given case values. The unit has multiple OK links, each corresponding to a case. A matching value of an input parameter to a specified case value entails following this case's OK link.

Table 6 (continued). Additional units introduced in WebRatio.

WebRatio also provides built-in validation rules used to validate single fields in the entry unit. The built-in validation rule can match input strings to patterns defined in WebML data types or perform Boolean operation with an input parameter in a single entry field. The predicates already predefined in WebML are: `Equal`, `NotEqual`, `LessThan`, `LessOrEqual`, `GreaterThan`, `GreaterOrEqual`, `MatchForRegularExpression`, `BeginsWith`, `EndsWith`, `Contains`, `NotContains`, `In`, `NotIn`, `IsNull`, `IsNotNull`, `EqualToField`, `NotEqualToField`, `MinLength`, `MaxLength`, `CreditCard`, `@-Mail`, `Time`, `TimeStamp`, `Date`, `Boolean`, `Byte`, `Short`, `Integer`, `Long`, `Float`, `Number`, `IntRange`, `FloatRange`.

In order to validate an entry field with a predicate not natively defined in WebML and WebRatio, and also to validate a set of input spanning multiple entry fields in a single entry unit, WebRatio allows extension to the validation rule using custom validation rules [WR-AFT06]. Specifying a custom validation rule entails implementing a `validate()` method for the validation rule object with the desired input-checking procedure. The custom validation rule created can be assigned to individual entry fields to validate them or to an entry unit to validate multiple entry fields. A complete description and example of building custom validation rules are described in [WR-AFT06].

WebRatio acknowledges the need of developers to define custom units to include business cases which are not covered by the WebML units already given in [WR-CUG06], since WebML units are designed to cover the most commonly occurring business cases of designing a data-intensive Web application. These custom units are created as plug-in units to WebRatio, can be modeled directly in the hypertext models, interact with other units through links and generate executable codes as well.

Specifying a custom unit in WebRatio require the following:

- Adding a unit definition to the unit library
- Adding a set of XSLT rules for validating the usage of the custom unit in the hypertext diagram, and for producing error and warning reports
- Adding a set of XSLT rules for documenting the usage of the custom unit in the `WebMLDoc` project documentation
- Adding a set of XSLT rules for producing the runtime XML descriptors associated with the custom unit
- If the custom unit is a content unit, adding one or more XSLT rules for producing the server-side tags or scripting to be inserted in the page templates
- Implementing a runtime class, which actually performs the business service for which the unit is designed

In designing a web application using WebRatio, a developer follows the following simplified steps:

1. Gather the system and business requirements, and create a use case list.

2. Based on the use case list’s actors and objects, create Entities and connect them with Relationships in the Data View. Synchronize the model to a database system and populate the database tables with trial or setup values.
3. Based on the use case actors, determine the division of site views according to the actors’ roles; a site view should accommodate the use cases of an actor.
4. Based on the use cases, create pages in the Hypertext View and populate them with units and links to facilitate each use case.
5. Generate executable code by compiling the models into classes and then building the codes, done automatically with a build command.

An example of model-driven development using WebRatio is described in Chapter 3.

Formatted: Bullets and Numbering

3 Model-driven development of a collaborative web application: case study

WebML and the CASE tool implementing it, WebRatio, offer the possibility of model-driven development of web applications that directly result in executable code. While a wealth of UML-based tools [Gol05, EMF, TOG] also aim at that goal, none implements the full set of abstractions encoded in WebML [WR-WMLg, WebSI]. WebRatio 4.3 is the latest version of an advanced CASE tool based on WebML for developing web applications. WebML, as a domain-specific language geared towards designing data-intensive web applications, has the advantage of simpler models compared to UML, resulting in fewer diagrams and faster design time. A comparison of modelling the same web page in UML 2.0 and WebML is discussed in [MFV06].

The term collaborative web application discussed in [MMCF03] is defined as an application serving the business requirements of an organization or group of persons comprising of several different roles, all working towards the same goal. The example of collaborative application discussed in [MMCF03] is a conference management system, which enables participants and organizers to work together in creating a scientific conference.

This project broadens the definition of collaborative web application, to encompass also community portals or better known as social networking applications, to which an alumni networking platform such as an alumni community's platform belongs. These social networking applications also comprise different roles (mostly users and administrators) and work toward similar goals, not necessarily the same one shared by all users, namely information and data sharing across the community.

The development of an alumni networking platform or web application is chosen as a case study for WebML, because as a social networking application the alumni networking application has a rich collection of functionalities to practice the development of a data-intensive web application.

Formatted: Bullets and Numbering

3.1 Business and system requirements

Reproduced from the alumni network website project documentation:

The alumni network currently uses multiple channels to communicate to its members, including: newsletter, *Yahoo!Groups*, alumni homepage and individual emails. These channels are independent of each other, therefore monitoring of activities is difficult if not impossible. Furthermore, the degree of participation in each channel differs, hindering effective communication. Additionally, the current channels do not address requirements from the alumni, for instance a marketplace forum which can facilitate the transfer and exchange of ideas, goods, and services among members, and also employment bourses.

The aim of this project is to consolidate the communication channels of the alumni network by creating a platform which serves the existing requirements as well as requirements that have not been served by existing channels. With a consolidated channel, monitoring of activities is made possible. Furthermore, internal marketing activities can be focused on the channel, thus enabling high participation rate and ultimately achieving effective communication to alumni network members.

3.2 Use case list

The alumni networking application has several actors and use cases derived from the business requirements. The actors in the use case list are: students, who are users exchanging data and information; alumni board members, who are themselves students but in addition possess administrative rights over the students and the alumni network's formal activities; institution administrators, who are cooperating with the alumni network in exchange of alumni data and facilitating alumni network events; and visitors who may view publicly available information about the alumni network. A complete use case list can be found in Appendix B.

Formatted: Bullets and Numbering

3.3 Designing with WebRatio

Designing a web application using WebRatio is straightforward when referring to the simplified designing steps discussed in Section 2.8.

3.3.1 Data model

The data model consists of the actors and objects specified in the use case list, along with their respective attributes and the relationships linking them. All actors except the visitor belong to a super entity named `User`, and each is specialized in the sub entities `Student` and `InstitutionalAdministrator`. The alumni board member, who is also a `Student`, is identified by an attribute in the `Student` entity. This is due to the constraint that a `Student` does not hold an alumni board member post indefinitely, therefore the role is not permanent and modeling the alumni board member as a separate entity does not fit this requirement. The visitor is not represented in the data model because it does not possess any attributes.

The model for roles concerned with access control, in the data model represented by the `AccessLevel` entity, defines the roles corresponding to the actors in the use case list with the exception of the visitor. The roles defined are `Students`, `AlumniBoardMembers`, and `InstitutionalAdministrators`. Each of these roles is associated to a protected module in the `Module` entity containing the pages fulfilling the set of requirements of each role. The complete data model for the case study is given in Appendix C.

3.3.2 Web model

The web model consists of four site views for each of the roles (or actors) specified in the use case list. Each site view contains pages and areas which serve the use cases listed for each role. The `Public` site view, intended for the role of visitor, is the home site view and includes pages for login, registration, and publication of general information. By performing a login, the user supplying the right credentials is directed to one of the three remaining protected site views.

The first protected site view is `Student`, which contains pages serving use cases related to the role of `Students` such as viewing and updating profiles, creating groups and events, initiating polls and discussions, sending messages, and uploading files. A `Student` may only update her own profile, but may view the profile of other `Students`. The amount of data in the profile of other students one can view depends on the preference set by the `Student` owning the profile. `Students` may also join existing groups by submitting a membership request or create a new group based on a category she determines. A `Student` creating a group is then granted the group moderator privileges, which means she can change the group description, upload group pictures, accept or decline membership requests from other users, remove postings and kick a member out of a group. A `Student` can be granted or removed from group moderator privileges by other moderators in a group. Within groups, `Students` can post topics, upload files, create polls and events. `Students` may also send messages to other `Students` using a two-step approach: the first step is assigning addressees by populating a list of addressees, and the second step is to determine the subject and body content of the message. Files are uploaded into `Albums`, which `Students` create within a `Group` or in his own profile as a private folder. Access to `Group Albums` and private `Albums` are limited to group members and personal settings, accordingly. Finally, `Students` may update the account data such as password, email and preferences.

The second protected site view is `BoardMember`, intended for alumni board members who hold administrative rights, which serves use cases such as user management, data management, event management, board management, announcements, newsletters and approving registration requests. A `BoardMember` can approve a registration request, automatically creating a `Student` instance and sending the credentials to the requesting user. `BoardMembers` can also edit and delete existing user data. Event management allows a `BoardMember` to monitor participation and fee payment of a particular type of `Event` and generate status report on the participation status of an `Event`. In data management, a `BoardMember` can manage the instances of various category entities, and manage the configuration of default class groups. A `BoardMember` is able to create `Announcements` visible to all `Students` and visitors to the public site. A `BoardMember` responsible for newsletter publications can upload `Newsletters` for public view. `BoardMembers` ultimately appoint and discharge among `Students`, practically assigning and de-assigning `BoardMember` roles to a `Student`, using the board management functions.

The third protected site view named `InstitutionalAdministrator` serves the use cases related to the role of institutional administrators such as announcements and reporting of student activities. Like `BoardMembers`, `InstitutionalAdministrators` share the use case of creating `Announcements` intended for `Students` and the public alike. The `InstitutionalAdministrator` may view a report summarizing the `Student` activities within the alumni network. `InstitutionalAdministrators` may also view selected `Student` profiles and send messages to `Students` according to the two-step approach.

Since the alumni board member is also a `Student` it is also required to enable him to access both site views. Therefore the `ChangeGroup` unit is provided in each protected site view, and the `Student` instances holding alumni board member posts are given the necessary access rights. The complete web model for the case study can be viewed in Appendices D-1 until D-4.

Due to the current implementation of WebML in WebRatio, a workaround is required to fulfill the requirements of sub-roles within user-defined groups. According to the use case list, `Students` may create their own groups, and each group's activities, discussion as well as group data is shared only by its members. `Students` then take additional sub-roles as group moderator or group member for a `Group`, with other `Students` not registered in that `Group` taking the role of guest. In order to reflect the different requirements of each sub-role, within the same `Student` site view the pages for each sub-role are defined. Navigation is then directed by querying attributes characterizing each sub-role using `Selector` units and passing the parameter to a conditional unit, the `IsNotNull` unit. By cascading combinations of `Selector` and `IsNotNull` units it is possible to direct navigation to one of the three sub-role pages of a `Group` (`moderator`, `member`, or `guest`). The workaround is shown in a simplified hypertext model reproduced in Fig. 24.

In the example shown in Fig. 24, a `Student` clicks on an entry in a list of available `Groups` (not depicted) in the `Group Home` page. The link navigates to the `LookupTransaction` transaction and carries the `GroupOID` as a parameter. The `LookupTransaction` functions as follows: first the `GroupOID` is entered as input parameter to a `Selector` unit named `MemberLookup` and along with the parameter `CurrentUserOID` supplied from the `GetUserOIDGlobalParameter` unit is computed using the following selector condition: `Group.GroupOID WHERE Group.GroupOID = X AND Y in Group.GroupMembers`.

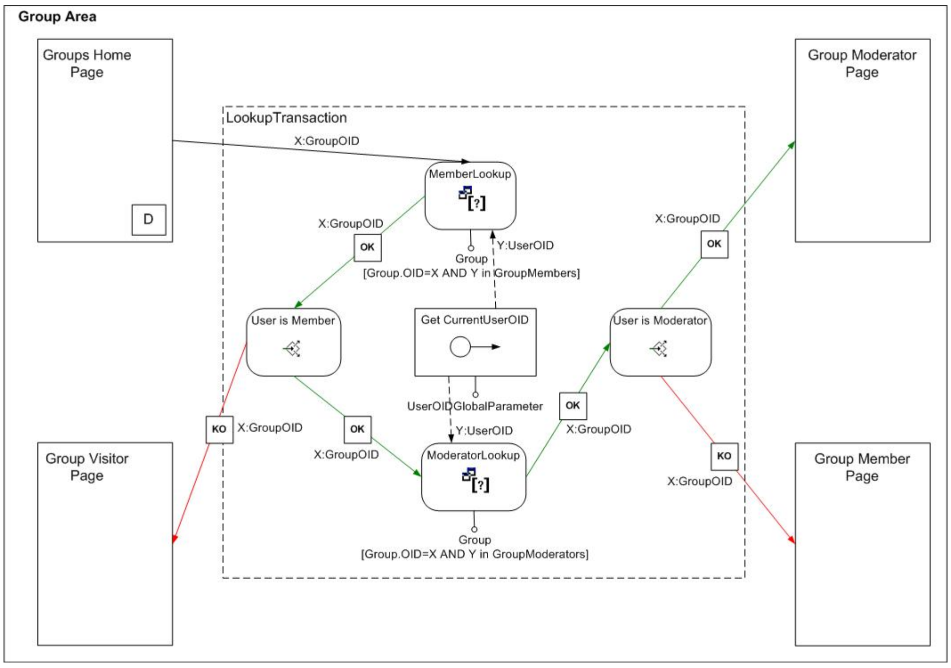


Fig. 24. Workaround for group membership requirements

The `GroupMembers` is the name of a relationship between the entities `Group` and `Student` in the data model representing `Students` who are members of a `Group` instance. The selector condition outputs a `GroupOID` stored in a parameter according to the given parameters to an `IsNotNull` unit named `User_Is_Member`. This unit checks whether the current user, represented with the OID parameter `Y`, is a member of the current `Group`, represented by the OID parameter `X`. If the current user is not a member of the selected `Group`, the `MemberLookup` selector unit would have produced null, therefore the KO-link is followed and the user is directed to the `GroupVisitorPage`. The `GroupOID` is carried along so that units in the `GroupVisitorPage` can load the appropriate `Group` instances. If the current user is a member, then the OK-link is followed and directs navigation to a second `Selector` unit named `ModeratorLookup`. This unit does a similar function to the `MemberLookup`, the only difference is that the relationship in check is the `GroupModerators`. The computed `GroupOID` is then directed to a second `IsNotNull` unit named `User_Is_Moderator`, which checks if the current user is a moderator for the specified `Group`. If the user is not a moderator of the `Group`, then the user is directed to the `GroupMemberPage` carrying the `GroupOID` parameter. If the user is a moderator of the `Group`, then the user gains access to the

GroupModeratorPage. This LookupTransaction navigation-directing transaction applies the pseudo-code shown in Fig. 25.

```
Select Group
Lookup User in GroupMembers
If User is GroupMember then
      Lookup User in GroupModerators
      If User is GroupModerator then
            Goto GroupModeratorPage
      Else Goto GroupMemberPage
      EndIf
Else Goto GroupVisitorPage
EndIf
```

Fig. 25. Pseudo-code for the LookupTransaction transaction

A crucial second type of requirements concerning sub-roles cannot be addressed using this workaround. The requirements specify limiting access to each block of personal, professional and academic data of a Student in the ViewProfile page. The access level for each block is set independently for any of the required sub-role (only the owner, only the owner and other Students, and the general public). Since these sub-roles are characterized by one or more attributes in the Student or User entity, a similar workaround mechanism needs to be designed. However, since each block is independently set, the implementation using one page for each sub-role implies creating pages for all possible combinations of blocks and access levels, in our case it would amount to 24. Furthermore, the combination of navigation-directing units would be too complex to model. In the end the workaround has to resort to manual coding.

3.3.3 Presentation model

The presentation model specifies a common style sheet for all of the pages. The layout of each page is then specified according to the requirements and aesthetics. The presentation model places units of a page in grids; the developer can choose where to display which units or not to display a unit at all. For the case study a layout template provided by WebRatio is used. Layout in the Student site view is differentiated from the other site views, because in the Student site view a side panel is required to hold links to pages for the current user at all times, even while browsing in other Students' profiles.

4 Areas of improvement in WebML

In Chapter 3 it is shown that an alumni networking platform can be considered a data-intensive web application. As a domain-specific language (DSL), WebML is designed to handle the design of data-intensive Web applications. In effect, its semantics, along with the concepts of pages, areas and site views as well as set of data and operation units are able to represent most of the logic in designing Web applications. Some shortcomings of WebML with respect to other model-driven technology are discussed, such as limited expressiveness about access control policies. These shortcomings are especially relevant to the development of collaborative, social-networking Web applications. These suggestions could be integrated in a future version of the language.

The notion of access control for information security has been in discussion for more than two decades. The current access control method widely used in business applications is the role-based access control (RBAC) [FK92]. RBAC is designed as a solution to meet industry and commercial application security requirements which were otherwise not fulfilled by Mandatory Access Control (MAC) and Discretionary Access Control (DAC). WebML adopts role-based access control in its inherent concept of site views. Site views are defined as a collection of pages serving the requirements of a specific group of users, which in turn means that each site view is geared to a specific role. For example, the alumni network application has different site views for students (users), alumni board member (administrator), and the public. WebML also supports the process of authorization of access credentials by means of the `Login` operation, in which username and password are supplied to perform login to the website [CFBBCM03].

Fig.11 depicts how access control is implemented in WebML. Users are grouped into specific roles, and then using the set of requirements for each role a site view is designed for each role. If a site view is protected it can only be accessed by authorized users, for instance only users having the access rights for `Role1` can access the protected site view `Siteview1`. Assignment of multiple roles to a user is also possible.

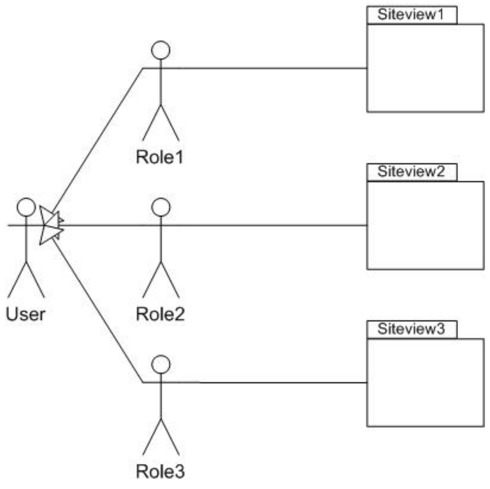


Fig. 11. Role-based access control in WebML

Further detailing into access control is done in WebRatio, the CASE tool implementing WebML. WebRatio details the concept of WebML access control by introducing the notion of modules. Modules are essentially site views, but an area or a page may also be defined as a module according to role definitions. The modules are then designated as private or public. The modules which are designated as public may be accessed by any visitor to the application's website, while in order to access the protected ones, access credentials need to be supplied [WR-ART06].

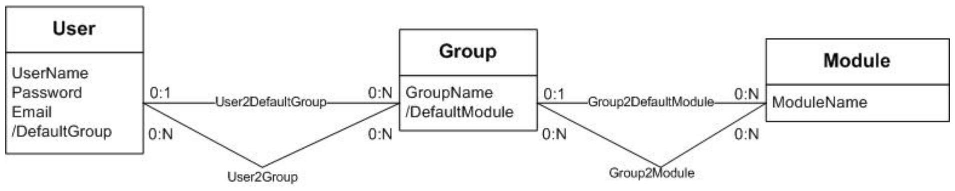


Fig. 12. Implementation of WebML access control in WebRatio

The concept of role-based access control in WebML is implemented in the `User-Group-Module` data model depicted in Fig. 12. The data model shows that each `User` belongs to one (or more) `Group`, which is an equivalent of role. The `User2DefaultGroup` relationship, represented by a derived attribute in the `User` entity, indicates to which `Group` a `User` mainly belongs. This is often the only user-to-group relationship that is defined for most users of an application. `Users` with more privileges such as administrators may have additional groups assigned to them; this is made possible by the `User2Group` relationship. Each `Group` entity contains a `DefaultModule` derived attribute, which represents the `Group2DefaultModule` relationship and defines which default module a `Group` is associated with. Again, for a `Group` having reasons to access other modules than its default, the `Group2Module` relationship is available for that purpose. The `Login` operation serves as the authorization process for access credentials supplied by the user; this operation directs the user to the default protected module corresponding to her `Group` or role. The `Logout` operation returns the user from a role-specific module to the public, non-role-specific module. The `ChangeGroup` operation enables users with multiple `Group` associations defined in the `User2Group` relationship to change groups on the fly. This means such users can switch to different site views without needing to logout and login. With this implementation, the formal definition of RBAC as described in Chapter 2.4 is fulfilled.

In a nutshell, the implementation of access control in WebRatio can be simplified as follows: a user has access credentials of username and password, each user is associated to a specific role, and by logging in a user accesses the module relevant to her role in the application. This implementation is also applied in more advanced access control features of WebML such as visibility control policies and protected alternative pages [WR-ART06].

The implementation of access control in WebRatio, as well as the advanced features of visibility control policies and protected alternative pages all rely on the static assignment of roles defined in the `User-Group-Module` data model. While this implementation is fine for industrial and commercial applications, in which roles can be defined on business units or functions in the organization, it does not fit perfectly with collaborative, social-networking applications. In social networking applications, users still have clearly defined roles, for example in the alumni network application's definition of students and administrators. However in addition to that users may form groups within themselves, thereby creating sub-roles. For instance, in an alumni network, students may form groups of students belonging to a certain previous study program, a certain nationality, or a certain career interest. Further example can be found in popular social networking applications such as MySpace [MySp], Orkut [Orkt], and XING (formerly OpenBC) [Xing], where it is possible for users to create, manage and subscribe to an arbitrary number of groups. Even data-sharing platforms have group functions, for instance the popular photo-sharing website Flickr [Flkr] and the video-sharing website YouTube [YouT], designed to help their users maintain a degree of access control over the content they posted on the websites.

Each (sub) group has its own set of requirements which is a subset of the student requirements, but contains also common group functions such as group meetings, group postings and data storage which is accessible only by group members. The groups represent sub-roles of users which have similar sets of requirements but still need access control to prevent unauthorized viewing or modification of group data. This "fine-graining" of user roles makes it almost impossible, if feasible at all, to define modules for each sub-role according to the implementation of WebML access control. Furthermore, in a social networking environment users may create new groups, cease subscription to a group, or join another group anytime, as well as post or take off content from group data storage as often as they like. Fig. 13 illustrates possible sub-roles of the role of user, and shows the interconnection between them.

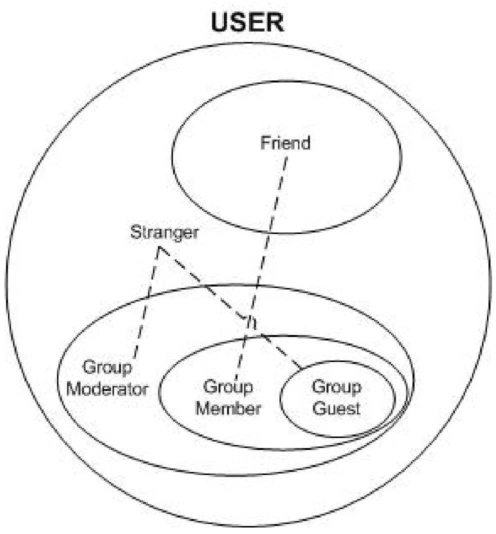


Fig 13. Roles-within-roles

Roles within a group also require access control. Owners or moderators of groups have high access rights within the groups they control: they are able to approve a membership request, cease a user's membership, change the description of groups, and delete a member’s posting, among others. Members, on the other hand, only have “basic” group access rights such as posting text and discussion to the group, as well as viewing the group’s shared data. A non-member has then only very limited access rights in a group she visits, namely for viewing the group’s description and requesting a membership.

In Fig. 13 the role of user is subdivided into group moderator, group member and group guest with regard to a particular user group. Outside the context of user groups, the social networking applications mentioned also implement the notion of “friends” or trusted circle of users relative to the current user, also depicted in Fig. 13. Users within this trusted circle may have more privileges such as sending messages, viewing personal data or content that is otherwise not meant for non-trusted users. A user may designate another user as trusted by essentially flagging her, and remove another user from her list of trusted users by changing the appropriate flag. This is done in practice by “adding” a contact to a user’s friend list or “removing” a contact from a user’s friend list, respectively. Again, this is a dynamic user role situation that needs attention when designing social networking applications.

Each subdivision in a generic role adds a layer to the sub-role-mapping, therefore increasing the complexity of the models needed to represent them. With the current specification in WebML, the sub-roles are not visible in the models and have to be manually implemented using the available units. The case study in Chapter 3 even shows that for data privacy issue manual coding in the executable code has to be done because the current WebRatio implementation does not support such use case. On the other hand, the invisibility of sub-role mapping decreases the accuracy of the models and implies additional editing work upon every change done to the application through the model.

5 Proposed extension to WebML

Chapter 4 discusses the shortcomings or limitations of WebML in the context of model-driven development of collaborative web applications. To overcome the limitation of access control in a dynamic, multi-group environment this project proposes the mapping of roles within roles in WebML. Afterwards an improvement to WebML is presented, in which authorization constraints are added to the WebML semantics as a complement to role-based access control using site views. An implementation of this concept is then described, namely in the form of an additional type of constraint introduced to existing WebML units.

5.1 Mapping of roles within roles

As shown in Chapter 4, there are two general types of user roles in software applications. The first type is the large, static user roles which are normally mirrored to the user's corresponding function or role in the organization. These roles clearly divide the users into groups each having a distinct set of requirements. Access control is therefore strict, and mobility among these groups is limited. An example of such type of roles would be the users and administrators of an application. The set of requirements of users are different than administrators, because users mainly perform productive tasks within the application while administrators manage the application's users and database. These roles do not change easily as they represent different positions in the organization.

The second type of roles is more fine granular and is not closely mirrored to the user's corresponding function. These roles do not divide users into groups of distinct sets of requirements; rather they divide users into groups defined by specific parameters. The parameters may be taken from the user's set of preferences, or state of the application system itself. The role configuration is more unstable and dynamic, reflecting the high user mobility among these roles. Access control is therefore relatively weaker in this type of roles. An example of such type of roles would be group moderators, group members and group visitors of a user-defined group (e.g. *Indonesian Students in TUHH*). These roles are adopted by a limited amount of users, and vary from one user-defined group to another. Mobility within and among the group is high, and users can change roles without necessarily changing the position in the organization (i.e. they are still *Students*). Table 7 gives a comparison between these types of roles as discussed.

Static roles	Dynamic roles
• Rigid, fixed user constellation • Larger group of users • Defined on distinct sets of requirements • Mirrored to user's corresponding function in organization • Low or no mobility among roles • Strict access control	• Dynamic user constellation • Smaller group of users • Defined on parameters • Not mirrored to user's corresponding function in organization • High mobility among roles • Weak access control

Table 7. Comparison of static and dynamic roles in an application

A user may possess both types of static and dynamic roles simultaneously, for instance in the alumni network application a Student (static role) may be a member of the ICS group (dynamic role). The same student may also be a member of many other groups and moderator of the Asian Students group. Once the student joins a new group or leaves one of his groups, his set of dynamic roles changes immediately. Thus the dynamic roles are essentially contained within the static role and their existence is dependent on the static role, effectively "fine-graining" the larger, static role into smaller, more dynamic roles.

On defining the mapping and access control of static roles within commercial and industrial applications, role-based access control [FK92] is the current standard widely employed. For the much newer dynamic roles, becoming popular with the proliferation of social-networking applications, no general standard for mapping has been defined. Access control for the dynamic roles is therefore implemented in the programming, using if-clauses checking the parameters defining each role.

This project proposes a mapping of dynamic roles in social networking applications in a three-dimensional plane, shown in Fig. 20. The "user" axis represents users already in the designated static role, and the "role" axis represents the dynamic roles a user can possess, while the "context" axis represents in which setting (e.g. user-defined group) the user-role definition is relevant. The number of roles represented does not change, but the users and context can vary from changes to the application data. This mapping is contained within the mapping of static roles in the application.

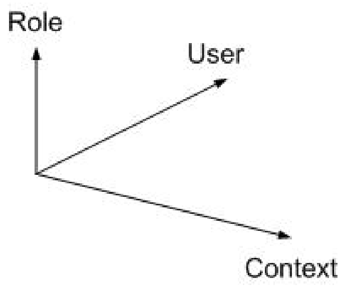


Fig. 20. Mapping of dynamic roles

The most common dynamic roles in social networking applications are the ones concerning "friends" or trusted circle of users, and the ones concerning group membership and moderation. These roles can be mapped in the three-dimensional map depicted in Fig. 20. For the "circle of trusted users" type of dynamic roles, the roles are defined as "friend" and "stranger", using an attribute in the user's class as a parameter, and the context is the current user, i.e. for each user there is (possibly) a different set of friends and strangers. For the type of dynamic roles concerning group membership and moderation, the roles can be defined as "moderator", "member", and "visitor", while the context is each group. If an application supports both types of dynamic roles, a separate mapping for each type must be present in the application's model.

In WebML, the mapping of dynamic roles can be done in the data model using relationships. In the data model, a relationship between two entities connects instances among the two entities and maps them in pairs. The two entities represent two axes, while the relationship represents the third axis. Using this mapping, the application can determine which dynamic role a user possesses in the current context. An example of the data model mapping for the "friends" and group membership types of dynamic roles is shown in Fig. 21.

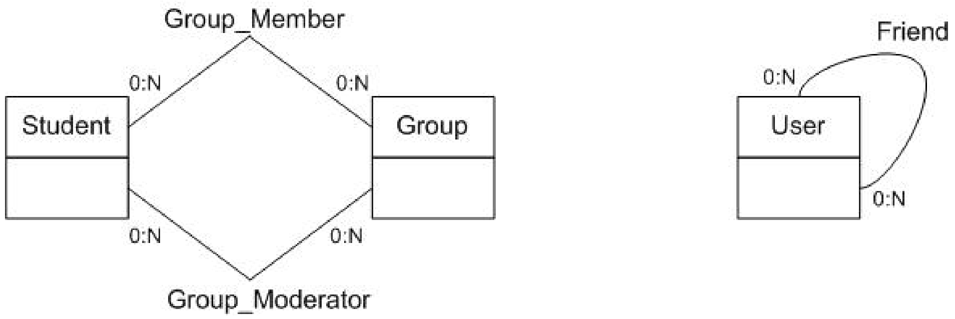


Fig. 21. Mapping of dynamic roles in the WebML data model

For the group membership type of dynamic roles, the entities involved in the example are `Student` and `Group`, each mapped to the `user` and `context` axis of the proposed mapping in Fig. 20, and the relationships between `Student` and `Group` represent the possible roles. In this case the role `axis` only has two positions ("moderator" and "member" – "visitor" does not need to be modelled since all other users are considered visitors) and the roles themselves are fixed, but not the users and the context, i.e. users can change group allegiance and new groups may be created at any time. For the "friends" type, both the `user` and `context` axis are represented by the `User` entity, because the context is the current user and other users are paired with the current user in a relationship to form the third axis. In this case, only friends need to be modelled in a role, all other users are automatically considered strangers.

The mapping of the more complex type of dynamic roles, the variable access level settings on different groups of data as described in the case study in Chapter 3, can also be done in WebML. In the requirements described, the roles for this case are `OwnerOnly`, `OwnerAndOtherStudentsOnly`, and `Everyone`. Only the roles `OwnerOnly` and `OwnerAndOtherStudentsOnly` have to be modelled, as shown in the previous example. Each of these roles is modelled using a relationship originating from and going to the `User` entity, similar to the "friends" type of dynamic roles. In order to model the different access level settings for each block an additional entity is required, i.e. a `SpecialAccessLevel` entity. This solution is not efficient, because the size of the `OwnerAndOtherStudents` table implemented in the database will scale exponentially with the increase of `Student` instances. Moreover, the current standard of WebML and implementation of WebRatio do not support an `If` unit which is required to handle the variable access level settings for each group of data.

5.2 Extending WebML with Authorization Constraints

In [LBD02] access control decisions are differentiated into two types: the first is declarative access control decisions, which depend on static information, namely the assignment of users and permissions to roles; the second is programmatic access control decisions that depend on dynamic information, namely the satisfaction of authorization constraints in the current system state. The declarative access control decisions are designated as role-based access control configurations, to which static roles fall into, while the programmatic access control decisions are able to handle the configuration of dynamic roles. A combination of both access control decisions can be enforced to complete the access control requirements of social networking web applications. Therefore the incorporation of authorization constraints from SecureUML for programmatic access control decisions into the role-based access control already present in WebML is necessary to fulfil these requirements.

According to [BDL05] in order to combine a design modelling language such as WebML with SecureUML, three formal prerequisites are required: a concrete syntax based on UML; an abstract syntax based on MOF; and a semantics with a first-order signature that includes a sort `Users`, a constant symbol `caller`, and a function symbol `UserName` mapping users to unique strings; and a transition system semantics where states are first-order structures over the signature. In this project the actual combination of WebML with SecureUML as suggested in [BDL05] is not performed, since WebML itself already describe security models using protected modules. Rather, the WebML language model is extended using authorization constraints similar to [LBD02] in order to fulfil the programmatic access control decision requirements.

An MOF-based metamodel for WebML is described in [WS-WMM]. The WebML Metamodel 0.1.1 describes WebML as consisting of the packages `Localization`, `Mapping`, `Navigation`, `Structure` and `Auxiliary`, as well as the elements `WebML`, `Property` and `Comment`. The `WebML` element is the root element of a WebML model instance which contains all other elements of the model. The `Property` element represents a property as a pair of name and value, while the `Comment` element provides additional information on an element. The `Localization` package provides classes for representing possible settings according to a geographical location.

The `Mapping` package contains classes for mapping entities to relational database management systems. The `Navigation` package contains sub-packages each responsible for the presentation, navigation, modification, and access of content. The `Structure` package contains classes for representing the organization and specification of data. The `Auxiliary` package is specific to the WebRatio implementation and contains entities used within for auxiliary purposes. Fig. 22 shows the WebML Metamodel 0.1.1 with second-level elements and sub-packages of the `Navigation` package.

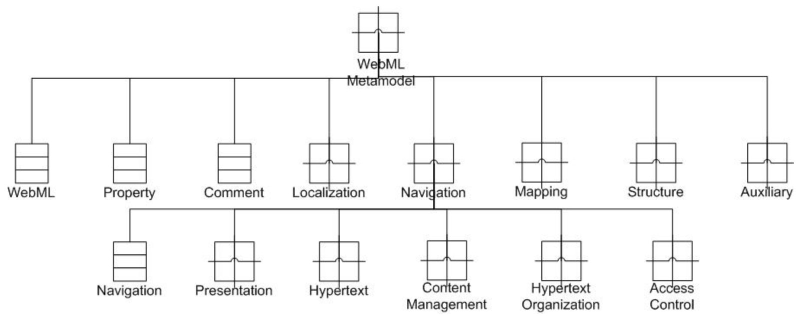


Fig. 22. WebML Metamodel 0.1.1

In the `Navigation` package, the content units of the hypertext model are contained within the `Hypertext` sub-package, while the operation units are defined in `ContentManagement`, `HypertextOrganization`, and `AccessControl` sub-packages. The `Hypertext` sub-package also contains elements supporting the content units, such as `Selector` and `ValidationRule`. To extend the WebML metamodel to support authorization constraints, an element `Authorization` in introduced to the `Hypertext` sub-package. The `Authorization` element selects instances due to conditions, much like the `Selector` element. Therefore it also contains the attributes `AuthorizationCondition` derived from `SelectorCondition`, `BooleanOperator` to handle the operation of multiple `AuthorizationConditions`, and `DefaultPolicy` to define default behaviour if no input parameters are available for all `AuthorizationConditions`. The proposed extension to the `Hypertext` sub-package is shown in Fig. 23.

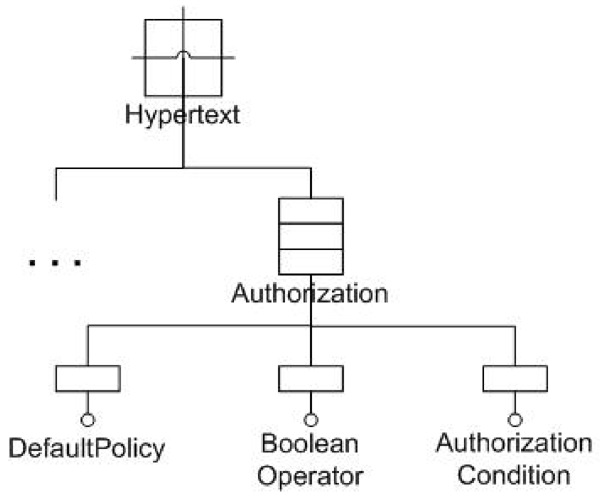


Fig. 23. Extension to the Hypertext sub-package of the WebML Metamodel 0.1.1

In order to merge the authorization constraints of SecureUML to the concrete syntax of WebML, a profile of WebML in the native language of SecureUML, which is UML, needs to be presented. Since WebML is based on UML, a profiling of WebML models in UML is possible. A UML 2.0 profile for WebML exists in [MFV06]. OCL 2.0, which is the constraint language used in SecureUML is also part of the UML 2.0 Specification [UMLs, UMLi]. A comparison of a WebML native diagram and its UML 2.0 profile for a running example is reproduced in Fig.14. As discussed in [MFV06], the UML 2.0 profile of a WebML diagram contains more models and instances of models compared to its native WebML. This is due to the fact that WebML models represent software artefacts which in reality stand apart and reside in different tiers.

The WebML diagram shown in Fig. 14 represents a `ClassPage` page containing two units: the `AllClasses` index unit and the `ClassDetails` data unit, both sourcing from the entity `Class`. An automatic link navigates from `AllClasses` to `ClassDetails` carrying the parameter `X` containing the selected class's OID. `ClassDetails` has a `Selector` taking the link parameter `X` to specify which instance of `Class` is displayed. This result in the automatic display of the first `Class` on the list in the `ClassDetails` upon navigation to the `ClassPage`, and by clicking an index entry in `AllClasses` the user can select which class's details are to be displayed.

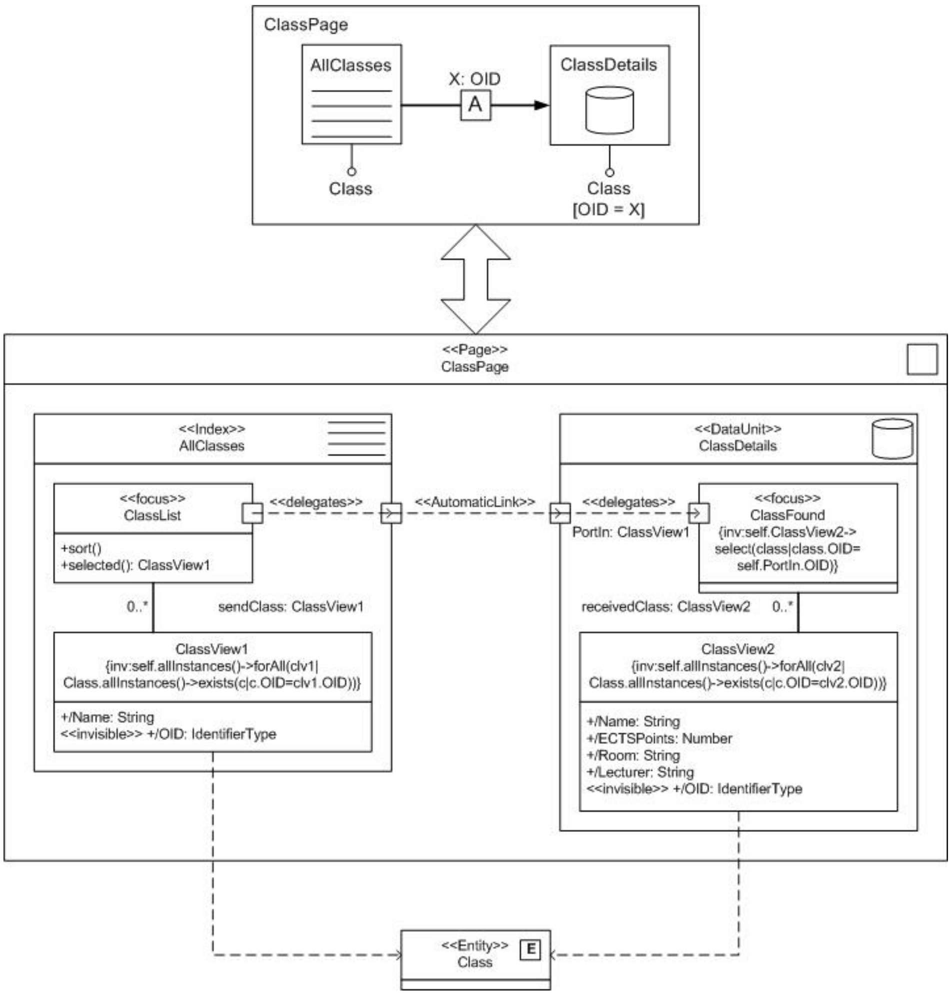


Fig. 14. Comparison of a WebML diagram and its UML 2.0 profile

In the UML 2.0 Profile depicted, the `ClassPage`, represented as a classifier, contains an index unit component and a data unit component, linked by an assembly connector with the `<<AutomaticLink>>` stereotype. The internal structure of the index unit is realized by a focus class, comprising methods for sorting the index instances and for selecting one instance. The focus class is connected by a one-to-many part-of association to class *ClassView1*, which represents a view over the data model entity *Class*. Instances of class *ClassView1* contain the *Name* attribute, necessary to build the index, and the hidden attribute *OID*, necessary for parameter passing. A delegation connector links the output port of the focus class to the outport port of the component, and specifies that the output value of the *select()* method is emitted by the index unit component's output port. The parameter associated with the `<<AutomaticLink>>` connector is received at the input port of the data unit component, which delegates its treatment to an inner focus class. The focus class contains on instance of class *ClassView2*, which represents another view over the data model entity *Class*. An OCL invariant in the focus class enforces the contained instance of class *ClassView2* to have the value of the OID attribute equal to the parameter value received at the input port.

5.3 Role-Based Access Control Augmented by Authorization Constraints

The concept of SecureUML is described in Chapter 2.6. In SecureUML, role-based access control for UML diagrams is strengthened with authorization constraints to capture business logic which is otherwise not possible to describe in the diagrams. As described in Chapter 4, WebML already possesses role-based access control in the form of site views or modules. In order to capture the "fine-graining" of roles within roles, a similar augmentation with constraints is proposed.

According to the SecureUML metamodel, for each UML model element there are permissions linked to user role and action type, and authorization constraints are attached to the model element to capture business logic. An example can be viewed in [LBD02]. In [MFV06], each WebML unit concerned with extracting and publishing data consists of two auxiliary classes: one for defining core logic or control behaviour of the unit, and another one for selecting content from the data model.

Selection constraints already exist in WebML using the constraint language WebML-OQL [WR-WMLg], derived from OCL. By applying a selection constraint to the core logic class the data set displayed by a WebML content unit can be specified. Authorization constraints can additionally be applied to determine whether a user has rights to access the particular unit. Fig. 15 shows an example of selection and authorization constraint each written in OCL and its equivalent notation in WebML-OQL. As previously explained, the selection constraint enforces the contained instance in the focus class to have the same OID attribute value as the one supplied in the input port.

```
/* Selection constraint from Fig. 14. */
context: ClassFound
inv: self.ClassView2->select(class|class.OID = self.PortIn.OID)
/* WebML-OQL equivalent */
Class WHERE Class.OID = X

/*  Authorization  constraint  to  allow  viewing  if  User’s  total
ECTSPoints from Classes participated <= 30 */
context: ClassFound
inv: self.PortIn2.value <= 30
/* WebML-OQL equivalent */
Class WHERE sum(CurrentUser.UserToClass.ECTSPoints) <= 30
```

Fig. 15. Selection and authorization constraints example in OCL and WebML-OQL.

The authorization constraint takes a second input port from a `CurrentUser` focus class linked to a class representing a view over the data model entity `User` with the focus on the current user. This constraint requires that the `CurrentUser`’s total `ECTSPoints` from the `Classes` participated is less than 30. By combining both constraints using the `AND` logical operator, the `Class` instance will be selected according to the specified OID and if the total `ECTSPoints` of the `CurrentUser` is less than 30. The combined constraints written in OCL and the equivalent WebML-OQL expression are shown in Fig. 16. In the selection constraints, the context is limited to the entity in question. Therefore a selection constraint may only constrain attributes belonging to the entity referenced by the content unit, as well as relationships involving the entity.

```
/* In OCL */
context: ClassFound
inv:  self.ClassView2->select(class|class.OID  =  self.PortIn.OID)  and
self.PortIn2.ECTSPoints <= 30
/* In WebML-OQL */
Class  WHERE  Class.OID  =  X  AND  sum(CurrentUser.UserToClass.ECTSPoints)
<= 30
```

Fig. 16. Combined selection and authorization constraints in OCL and WebML-OQL

For the authorization constraints, however, additional information regarding the state of the current user or current time is added to the context. The authorization constraints may therefore constrain attributes or relationships not directly related to the entity in question, but related to the sub-role defining attributes or relationships. These attributes and relationships in turn are limited to the `User` entity, as the logic for determining the sub-roles of each user is described by an attribute belonging to or a relationship involving the `User` entity. Fig. 17 shows the UML 2.0 Profile of the WebML model in Fig. 14 with the addition of an authorization constraint described in Fig. 15.

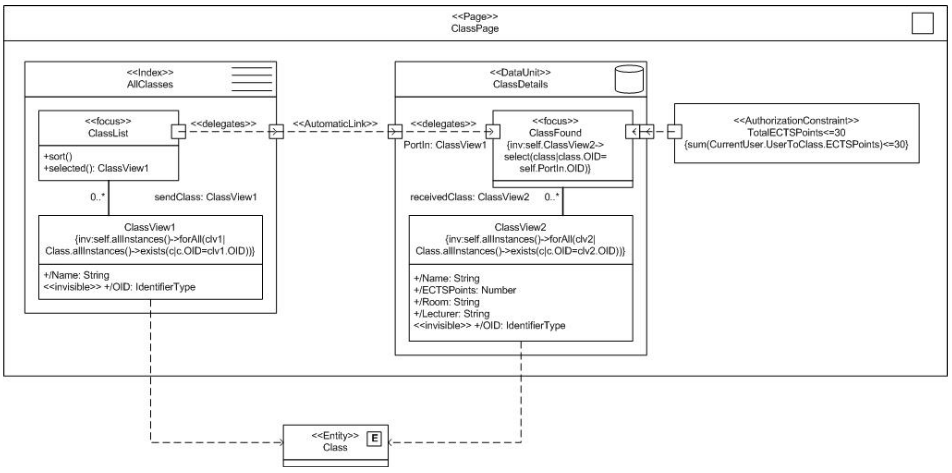


Fig. 17. Authorization constraint in the UML 2.0 Profile of a WebML model

Returning to the WebML models, selector constraints are modelled in the textual notation using the “`selector`” keyword, while in the graphical representation they are shown under each data source using square brackets, as can be seen in the previous example in Figure. The authorization constraints also follow similar modelling: in the textual notation a keyword such as “`authorization`” can be used, whereas in the graphical representation a similar square-bracket notation is possible. Fig. 18 shows the WebML model transformed from the UML 2.0 Profile of Fig. 17.

In the example in Fig. 18, the authorization constraint is described in the graphical representation by adding a constraint line to the `ClassDetails` data unit. The authorization constraint is distinguished from the selection constraint by the keyword `AUTH`.

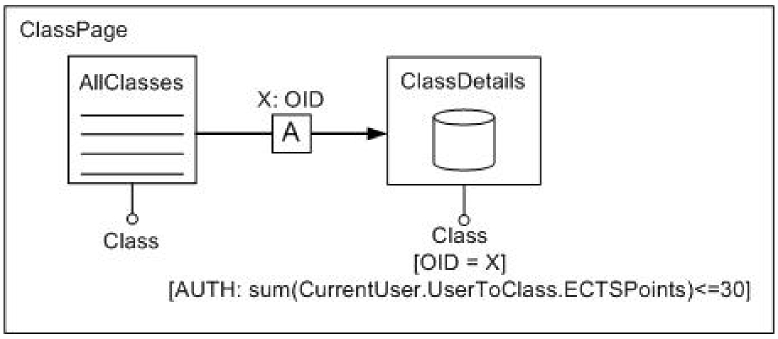


Fig. 18. Example of authorization constraint in a WebML model

This representation follows that of `pre-selectors` in the multichoice index unit implemented in WebRatio using the keyword `PRE` in order to distinguish the pre-selectors from selectors [WR-WRg]. Fig. 19 shows a possible textual representation to the WebML model described in the example in Fig. 18.

```
Page ClassPage
(units AllClasses, ClassDetails)

IndexUnit AllClasses
(source Class;
attributes Name;
orderBy Name)

Link AllClassesToClassDetails automatic
(from AllClasses to ClassDetails;
parameters X:OID)

DataUnit ClassDetails
(source Class;
selector OID = X;
authorization sum(CurrentUser.UserToClass.ECTSPoints) <= 30;
attributes Name, ECTSPoints, Room, Lecturer)
```

Fig. 19. The textual representation of the WebML model described in Fig. 18

Using the authorization constraints, a solution for the group membership issue of the case study discussed in Chapter 3 can be realised. By adding the respective authorization constraints to units only authorized for group members and group moderators, the modelling of the group pages can be reduced to one page, also eliminating the navigation-directing queries depicted in Fig. 24 previously needed. The proposed solution for this issue is shown in Fig. 25.

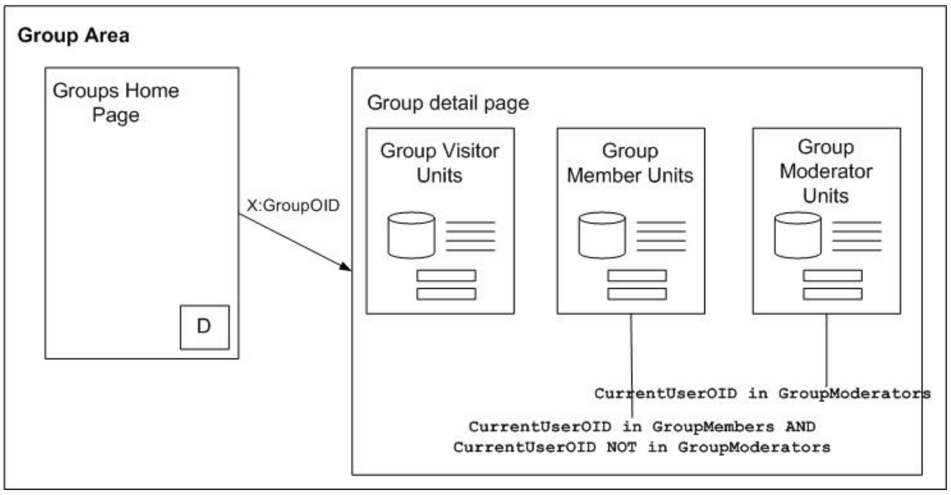


Fig. 25. Proposed solution for the group membership requirement in the case study

This proposed solution is presented in a simplified WebML hypertext model. In the solution, the `Group Area` only consists of two pages, the `GroupsHome` page and the `GroupDetail` page. The `GroupsHome` page contains an index unit listing all available `Groups`. When a user clicks on one of the entries in this list, the navigation is directed to the `GroupDetail` page with the `GroupOID` carried as a parameter. Within the `GroupDetail` page the current user's OID, provided by a `Get` unit, is used to compute all of the protected units in the page according to each authorization constraint. The constraints are formulated so that it reflects the access control requirement for group membership. The constraint for Group Member units is `CurrentUserOID IN GroupMembers AND CurrentUserOID NOT IN GroupModerators`, while the constraint for Group Moderator units is `CurrentUserOID IN GroupModerators`. The Group Visitor units are not constrained; therefore it is always visible and available regardless to the state of the current user. This means that a visitor not belonging to any of the two relationships can only access units which are categorized in the Group Visitor units, a group member can additionally access units designated by Group Member units, and a group moderator can furthermore use the Group Moderator units.

Authorization constraints can also be used to fulfil the requirement of variable access level settings for different groups of data in one viewing page described in Chapter 3. The proposed solution is depicted in Fig. 26.

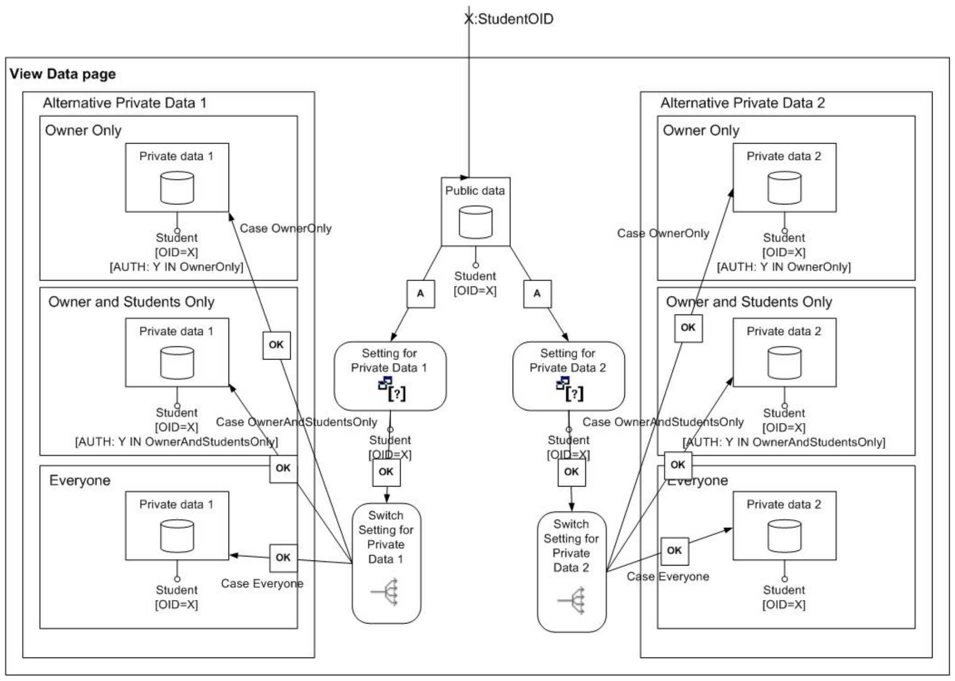


Fig. 26. Proposed solution for the variable access levels settings of multiple data groups requirement in the case study

In this proposed solution, a page serving the requirement of viewing a `Student`'s profile is presented. The `Student` profile data is divided into three groups: the `PublicData`, the `PrivateData1` and the `PrivateData2`. Each of this group of data is sourced from the `Student` entity; additionally an attribute each in the `Student` entity also determines the setting of access level for `PrivateData1` and `PrivateData2`. Access to any of the two groups of private data can be set independently according to the roles already described in Chapter 5.1 (`OwnerOnly`, `OwnerAndOtherStudentsOnly`, `Everyone`).

The solution works as follows: first, an incoming link directs navigation to the View Data page, carrying the selected `StudentOID` as a parameter. The first stop is the `PublicData` data unit, which is not constrained and displays the group of data deemed public. Next, two automatic links lead to similar paths for each of the private data blocks. The automatic links ensure that the navigation path is automatically initiated. Taking the path to `PrivateData1`, the link reaches a `Selector` unit named `Setting_for_Private_Data_1`, which selects the preference of the profile owner concerning the access level for the `PrivateData1` block and outputs it as a parameter. The output of this `Selector` unit is fed to a `Switch` unit named `Switch_Setting_for_Private_Data_1`. This `Switch` unit determines the OK-link to follow on the basis of the input parameter from the `Selector` unit. There are three OK-links originating from the `Switch` unit to an alternative page defined in `Alternative_Private_Data_1`, each corresponding to the possible access level settings. If the profile owner has set the access level for `PrivateData1` block as `OwnerOnly`, the OK-link with the label `Case OwnerOnly` is followed. Inside each of the alternative pages there is a data unit named `PrivateData1` displaying the data block for `PrivateData1`. An authorization constraint is assigned to each data unit according to the alternative's purpose. For instance, the alternative `OwnerAndOtherStudentsOnly` has the authorization constraint `Y IN OwnerAndStudentsOnly`, meaning that the `StudentOID` of the current user accessing the page is computed as `Y` and looked up in the `OwnerAndStudentsOnly` relationship for the selected `Student` profile. The usage of alternatives ensures that only one of the three possible data units is shown at one time. During the propagation along the path, the selected `StudentOID` is always carried as a parameter in the links. A similar path is followed for `PrivateData2`, resulting in independent switching of access levels for each private data block according to the profile owner's preferences.

Considering the fact that the mapping of roles in this case leads to scalability problems as described in Chapter 5.1, this solution is also not efficient. On the other hand this solution already enables mapping of such type of dynamic roles within the WebML models, providing a high-level view to the business logic of variable access level settings in multiple groups of data and bringing the model closer to the implemented code.

6 Related work

Formatted: Bullets and Numbering

6.1 Model-based Tools

Some CASE tools implementing UML are described in [Gol05], [EMF], and [TOG]. These tools do not support the implementation of WebML, therefore they are not considered competitors of WebRatio. A UML-based tool implementing the security designs of SecureUML is also presented in [BDW06]. This tool also does not support WebML.

WebML is proposed in several projects and planned to be implemented in tools as part of these projects. The WebSI project [WebSI] aims to develop three suites of tools for designing data-intensive web applications in the ASP framework. The Multi-channel Adaptive Information System project [MAIS] was targeted to develop reference models, architectures and prototypes to provide a flexible environment to adapt the interaction and provided interaction according to the changing requirements, execution contexts, and user needs of various kinds of interaction devices (PC, laptop, palmtop, cellular phone, TV sets and others). The Collaborative Open Environment for Project-Centered Learning project [COOPER] is dedicated to support long-distance cooperation of students working on projects. As part of this project the usage of WebML in process modeling is proposed [BCFM06]. The W3I3 tool suite [W3I3] and its project aim to propose a model-driven approach to Web site design especially suited for multi-device, mobile e-commerce applications. These projects are either still in development or already ceased, and the tools resulting from them are not readily available. This strengthens the reason for selecting WebRatio as the most advanced and market-ready tool implementing WebML. Moreover, none of these tools support the J2EE framework as WebRatio does.

Formatted: Bullets and Numbering

6.2 Extensions to WebML

Extensions to WebML can be found in several stages of development. The first and most-researched is in extending WebML to support workflow-driven web applications, for which a collection of work is available in [WWF]. In [BCCF02] a workflow model is introduced to WebML, along with additional supporting units in order to accurately model workflow-driven hypertexts. In [BT05], the workflow data model is extended and some other units are added to support exception handling within workflow-based web applications. [Bra06] proposes a visual editing tool, based on BPMN notations, that allows modeling of workflow models which is integrated to WebRatio to enable seamless design of data and hypertext models. A demonstration of WebML with workflow- and Web services-extensions is described in [BCCDFM04].

Another area of development is in the supporting of Web services in WebML. A collection of papers in this area is available in [WWS]. In [BCCFM02] a data model to support the definition of Web services in WebML along with some units is discussed. This work also extends the workflow model described in [BCCF02] to support workflow-driven Web services. [BCCFM03] provides an overview of some architectural issues raising form the integration of data-intensive web applications and web services by examples.

Other relatively new development areas include modeling context-aware web applications in WebML. In the proposal of [CDMF07], an extension to WebML with context-aware data model and units is introduced. This proposal is applied within the MAIS project [MAIS] in a location-aware tourist information system.

The extensions to WebML concern areas of development other than access control. Especially limited is the amount of related work on defining dynamic roles in social networking applications in any modeling language, let alone WebML. Therefore this project considers related work on access control in UML-based modeling languages, such as [LBD02].

7 Future Work and Conclusion

Formatted: Bullets and Numbering

7.1 Conclusion

The practical part of this master thesis has demonstrated the capabilities of WebML, combined with the supporting CASE tool WebRatio, in creating a collaborative, social-networking web-based application from the design phase up to the executable code, and thus making a strong argument in favor of *model compilers* technology in general. Some workaround involving a navigation-directing combination of units in the hypertext model and in the end manual coding is necessary in order to satisfy the requirements of access control. The introduction of authorization constraints in the hypertext model allows developers to solve this type of workaround, simplifying the model and enabling the business logic describing fine-graining of user roles such as groups-within-groups to be specified directly within the models.

In addition to presenting solutions to the problems discussed in the case study, so-called dynamic roles have been proposed in this thesis, as a useful addition to the well-known static roles. A conceptual mapping of dynamic roles and its implementation in WebML are also presented. Furthermore, the extension of the WebML metamodel in incorporating authorization constraints has also been described. Finally, the concrete syntax of WebML has been extended to support the notation of authorization constraints.

Formatted: Bullets and Numbering

7.2 Future Work

Further work stemming from this project involves exploring the addition of authorization constraints in the WebML data model. Specification of authorization constraints in the data model enables automatic assignment of authorization constraints in the hypertext model with the help of appropriate transformation rules (a task to be supported by the model compiler). Therefore specifying the relationship between the data model and the hypertext model in the context of access control policies is also a focus for further research.

As discussed in the related work on extensions section of Chapter 6, the amount of work done in defining and mapping dynamic roles is still limited. Therefore mapping of such roles in other modeling language can be a starting point for future work. The definition of dynamic roles in this project may be improved to yield more efficient implementations, particularly in complex dynamic role scenarios such as the variable access levels for multiple data-groups case.

In [LBD02] authorization constraints are not only used to facilitate programmatic access control decisions, but also to enforce business rules which are difficult or impossible to model in the application model. Future work can then be done in exploring the possibilities of applying authorization constraints to incorporate business rules in WebML models.

On the implementation side, further work can be done in incorporating the authorization constraints into existing and future WebRatio units. The authorization constraints can even be assigned to unit containers such as pages and areas in order to serve a broader set of requirements concerning access control in sub-roles. An authorization wizard with similar interface to the wizards available in WebRatio may also be developed to aid developers in designing access control policies to their applications.

References

[BCCDFM04] Brambilla, M.; Ceri, S.; Comai, S.; Dario, M.; Fraternali, P.; Manolescu, I.: Designing Data-Intensive Web Applications. Institute for Computer Science, University of Freiburg, Germany, 2002.

[BCCF02] Brambilla, M.; Ceri, S.; Comai, S.; Fraternali, P.: Specification and Design of Workflow-Driven Hypertexts. Journal of Web Engineering (JWE) Vol. 1 Number 1, 2002.

[BCCFM02] Brambilla, M.; Ceri, S.; Comai, S.; Fraternali, P.; Manolescu, I.: Model-driven Specification of Web Services Composition and Integration with Data-intensive Web Applications. IEEE Bulletin of Data Engineering, December 2002.

[BCCFM03] Brambilla, M.; Ceri, S.; Comai, S.; Fraternali, P.; Manolescu, I.: Model-driven Development of Web Services and Hypertext Applications. SCI2003, Orlando, Fla., USA, 2003.

[BCFM06] Brambilla, M.; Ceri, S.; Fraternali, P.; Manolescu, I.: Process Modelling in Web Applications. ACM-TOSEM, October 2006.

[BDL05] Basin, D.; Doser, J.; Lodderstedt, T.: Model-Driven Security: From UML Models to Access Control Infrastructures. ETH Zürich / Interactive Objects Software GmbH Freiburg, 2005.

[BDW06] Brucker, A. D.; Doser, J.; Wolff, B.: A Model Transformation Semantics and Analysis Methodology for SecureUML. Tech. Report 524, ETH Zürich, 2006.

[Boe88] Boehm, B.: A Spiral Model of Software Development and Enhancement. IEEE, 1988.

[Bra06] Brambilla, M.: Generation of WebML Web Application Models from Business Process Specifications. ICWE2006, Menlo Park, CA, USA, 2006.

[BT05] Brambilla, M.; Tziviskou, C.: Fundamentals of Exception Handling within Workflow-based Web Applications. Journal of Web Engineering (JWE) Vol. 4 Issue 1, March 2005.

[CDMF07] Ceri, S.; Daniel, F.; Matera, M.; Facca, F.: Model-Driven Development of Context-aware Web Applications. ACM Transactions on Internet Technology (TOIT), Volume 7, Number 2, May 2007.

[Cer06] Ceri, S.: Process Modeling in Web Applications. ACM Trans. on Softw. Eng. and Methodology, 15(4):360–409, 2006.

[CF01] Comai, S.; Fraternali, P.: A Semantic Model for Specifying Data-Intensive Web Applications using WebML. Semantic Web Workshop, Stanford, US, July 2001.

[CFBBCM03] Ceri, S.; Fraternali, P.; Bongio, A.; Brambilla, M.; Comai, S.; Matera, M.: Declarative Specification of Web Applications Exploiting Web Services and Workflows. ACM SIGMOD/PODS 2004 Conference, Paris, France, 2004.

[COOPER] Collaborative Open Environment for Project-Centered Learning (COOPER) EU-Initiative, http://cooper-project.org/index.html. Last accessed: 2 May 2007.

[DB2] IBM Software –DB2 Product Family – Family Overview, http://www.ibm.com/db2. Last accessed: 2 May 2007.

[DoD85] U.S. Department of Defense: Trusted Computer System Evaluation Criteria, DOD 5200.28-STD. U.S. Department of Defense, 1992.

[EMF] Eclipse Modeling Framework Project, http://www.eclipse.org/modeling/emf/?project=emf. Last accessed: 16 April 2007.

[FK92] Ferraiolo, D.; Kuhn, R.: Role-Based Access Control. Proceedings of 15th National Computer Security Conference, 1992.

[Flkr] Flickr – Photo Sharing, http://www.flickr.com. Last accessed: 15 April 2007.

[Gol05] Goldberg, B.: The DASL Language: Programmer's Guide and Reference Manual. Tech. Report TR- 2005-128, Sun Microsystems Research Labs, 2005.

[JS05] Judd, C.; Shitu, H.: Pro Eclipse JST: Plug-ins for J2EE Development. Springer-Verlag New York, 2005.

[J2EE] Sun Developer Network: Java EE at a Glance. Sun Microsystems, http://java.sun.com/javaee/ (Last accessed: 22 April 2007)

[J2SE] Sun Developer Network: Java SE at a Glance. Sun Microsystems, http://java.sun.com/javase/ (Last accessed: 22 April 2007)

[KW05] Klepke, A; Warmer, J: Getting Started with Modeling Maturity Levels. Jupitermedia Corporation, 2005. http://www.devx.com/enterprise/Article/26664 (Last accessed: 19 April 2007).

[LBD02] Lodderstedt, T.; Basin, D.; Doser, J.: SecureUML: A UML-Based Modeling Language for Model-Driven Security. Morgan Kaufmann Publishers, USA, 2003.

[MAIS] Multi-channel Adaptive Information Systems project page, http://black.elet.polimi.it/mais/index.php. Last accessed: 2 May 2007.

[MFV06] Moreno, N.; Fraternali, P.; Vallecillo, A.: A UML 2.0 Profile for WebML Modeling. Model-Driven Web Engineering Workshop, Palo Alto, California, July 11, 2006.

[MMCF03] Matera, M.; Maurino, A.; Ceri, S.; Fraternali, P.: Model-driven design of collaborative web applications. Software – Practice & Experience, 2003: 33: 1 - 32.

[MSA] Microsoft Access Homepage – Microsoft Office Online, http://office.microsoft.com/access/default.aspx. Last accessed: 2 May 2007.

[MSSQL] Microsoft SQL Server 2005 Home, http://www.microsoft.com/sql/default.mspx. Last accessed: 2 May 2007.

[MySp] MySpace, http://www.myspace.com. Last accessed: 15 April 2007.

[MySQL] MySQL AB: Developer Zone, http://www.mysql.org/. Last accessed: 2 May 2007.

[OCL2] Object Management Group: OCL 2.0, OMG Final Adopted Specification ptc/03-10-14. Object Management Group, 2003.

[Orcl] Oracle Home, http://www.oracle.com/index.html. Last accessed: 2 May 2007.

[Orkt] Orkut, http://www.orkut.com. Last accessed: 15 April 2007.

[PGSQL] PostgreSQL: the world's most advanced open source database, http://www.postgresql.org/. Last accessed: 2 May 2007.

[TOG] Borland Together Visual Modeling Tool for Software Design, http://www.borland.com/us/products/together/index.html. Last accessed: 16 April 2007.

[UML2i] Object Management Group: Unified Modeling Language Infrastructure, version 2.0. formal/05-07-05. Object Management Group, March 2006.

[UML2s] Object Management Group: Unified Modeling Language Superstructure, version 2.0. formal/05-07-04. Object Management Group, August 2005.

[WebSI] WebSI: Data-centric Web Services Integrator, http://www.ib-ia.com/websi/html/home.htm. Last accessed: 2 May 2007.

[WK03] Warmer, J.; Klepke, A.: The Object Constraint Language: Getting Your Models Ready for MDA. 2nd Edition. Addison-Wesley, 2003.

[WR] WebRatio Home. http://www.webratio.com (Last accessed: 23 April 2007)

[WR-AFT06] The Web Ratio Team: Advanced Features Tutorial for Web Ratio 4.3. Web Models s.r.l., 7th November 2006.

[WR-ART06] The Web Ratio Team: Access Rights Tutorial for Web Ratio 4.3. Web Models s.r.l., 7th November 2006.

[WR-CUG06] The Web Ratio Team: Custom Units Tutorial and Reference Guide for Web Ratio 4.3. Web Models s.r.l., 7th November 2006.

[WR-WMLg] The WebRatio Team: WebML Guide for WebRatio 4.3. Web Models s.r.l., 7th November 2006.

[WR-WRg] The WebRatio Team: WebRatio 4.3 User Guide. Web Models s.r.l., 7th November 2006.

[WS-WMM] Wimmer, M.; Schauerhuber A.: A Metamodel for Modelling Web Applications. Technical University of Vienna. http://www.big.tuwien.ac.at/projects/webml/ (Last accessed: 2 May 2007)

[WWF] WebML Papers on Workflow-Driven Web Applications. WebML.org. http://www.webml.org/webml/page41.do?dau47.oid=13&UserCtxParam=0&GroupCtxParam=0&ctx1=EN (Last accessed: 23 April 2007)

[WWS] WebML Papers on Web Services. WebML.org. http://www.webml.org/webml/page41.do?dau47.oid=11&UserCtxParam=0&GroupCtxParam=0&ctx1=EN (Last accessed: 2 May 2007)

[W3I3] Bonifati, A.; Ceri, S.; Fraternali, P.; Maurino, A.: Building Multi-device, Content-centric Applications using WebML and the W3I3 Tool Suite. Proceedings from Conceptual Modeling for E-Business and the Web: ER 2000 Workshops on Conceptual Modeling Approaches for E-Business and The World Wide Web and Conceptual Modeling, Salt Lake City, Utah, USA, October 2000.

[Xing] XING, https://www.xing.com/. Last accessed: 15 April 2007.

[YouT] YouTube, http://youtube.com. Last accessed: 15 April 2007.

Appendix A: WebML-OQL Syntax

```
<DIGIT: ["0"-"9"]>
<LETTER:
[ "\u0024", "\u0041"-"\u005a", "\u005f", "\u0061"-"\u007a",
"\u00c0"-"\u00d6", "\u00d8"-"\u00f6", "\u00f8"-"\u00ff",
"\u0100"-"\u1fff", "\u3040"-"\u318f", "\u3300"-"\u337f",
"\u3400"-"\u3d2d", "\u4e00"-"\u9fff", "\uf900"-"\ufaff" ]>
<STRING: "\’" (<LETTER> | <DIGIT> |
"@" | " " | "!" | "?" | "#" | "$" | "£" | "%" | "/" | "^" |
"|" | "[" | "]" | "," | ";" | "." | ":" | "_" | "-" | "+" |
"*" | "§" | "´Y")+ "\’">
<NUMBER: (<DIGIT>)+ ("." (<DIGIT>)+)?>
<IDENTIFIER: (<LETTER>)+ ("_" | ":" | <DIGIT> | <LETTER>)*>
<OPERATOR: "+" | "-" | "/" | "*">
<COMPARATOR: "<" | "<=" | "=" | "!=" | "<>" | ">=" | ">" |
"contains" | "beginswith" | "endswith">
<AGGRFUNCTION: "min" | "max" | "avg" | "sum" | "count">
<SELF: "Self">
<AND: "AND">
<OR: "OR">
<WHERE: "WHERE">
<ISA: "ISA">
<NOT: "NOT">
<IN: "IN">
<TO: "TO">
<IS: "IS">
<AS: "AS">
<NULL: "NULL">
<TRUE: "TRUE">
<FALSE: "FALSE">
<LEFTBRACKET: "(">
<RIGHTBRACKET: ")">
<DOT: ".">

<EntityQuery : Step <WHERE> Condition ( ";" | <EOF> )>
<RelationshipQuery  :  (  <SELF>  <TO>  Step  |  PathExpression  )  (  <WHERE>
Condition )? ( ";" | <EOF> )>
<AttributeQuery : AttributeValue ( <WHERE> Condition )? ( ";" | <EOF> )>
<Step : <IDENTIFIER> ( <LEFTBRACKET> <AS> <IDENTIFIER> <RIGHTBRACKET> )?>
<PathExpression : ( <SELF> | <IDENTIFIER> ) ( <DOT> Step )*>
<AttributeValue   :   (   AttributeExpression   |   <LEFTBRACKET>   AttributeValue
<RIGHTBRACKET> )
(    <OPERATOR>    (    AttributeExpression    |    <LEFTBRACKET>    AttributeValue
<RIGHTBRACKET> ) )*>
<AttributeExpression : ( <STRING> | <NUMBER> | PathExpression |
<AGGRFUNCTION> <LEFTBRACKET> PathExpression <RIGHTBRACKET> )>
<Member : ( <NOT> )? <IN> PathExpression>
<IsNull : <IS> ( <NOT> )? <NULL>>
<WhereExpression : ( ( <IDENTIFIER> | <SELF> ) <ISA> <IDENTIFIER> |
AttributeExpression ( Member | IsNull |
<COMPARATOR> ( AttributeExpression | <TRUE> | <FALSE> ) ) |
( <LEFTBRACKET> Condition <RIGHTBRACKET> ) )>
<LogicalTerm : WhereExpression ( <AND> WhereExpression )*>
<Condition : LogicalTerm ( <OR> LogicalTerm )*>
```

Appendix B: Use Case List for the Case Study

User registers
A visitor submits a **RegistrationRequest** containing her personal data to the website.

User donates Money
A **User** donates some amount of money to the alumni network through the website.

User views public content
A **User** views the publicly available content of the alumni website.

Student updates Profile
A **Student** updates her preloaded profile in the alumni network database.

User searches Profile
A **User** searches the profile database of **Students** according to search criteria and picks out selected profiles

User views Profile
A **User** views the profile of a selected **Student**. Depending on access level and settings given by the profile's owner, **NITAdministrator** or **NITSponsor** may not be able o see some data.

Student enters Job
A **Student** enters a description of her **Job** which includes chronological as well as geographical information.

Student enters Address
A **Student** enters a description of her **Address** which includes chronological and geographical information.

User sends Message
A **User** sends a **Message** which includes title and body to one or multiple recipients.

Student creates Group
A **Student** creates her own **Group** inside the alumni network which can be categorized, set as private or open groups as well as a marketplace model.

Student becomes a private Group member
A **Student** submits a **GroupMembershipRequest** to a private **Group** she intends to join. Upon approval from the **Group** owner, **Student** becomes a member of the **Group**.

Student creates Topic
Within a specific **Group** a **Student** creates a **Topic** to be discussed and which is visible to all group members.

Student adds Posting to a Topic
A **Student** posts a **Posting** to a **Topic** discussion within a **Group**.

Student creates Album
A **Student** creates an **Album** which can be categorized and may contain media **Files**.

Student adds File to Album
A **Student** uploads a **File** to a specific **Album**.

Student uploads Resume
A **Student** uploads her **Resume** to the alumni network database.

Student answers Poll
A **Student** chooses the relevant **PollOption** for a specific **Poll**.

Student views PollResult
A **Student** views the results of a concluded **Poll**.

Student creates Poll
A **Student** creates a **Poll** in a **Group** to be answered.

AlumniBoardMember approves RegistrationRequest
An **AlumniBoardMember** approves a **RegistrationRequest** submitted by a **User**.

Student creates Event
A **Student** creates an **Event** description which includes chronological and geographical information.

AlumniBoardMember uploads Newsletter
An **AlumniBoardMember** uploads an alumni **Newsletter** for public downloading.

AlumniBoardMember creates Announcement
An **AlumniBoardMember** creates an **Announcement** for public viewing.

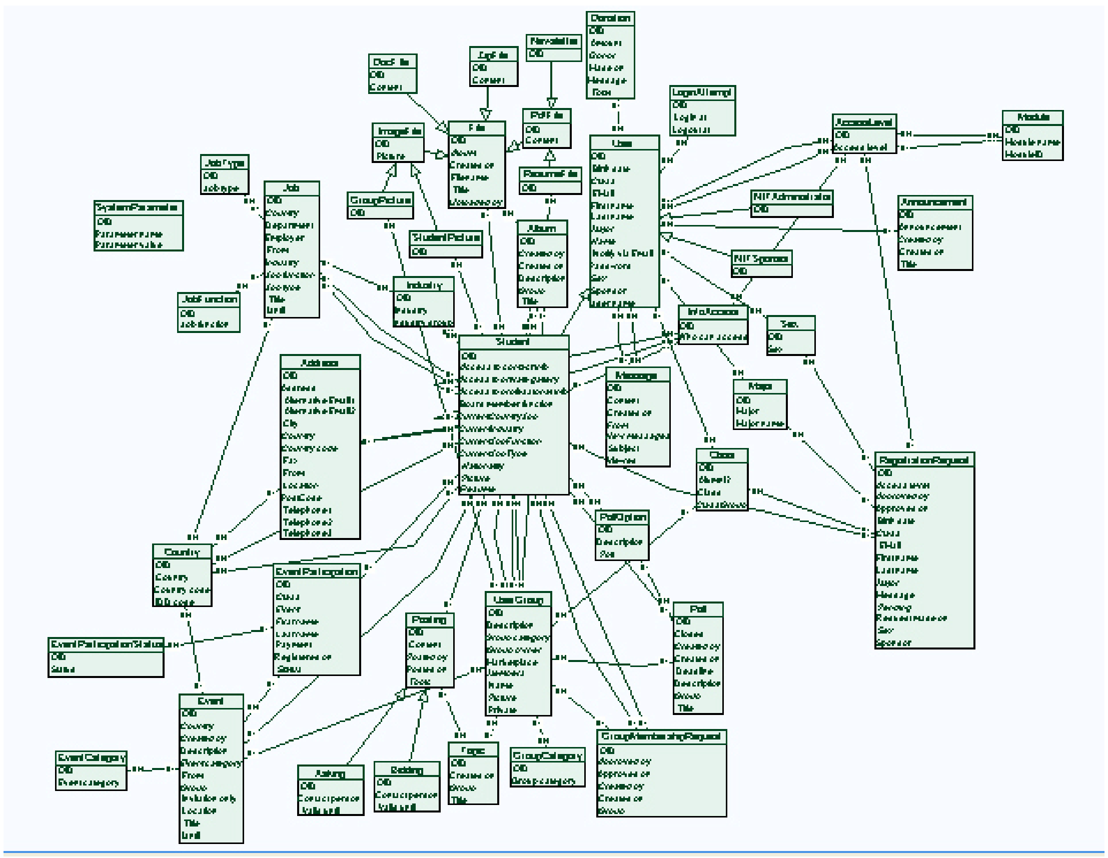
AlumniBoardMember exports search result to Excel
An **AlumniBoardMember** exports a search result into an Excel file.

AlumniBoardMember performs backup
An **AlumniBoardMember** performs backup to the alumni network database.

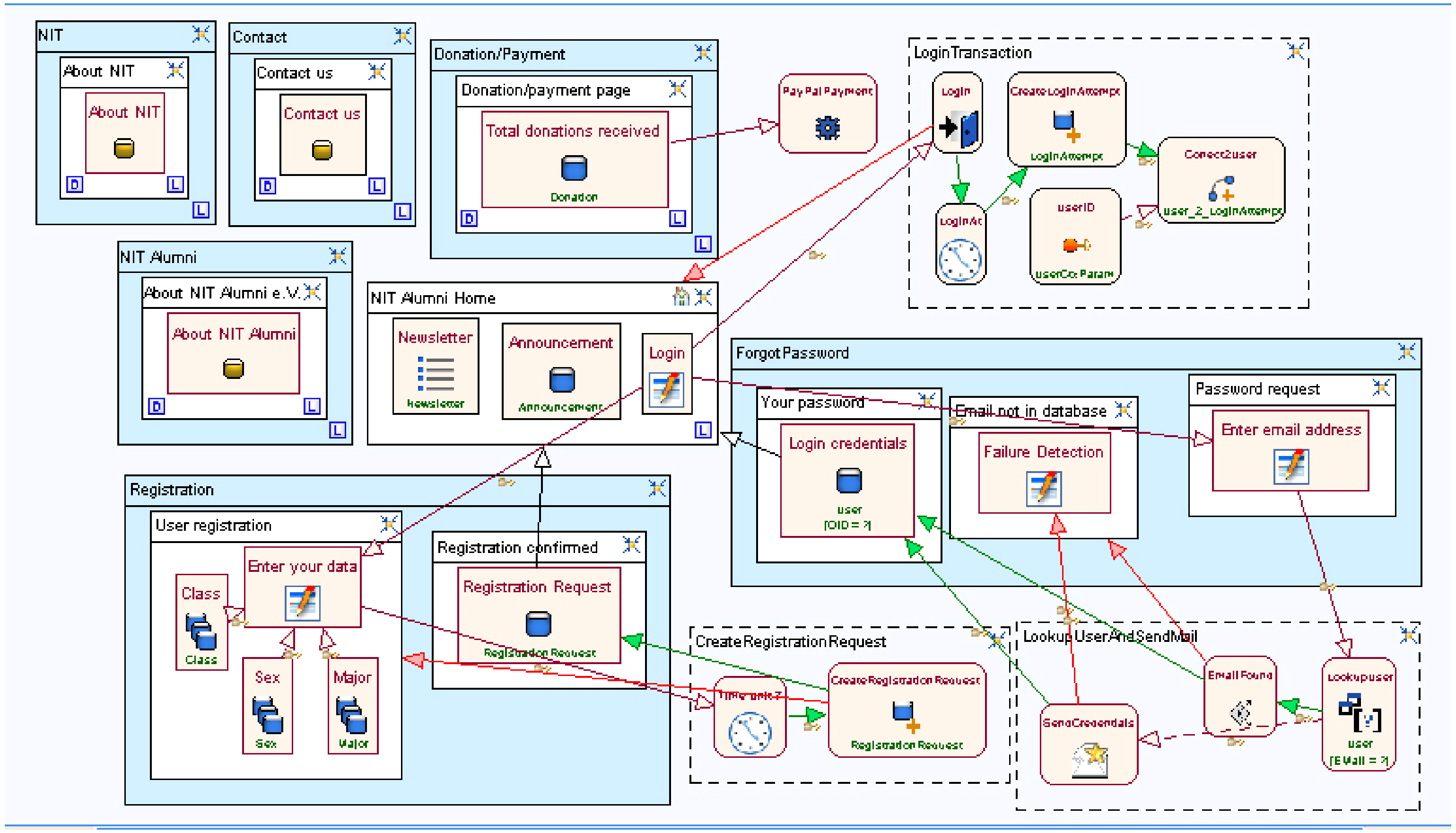
User views Statistics
A **User** views the usage statistics of the website.

AlumniBoardMember deletes User
An **AlumniBoardMember** deletes the profile of a **User** with notification through the email address of the **User**.

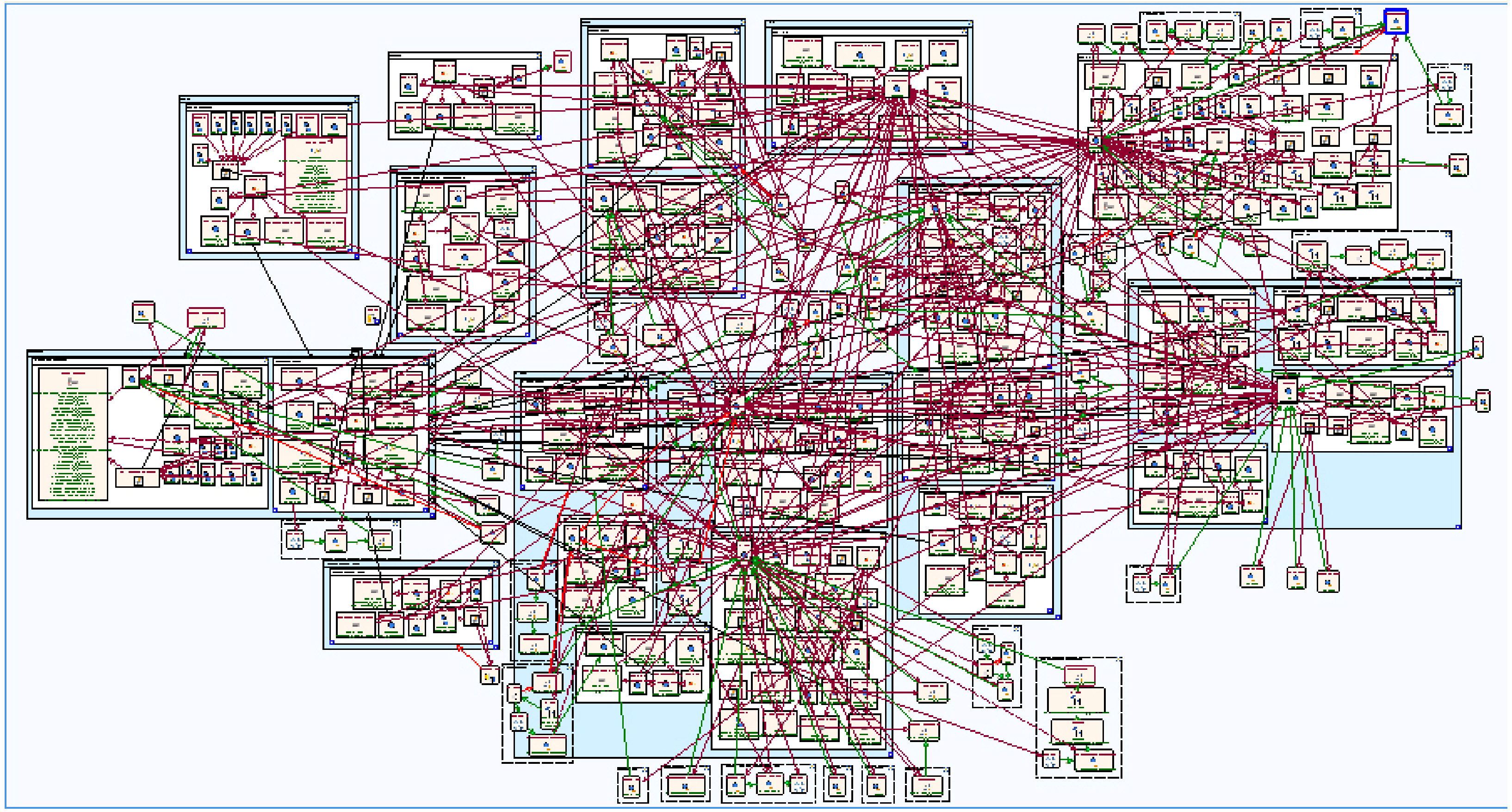
Appendix C: WebML Data Model for the Case Study



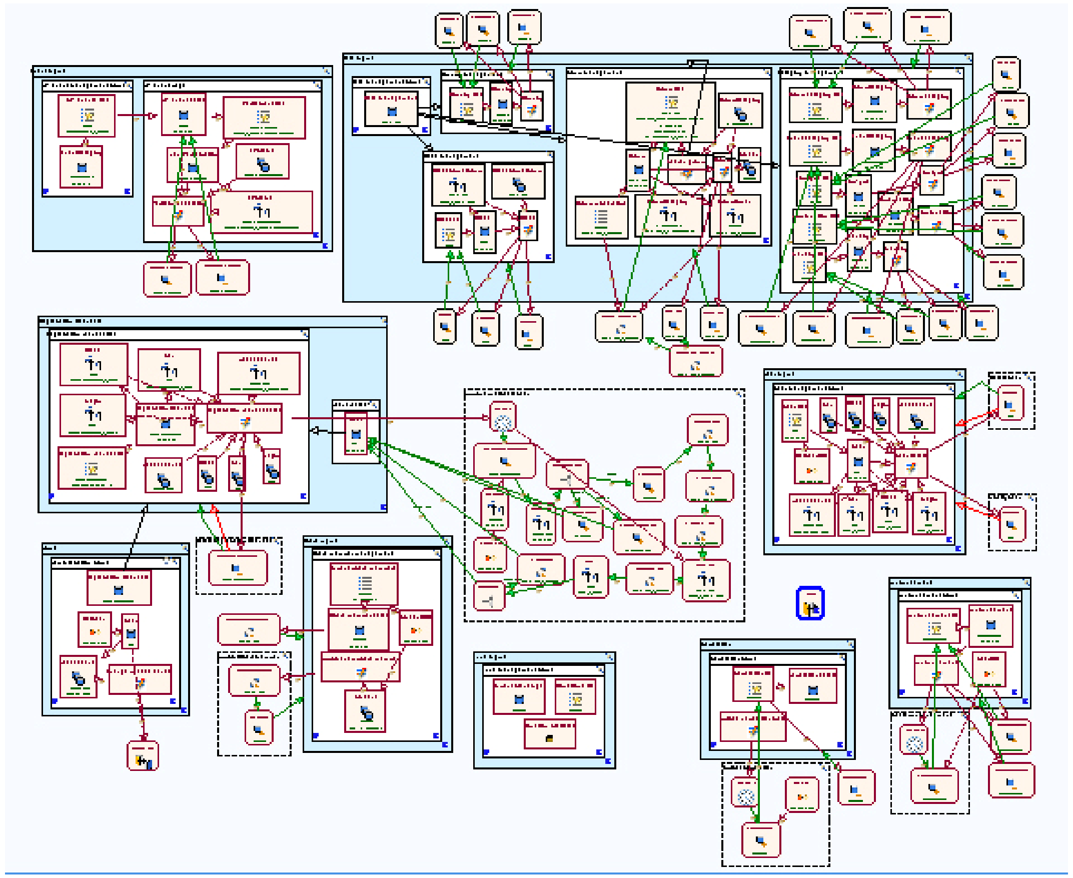
Appendix D-1: WebML Hypertext Model for the Case Study – Public site view



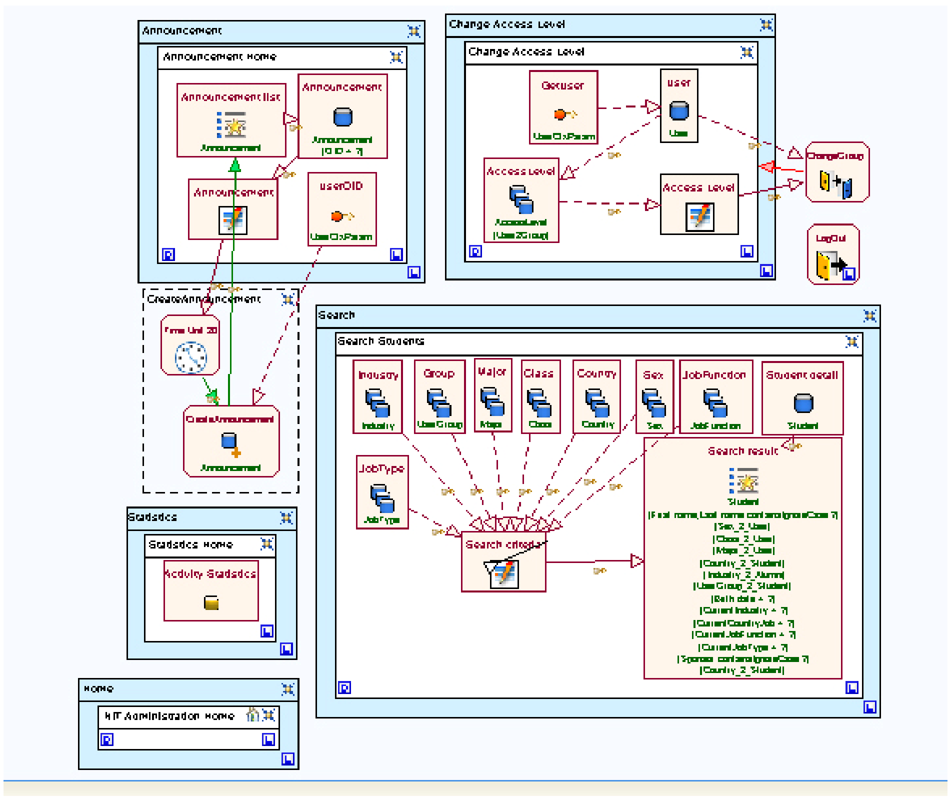
Appendix D-2: WebML Data Model for the Case Study – Student site view



Appendix D-3: WebML Hypertext Model for the Case Study – BoardMember site view



Appendix D-4: WebML Hypertext Model for the Case Study – InstitutionalAdministrator site view


Announcement
Announcement Home
Announcement list
Announcement
Announcement
Announcement
[OID = ?]
Announcement
userOID
UserOIDParam
CreateAnnouncement
Time Unit 30
CreateAnnouncement
Announcement
Change Access Level
Change Access Level
Getuser
UserOIDParam
user
User
Access Level
AccessLevel
[User2Group]
Access Level
ChangeGroup
LogOut
Search
Search Students
Industry
Industry
Group
UserGroup
Major
Major
Class
Class
Country
Country
Sex
Sex
JobFunction
JobFunction
Student detail
Student
JobType
JobType
Search criteria
Search result
Student
[First name,Last name contains(ignoreCase ?)]
[Sex_2_User]
[Class_2_User]
[Major_2_User]
[Country_2_Student]
[Industry_2_Alumni]
[UserGroup_2_Student]
[Birth date = ?]
[Current Industry = ?]
[CurrentCountryJob = ?]
[CurrentJobFunction = ?]
[CurrentJobType = ?]
[Spouse contains(ignoreCase ?)]
[Country_2_Student]
Statistics
Statistics Home
Activity Statistics
Home
NIT Administration Home