

# A data-modelling approach to web application synthesis

Davide Di Ruscio, Henry Muccini, and Alfonso Pierantonio

Dipartimento di Informatica

Università degli Studi dell'Aquila, I-67100 L'Aquila, Italy

## Abstract

Most web applications are data-intensive, i.e. they heavily rely on dynamic contents usually stored in databases. Website design and maintenance can greatly benefit from conceptual descriptions of both data and hypermedia aspects, i.e. those design dimensions which distinguish this application class: the data upon which the content is based, the way dynamic contents are composed together to form pages, and how pages are linked together in order to move across the application content. The paper proposes *Webile*, a visual Domain-Specific Language based on UML, which enables a model-driven approach to high-level specification of web applications. In contrast with other approaches, *Webile* exploits the UML meta-model architecture by serialising the specifications in the XMI interchange format. This representation provides interoperability amongst different operative platforms and enables a XSL transformation-based automatic generation of the applications being designed.

## 1 Introduction

Increasingly, e-business sites rely on dynamic contents usually stored in database because a much wider range of interaction than static HTML pages is achievable. Web applications generating pages at runtime, can significantly increase their flexibility in delivering page content meeting more demanding requirements. Data-intensive web applications are conveniently described as a hybrid between hypermedia and an information system [17]. Due to this twofold nature, data-intensive web applications require specialised techniques to integrate database technology and web engineering in the same software process. Differently from traditional software systems, web applications development introduces new challenges [17, 24]: web applications need to take into account cognitive and aesthetic aspects, producing a high level of graphical quality, to handle and link together hyper-documents, to analyse navigational information and to handle data in different formats and with different nature. Different approaches have been proposed which can be grouped according to the following categories:

- *language-centric*, page generators which merge together programming constructs and (to different extents) native support to the web domain (for instance, ASP [33], JSP [37], and PHP [1, 2]), whose chief function is to extract content from data sources at run-time, and include it into page templates; although, such systems are productive tools they seem to

fail when representing applications at a more conceptual level, since they are located at the lowest degree of abstraction; in fact, applications are represented with implementation-level languages and conceptual information, such as content, navigation and presentation, is directly encoded in the physical programs that form the application [17];

- *model-centric*, of and formalisms (e.g. Strudel [15], Araneus [34], WebML [4] and AutoWeb [18]) which tend to describe the application at (several) design stages, in order to reduce the cognitive distance between the requirement collection and the subsequent phases in the development process; they enable high-level, platform-independent descriptions of data-intensive web applications.

This paper presents a visual Domain-Specific Language (vDSL) [13, 23] for the conceptual and uniform description of the design dimensions of data-intensive web application: *a)* the data content of the site given in terms of entities and relationships as UML class diagrams [21]; *b)* the page structures specifying declaratively the way dynamic contents are composed together resulting in a view of the data; *c)* the navigation of the site depending on the way information is accessed by the page structures and providing the links in order to move across the application content. The vDSL has been given the name *Webile*.

The main contribution of this paper is to define *Webile* and especially because it is given in terms of a UML profile. The problem is approached at a conceptual level (thus it is *model-centric*) with two major goals:

- i)* To provide a conceptual modelling technique which relates together data and web: UML class diagrams offers a standard notation similar to Entity/Relationship models (E/R) [6, 39] based on structures using entities that relate to one another. E/R diagrams are a universal mean to conceptually model data and many recent proposals (as those mentioned above) for modelling data-intensive web applications are built on top of the E/R diagrams (as shown in [17]). The UML extension mechanisms allows to specialise the meta-model element, in order to model the abstractions and their relationships over a specific domain. The *Webile* approach introduces the necessary extensions to inherit the E/R diagram concepts borrowing the same ideas defined in the IBM Rational Software Data Modelling Profile (*data content*), DMP for brevity from here on, and to cover the hypermedia concepts (*page structure* and *navigation*).
- ii)* To automatically generate web applications code: in contrast with other approaches *Webile* exploits the UML meta-model architecture (see Table 2) by serialising the specifications in the XMI interchange format [22]. This representation provides interoperability and enables a XSL transformation-based automatic generation of the applications being modelled as presented in the sequel of the paper.

Summarising, *Webile* is a vDSL which extends incrementally UML DMP to the web application domains. New modelling elements are introduced which can be automatically processed in order to automatically generate web application by means of XML based techniques. The paper is organised as follows. Next section provides the technical background being assumed in the paper. Section 3 describes the *Webile* notation. Section 4 presents how the code generation process is

Database Concept	UML Structure	Data Modelling Stereotype
Database	Component	« Database »
Schema	Package	« Schema »
Table (entity)	Class	« Table »
Domain	Class	« Domain »
Relationship	Association	« Non – Identifying »
Strong Relationship	Composite Aggregate	« Identifying »
Index	Operation	« Index »
Primary Key Constraint	Operation	« PK »
Foreign Key Constraint	Operation	« FK »
Unique Constraint	Operation	« Unique »
Check Constraint	Operation	« Check »
Trigger	Operation	« Trigger »
Stored Procedure	Utility Class	« SP »

Table 1: UML data modelling profile stereotypes

defined. Section 5 discusses related research work and finally Section 6 concludes this paper and analyses future work directions.

## 2 Technical background

In this section, the preliminary notations used in this work is presented. In particular, a brief overview of the UML Data Modelling Profile extensions to a UML class diagram is introduced. A detailed and extended description of this profile can be found in [11]. This description is the starting point for the definition of *Webile*. The section covers also the XML-based Metadata Interchange (XMI) and the Meta Object Facility (MOF), respectively. These are relevant notions upon which the *Webile* profile is based together with the subsequent transformation framework.

### 2.1 The UML data modelling profile

UML Data Modelling Profiles were introduced by IBM Rational Software in the year 2000. The profile extends a UML class diagram by introducing a collection of data modelling stereotypes and provides an easy to use and understand adoption of UML for database modelling and design. A *profile* groups together a set of related UML extensions. A UML *extension* [21] modifies the original UML semantics in order to build semantically different models. Stereotypes, tagged values and constraints represent UML general extension mechanisms: *stereotypes* define a new kind of model element and are described by placing their name around angle brackets (or by using icons); *tagged values* represent UML element properties; *constraints* are restrictions on UML elements that limit the usage, or the semantics, of the elements themselves [14]. Many profiles have been proposed over the last few years, as for instance the UML profile for web applications [9], for security [27] and for CORBA. In Table 1 the organisational elements of DMP, as they are reported

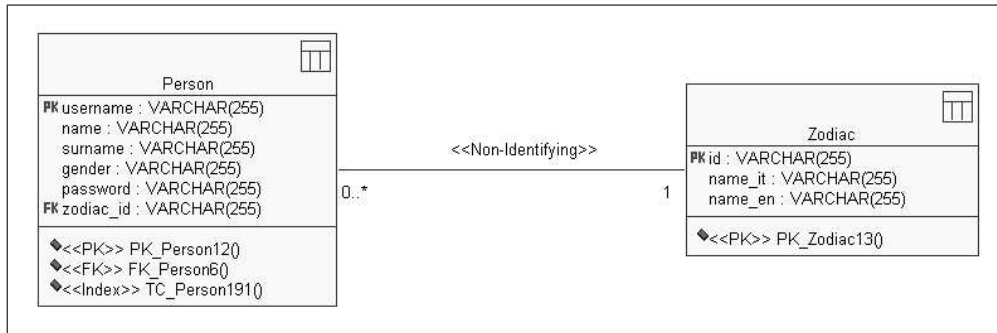


Figure 1: UML DMP Example

in [11]. The table has to be read in this way: the first column identifies the data concepts the profile wants to model; the second column identifies the UML element which has to be stereotyped; finally, the last column identifies the stereotype name associated to the related UML structure.

The database is the system for physical data storage and controlled access to stored data. It is the biggest specialised element for data modelling. The database defines the type of the database and the constraints for the data modelling like data types, stored procedure, syntax, etc. The assignment of *schemas* to the database defines the basic structure of the information storage. A *table* is the basic modelling structure of a relational database. It represents a set of records of the same structure, also called rows. Each of these records contains data. The column is the basic organisational element inside of the relational database. Every data has to be stored in a column of a row in a table. The columns add the tagged value of the data type, which has to be. There is a number of tagged values associated with columns to model the null ability and the uniqueness, for instance. Keys are also covered, in particular *primary keys* are used to uniquely identify a row in a table, whilst *foreign keys* access data in other related tables. The primary key is often content independent and automatically generated by the database to ease updates of data. Foreign keys are always derived from a relationship to other table. For easy recognition of the key columns in the table, they are tagged with either Primary Key ( $\ll PK \gg$ ) or Foreign Key ( $\ll FK \gg$ ) stereotype. A dependency of any kind between tables in a data model is called *relationship*. A relationship is a summary of a stereotyped association and a set of primary and foreign keys. Every relationship is between a parent and child table, where a parent table must have a primary key defined. The child table creates a foreign key column and a foreign key constraint to address the parent table.

Figure 1 illustrates a sample DMP class diagram. Each rectangular represents a table in DMP - the small icon on the right top side identifies its new semantics. The association between tables represents a relationship between the two tables. In the sample in Figure 1, *Person* and *Zodiac* are depicted as tables. The *Person* table attributes are *username*, *name*, *surname*, *gender*, *password* and *zodiac\_id*, where *username* is a primary key and *zodiac\_id* a foreign one. The *Zodiac* table attributes are *id*, *name\_it* and *name\_en* with *id* primary key. A 1:N association is used between the two tables specifying that each person is associated with a zodiac entry.

The profile can be used in combination with Data Modeler, an IBM Rational Software add-in [12] for the Rose CASE tool. This software assistant allows the designer to map the data model diagram to the class diagram (also known as object model) and the other way round. In general, the Data Modeler is an application which implements the DMP and performs tasks, such as some

static analysis and type-checking amongst others.

## 2.2 XMI and MOF

The UML specification provides the XML Metadata Interchange format (XMI) [22] to enable easy interchange of metadata between modelling tools and metadata repositories in distributed heterogeneous environments. XMI integrates three key industry standards: the eXtensible Markup Language (XML), a W3C standard, the UML, and the Meta Object Facility (MOF) [20], an OMG meta-modelling standard which is used to specify meta-models.

A fundamental aspect of UML is the four layered meta-modelling architecture for general purpose manipulation of metadata in distributed object repositories (see Table 2) which makes it suitable for our universal object-oriented modelling approach. Each layer is an abstraction of the underlying layer with the top layer (M3) at the highest abstraction level. The M0 layer is comprised of the information that we wish to describe (the data). This is the source code in different languages, e.g. Java, C++ or PHP as in our case. On the model layer (M1) there is the meta-data of the M0 layer, the so-called model. Object-oriented software is typically described on the M1 layer as a UML model. The meta-model on the M2 layer consists of descriptions that define the structure and semantics of meta-data (e.g. the UML model). These are the meta-models, e.g. UML 1.4, UML 1.5, and define the language respectively notation for describing different kinds of data (M1). Finally on the M3 layer there is the meta-meta-model. MOF is used to describe meta-models and define their structure and semantic. It is an object-oriented language for defining meta-data. MOF is self-describing. In other words, MOF uses its own meta-modelling constructs. XMI was partially influenced by the ideas for a tool-independent CASE data interchange format called CDIF [7], which was based on Entity-Relationship (ER) descriptions. CDIF addresses the problem of model data interchange between CASE tools. Without a standardised interchange format for integrating more than one CASE tool, proprietary import/export filters must support the exchange of model data. In addition, new interfaces have to be implemented for tool integration. XMI is supported in a wide range of industry applications. In the machine tool domain for example, STEP, a standard for the exchange of product definition/model data, will be compatible with XMI in the future. In the current version 1.5 of the UML standard it is possible to completely interchange model information. Nevertheless, it is not yet possible to interchange the graphical views of the model in terms of diagrams, which will be supported in the forthcoming UML 2.0.

Meta-level	MOF terms	Examples
M3	Meta-metamodel	MOF model
M2	Meta-metadata, Metamodel	UML Metamodel
M1	Metadata, Model	UML Models
M0	Data	Modeled Systems

Table 2: A typical OMG metamodel architecture

### 3 Webile: A UML web specific data model profile

The *Webile* approach is model-based and defines a UML profile to describe in a uniform way the aspects which characterise data-intensive web applications. Models for data content, page structure, and navigation are given by means of class diagrams. This does not necessarily imply that all the aspects are modelled at once, on the contrary the separation amongst the various level of web site design and implementation is fundamental and therefore should be strongly pursued. This results in a layered class diagram where each aspect is described in separate layers connected one with another by means of mappings. In this paper, since the focus is mainly given on the structuring mechanism given by the profile we do not present how layers are arranged and the samples will be given by means of flat diagram. Essentially, the model of data can be extended to describe also hypertext views over the database in order to describe pages. The views can be regarded basically as relations bringing together those contents stored in the database which must be published.

*Data content.* The data are modeled logically in a way which is strongly influenced by the Rational's UML Data Modelling Profile, which provides an easy to use and understand adoption of UML for the need of database modelling, and database design. The concepts of tables and relationships used in a database maps to the concepts of classes and associations in the core UML. In this setting, columns add the tagged value of the data type, which is depending on the chosen target database. *Webile* borrows the concepts of table and relationships from DMP, but it adopts more expressive data types. This represents the main difference between DMP and our approach concerning the modelling of data. Indeed, we introduced different types to better deal with those features which are inherent to the modeled domain. In particular, we intend to bind given database types with those input types which are used on the web, such as Text, Textarea, and Checkbox just to mention a few. Thus, each attribute is assigned a name and a type expression, which consists of a list of property descriptors. For instance, we may want to declare an attribute which corresponds to the name of an article as follows

*article: [Text("Article Name","Nome Articolo")][Multilanguage("en","it")][VARCHAR(30)]*

The basic idea is to have an attribute called article which is going to be mapped to a column whose SQL type is *VARCHAR(30)* similarly to Rational's DMP (the basic types in DMP are depending on the selected target database). Additionally, this type information is refined to include a further descriptor

*Multilanguage("en","it")*

to denote that this item of information must be formulated in two languages, English and Italian, respectively. This is important since many e-business have an international perspective and content must be often published in different languages. The intended semantics causes that in the table the attribute article is mapped into two columns *article\_en* and *article\_it*, respectively. Finally, the term

*Text("Article Name","Nome Articolo")*

is referred to an abstract form element. In particular, it specifies that when the information has to be eventually entered in a form, the *Text* HTML input type must be used (in case the chosen media is a HTML browser) and the corresponding label denoting this entry must be *Article Name*

or *Nome Articolo*, according to the language chosen by the user. These descriptions are used to automatically generate proper client-side and server-side code, respectively. The general form of an attribute declaration is the following

*attribute: [input\_type("Label1",...)] [Multilanguage("lan1",...)] [SQL\_type]*

and can be explained with the UML class diagram in Fig. 2 part of the Webile profile definition. Such class diagram restricts the semantics of the UML primitive type *string* by refining it into a composition, called *WDMtype*, of both *Webile implementation* and *presentation* types. The former denotes the internationalisation and the SQL type which are usually given once certain implementation decisions are taken; the latter is the kind of widget which should be used in entering the data in a form whilst manipulating it. The type expressions are not usually given at once, rather they should be regarded as the outcome of a refinement process which may take several steps.

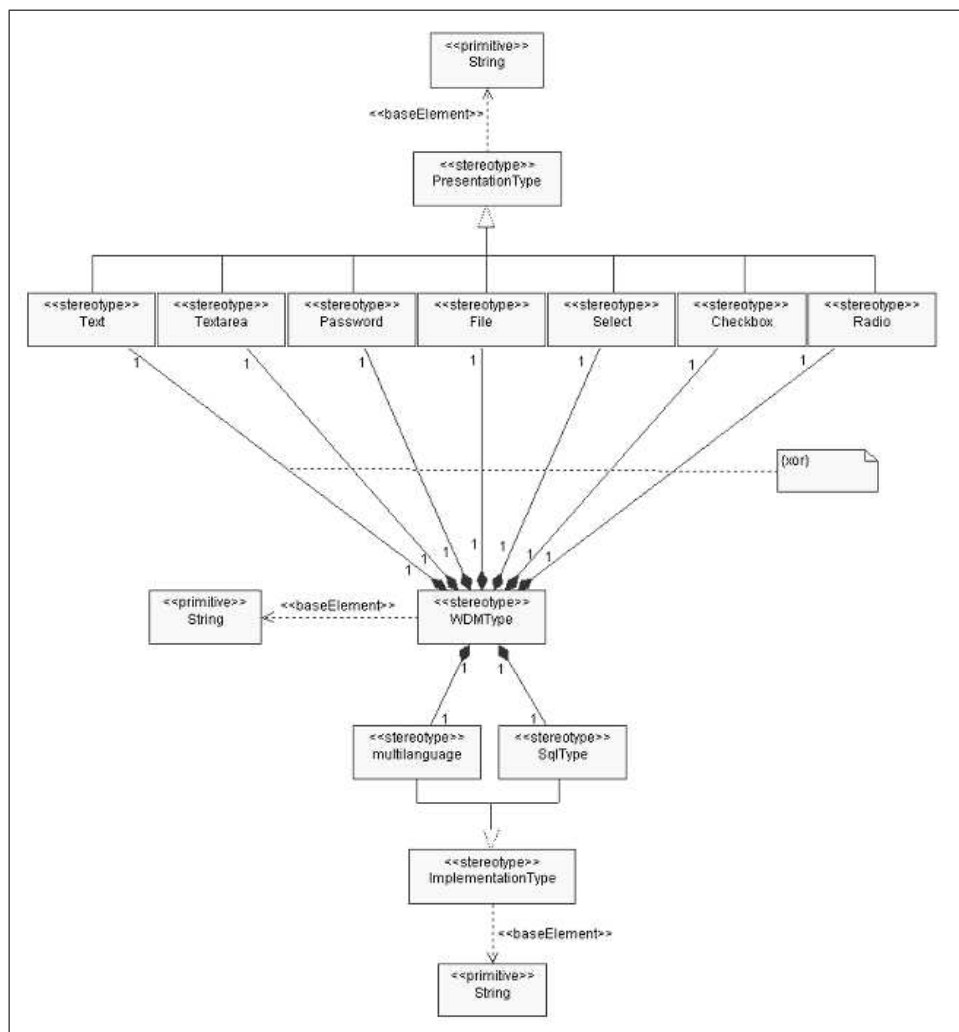


Figure 2: Webile data modelling types

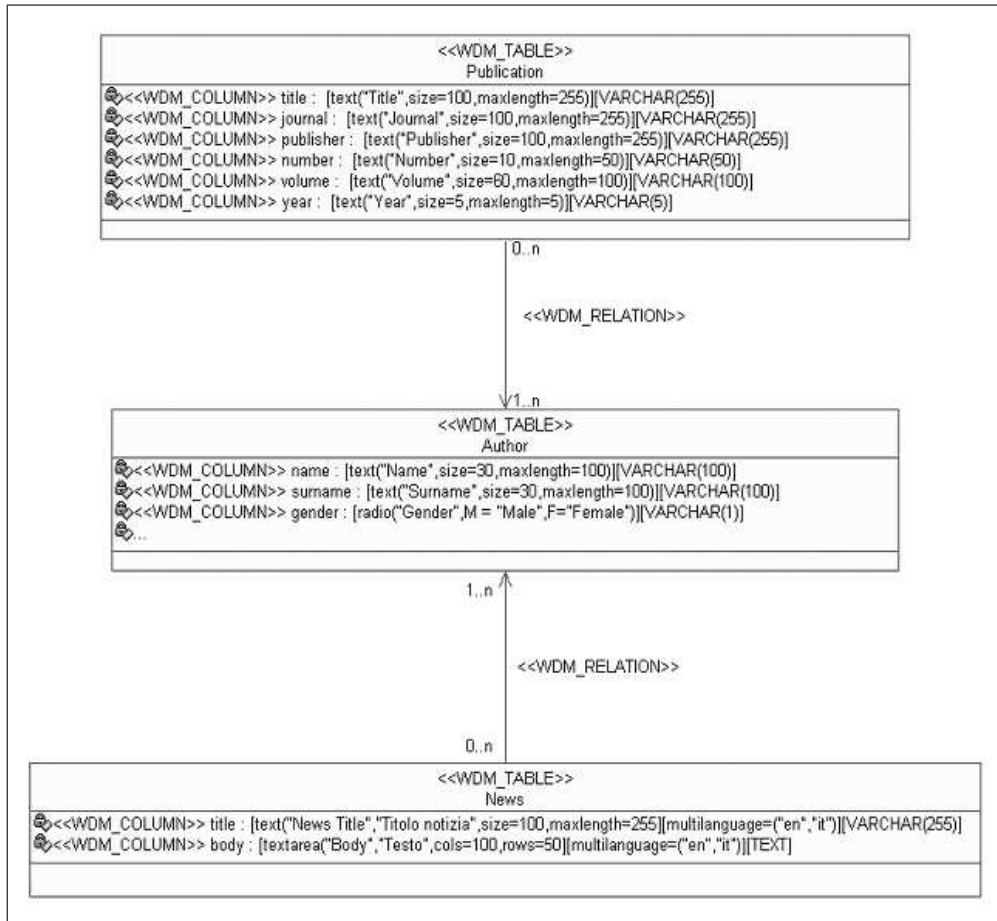


Figure 3: A Sample Data Model

Figure 3 contains a simple conceptual model depicting a small database to store data about authors, their publications and news. In this model the tables *Publication* and *News* are depending on *Author* since they have references to it as shown by the relations which are also attributed with the cardinality. This *Webile* fragment is enough to automatically generate the DDL and the code fragment necessary to perform the data entry management.

*Page structure.* A data-intensive Website can be regarded as a hypertext view of the database where all its content is stored. A page structure is an intentional description of a set of web pages with common features, such as home pages of people whose data are stored in a table. A page structure is modeled by the stereotype  $\ll PageStructure \gg$  which is associated with tables containing the data sources forming the content being described. The page structures are associated with tables by means of the stereotyped association  $\ll DataSource \gg$ . These associations are tagged with a collection of tagged values, amongst them *Columns* and *Cardinality*. The former describes the list of columns whose values are part of the page structure, whilst the latter describes the cardinality of the items to be included in the content, i.e. whether the content is publishing one item or multiple items. In Fig. 4, the example illustrated in Fig. 3 is refined by augmenting it with the page structure *AuthorList*, *AuthorHome*, and *NewsItem*. The idea is to model web pages which

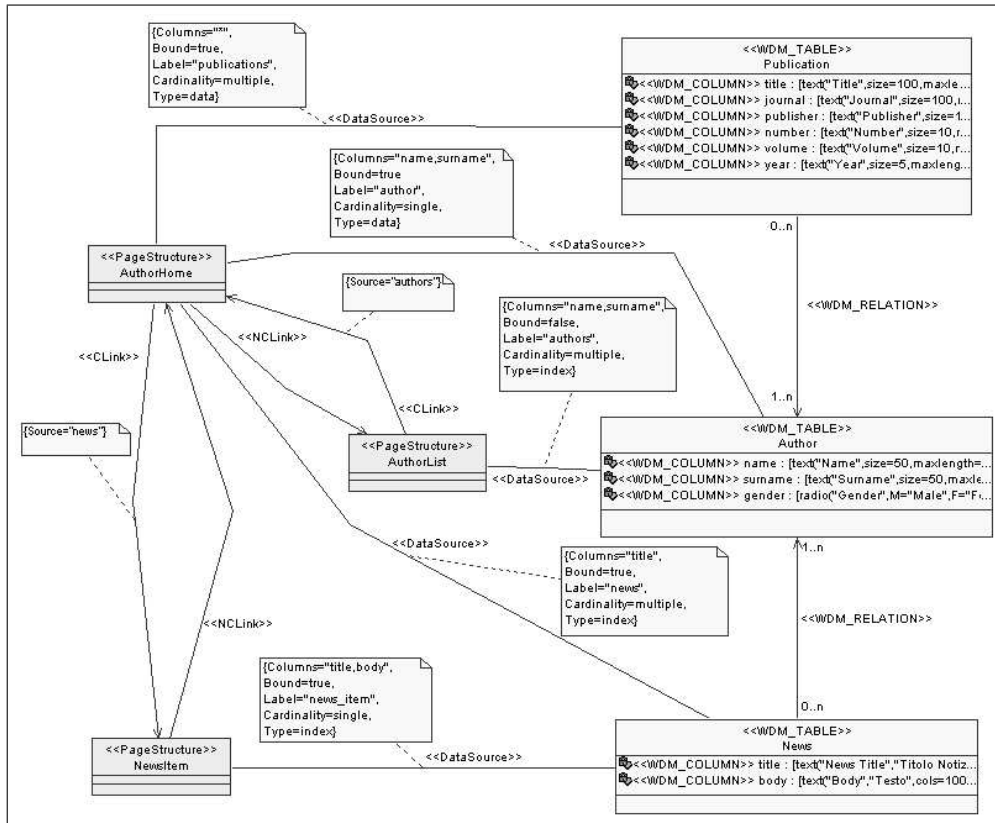


Figure 4: A Weble Model

publishes the list of all authors stored in the table *Author*, the home page of a single author, and one single news, respectively, by means of the stereotyped classes `<<PageStructure>>`. Which data and how they should be published in these contents are specified by the tagged values given in the stereotyped association `<<DataSource>>`. The *AuthorHome* page structure publishes name and surname of the author, all the data about all her/his publications, and the title of news entered by the author. This is specified in the associations with *Author*, *Publication*, and *News*, respectively, by the following tagged values

1. *AuthorList/Author* association tagged values:  
*Columns="name, surname", Bound=false, Label="authors", Cardinality=multiple, Type=index*
2. *AuthorHome/Author* association tagged values:  
*Columns="name, surname", Bound=true, Label="author", Cardinality=single, Type=data*
3. *AuthorHome/Publication* association tagged values:  
*Columns="\*", Bound=true, Label="publications", Cardinality=multiple, Type=data*
4. *AuthorHome/News* association tagged values:  
*Columns="title", Bound=true, Label="news", Cardinality=multiple, Type=index*

as reported in the figure. Each tagged value describe a specific aspect of the stereotyped association  $\ll DataSource \gg$ , most of the tagged values has default values, but for the sake of clarity in this example we have listed all of them even if not necessary.

- *Columns* describes the list of columns whose values should be considered for the target class of the association, i.e. the page structure being modeled; similarly to SQL it is possible to use the wild card for denoting all columns as in tagged values (3) in the previous list.
- The *Bound* value affects the way the data to be published by the target page structure are filtered in current navigational context; this tagged value is meaningful in situations when a page structure is  $\ll DataSource \gg$  associated with two different tables which are in turn associated by means of a relationship path, as for instance for page structure *AuthorHome* associated with *Author* and *Publication*, with *Author* and *Publication* associated by means of a 1:N relationship; by specifying the *Bound* equal to *true* on the  $\ll DataSource \gg$  association between *AuthorHome* and *Publication*, we denote the fact that the data from *Publication* are selected by means of a join between the tables with join condition unifying over the 1:N relationship between *Author* and *Publication* tables as specified in (3).
- The *Label* tagged value is necessary to name *DataSource* association in case ambiguity arises due to multiple associations between page structures and tables; this is important in combination with different *DataSource* association involving the same classes.
- *Cardinality* is used to specify whether a *single* or *multiple* data should be published in a page structure; for instance, in the association denoted by the tagged values (2) the *Cardinality* is *single*, in fact the page *AuthorHome* is intended to contain all the data corresponding to a given specific author, whereas the association denoted by (1) determines the fact that the page *AuthorList* contains the data of all the authors as *Cardinality* is *multiple*.
- The *Type* tagged value indicates whether the data being contained in the corresponding page structure should give place to plain content or to navigational content, i.e. to content with links to other page structure (within the same model) reachable by means of the propagation of the primary keys due to the *Bound* feature; for instance, the page structure *AuthorList* contains an *index* which allows to link to pages of single authors as stated in (1) or for instance as in (4) where from an *AuthorHome* there is a news index which allows one to access each of them.

*Navigation.* Similarly to the page structures, navigation is modeled by means of stereotyped associations  $\ll CLink \gg$  and  $\ll NCLink \gg$  denoting contextual and non-contextual link, respectively. Fig. 4 reports a contextual link from *AuthorList* to *AuthorHome* and from *AuthorHome* to *NewsItem*. Contextual links are associations which propagate parameters from the source page structure to the target one in order to determine how data are filtered, as in the link between *AuthorList* and *AuthorHome*, for instance. Non-contextual links are much simpler since they connect page structures which are not semantically correlated.

The contextual links and the bound mechanism define completely the data filtering. As a concluding remark to this section, one should mention that, in contrast with other approaches (e.g. [34]) we do not deal with presentation issues but regard the contents as structured data. On a

physical level it may lead to different implementation styles. Indeed, in a preliminary version of our implementation we preferred to build the presentation of the contents by means of a templating engine. Alternatively, we could use an XML/XSL approach.

## 4 Code generation: an overview

As mentioned in the previous sections, the OMG Metadata Architecture (see Table 2) provides a classification of UML concepts and notions, including light/heavyweight extension mechanisms, models and artifacts. The lightweight extension mechanisms given in the M2 meta-level allowed us to define the *Webile* profile. *Webile* class diagrams specify web applications and lie on the M1 meta-layer. Once the implementation is obtained, it refers to the M0 meta-layer.

In this section, we outline the mapping of M1 objects (*Webile* models) into M0 objects (implementations) by means of a XSLT-based generation framework as depicted in Fig. 5. The manipulation of the models in the M1 meta-level is based on the XMI format which provides an interoperability platform for accessing UML models. *Webile* has been defined in order to have models, which are rich enough to generate the following tools

- i) *data-manipulation*, the type expressions in the table classes are used to generate meta-form descriptions which are used to obtain, for instance, HTML-based data-entry pages. Such meta-form descriptions are device-independent descriptions which are updated according to the kind of web agent being used by the user. In the generation process, entities and relationships have a fundamental role, from each entity class a collection of data management tools is obtained, whereas a 1:N relationship gives place to a specific widget in the meta-form which allows the user to select an item belonging to the entity with cardinality 1 in the relationship. Such a widget is rendered as a combo box or a list of radio box, for instance, once the web agent is determined.
- ii) *front-office*, the whole diagram is used to build a collection of pages linked one with another according to the given specification. Each page structure gives place to a server-side program which according to the  $\ll DataSource \gg$  associations retrieves the data from the corresponding tables. The queries which are generated are also compliant to the filtering policies defined by the association's tagged values, such *Bound* and *Cardinality*.

These tools are generated by means of XSL transformations, whose rules localise the abstractions in the XMI representation of the model and transform them into appropriate documents. Amongst such documents there are those in i) and ii), but there are also auxiliary tools which include the DDL and the DML written in SQL (via an abstraction layer), and a library containing the data-retrieval functions which consist of SQL code embedded in some host routines. Rendering issues have been implicitly treated in the generated system by means of a templating engine. Each page structure through a naming policy is associated with a HTML template which presents formal parameters which are updated with the data obtained by the  $\ll DataSource \gg$  associations.

The current prototypical implementation is totally based on XSL transformations and XML-based techniques, whilst the generated system is a collection of routine written in PHP, SQL, and JavaScript (only for form validation purposes). The meta-form notation which has been used is a

server-side dialect of a small subset of Xforms. The Metadata Architecture being used is the one proposed in Table 3: OMG Metadata Architecture.

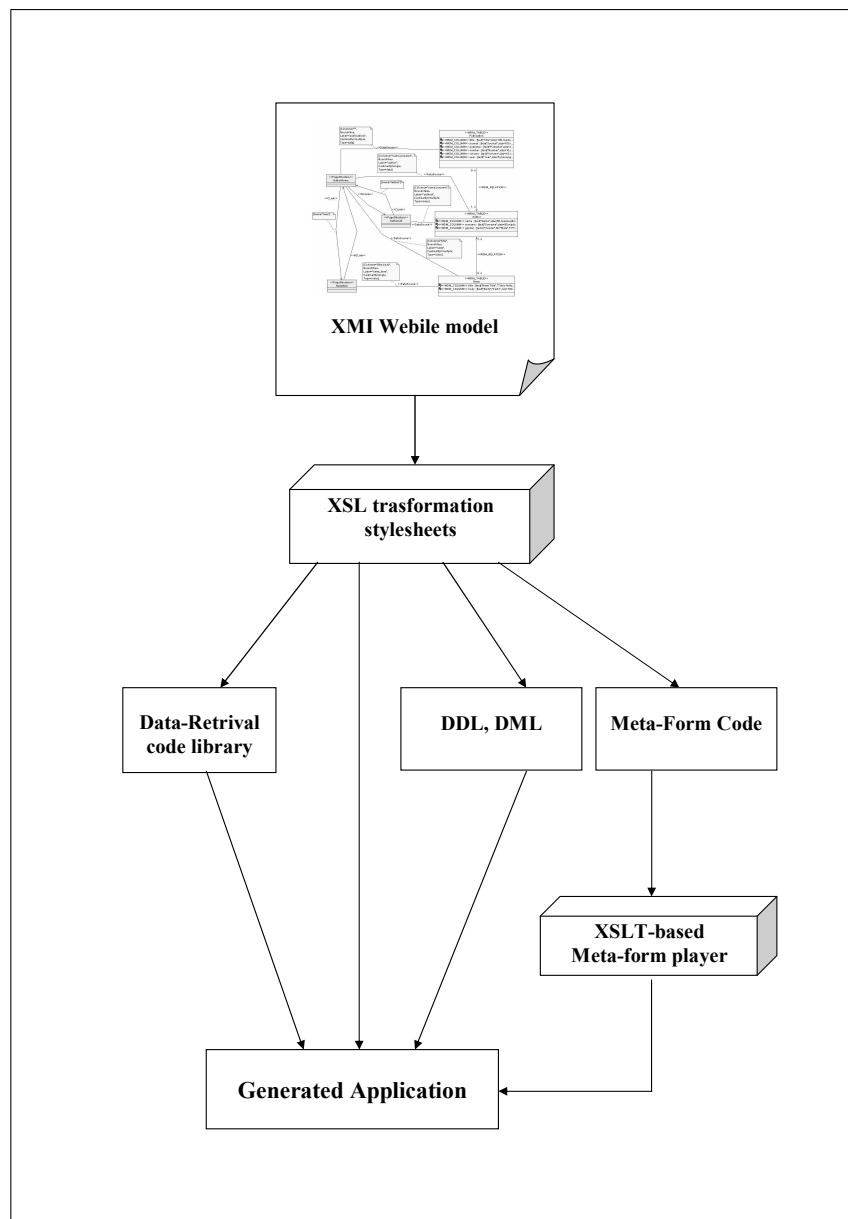


Figure 5: Webile generation framework

## 5 Related Work

Related work may span in different research areas. In this section we list and analyse those research areas which can be directly related to our work. We describe how UML has been used for data modelling purposes, how UML has been used in order to model web applications, and finally,

current work on model-based code generation for the web. For space limitation, work on model-based design of web applications will be only sketched.

*UML and Data Modelling.* UML DMP [19], already outlined during this paper, is the Rational's profile to data modelling with UML. This profile is not yet standard OMG but fully integrated into the Rational's tool, Rational Rose, with its data modeler add-in. The analysis on the integration of UML and database modelling has been already advocated by a number of works ([10, 11]). Another UML profile for data modelling has been proposed by the Agile Data community. Three data models are proposed: Logical, Physical, and Conceptual. Their notation focuses on the physical modelling of a relational database, although it does cover other aspects of data modelling as needed. Differently from them, we focus on the conceptual, entity-relationship, data level, our approach is specific to web applications and we use the model to generate data-intensive web pages. On the other side, their profile includes physical and logical modelling we do not treat at all.

Naiburg and Maksimchuk, in their book on UML for Database Design [36], show how the UML can serve as a unifying framework that facilitates the integration of database models with the rest of a system design. They propose a software process which, starting from a database conceptual description, refines this information into a logical level until a physical description is obtained. They make use of the UML DMP (as it is) at the physical design level.

Muller [35] proposes a step by step design process, to design and build databases using an OO model. The process goes from requirements analysis to schema generations. UML class diagrams are translated into relational, object-relational, and object-oriented schemas for all major DBMS products. Differently from our approach, both Naiburg, Maksimchuk and Muller propose a software process (we have in our future work directions), not specifically oriented to web applications neither to code generation.

*UML and web applications modelling.* There are different approaches proposed to model web applications using UML. The one we consider more important are those proposed by Conallen [8, 9] (the Rational's official solution) and N. Koch and R. Hennicker [24, 25]. The Conallen's approach is based on the Rational Unified Process (RUP) [30], thus, it is Use-Case driven, Architecture-centric, iterative and incremental. By adapting the RUP process to web applications, requirements are modeled using Use Case diagrams and scenarios, design identifies real objects (links, frames, forms, ASP object and other web-related elements) and the web architecture (through class, component and sequence diagrams), implementation creates client pages, server pages and business objects.

Koch and Hennicker propose an approach to model hypermedia applications using UML. They structure the "design" of web applications into three different aspects: content, navigational structure and presentation. Since these aspects are related together, they propose to model each one using a different UML model and to relate them together using mapping rules.

Conallen's approach focuses on the web system architecture. UML class diagrams extend the traditional concept of class, introducing client side pages, server side pages, forms, frames, links and business logics objects. Different aspects of the system (presentation, business and backend) may be described in the same diagram. However, data cannot be conceptually modeled and data-driven web pages cannot be generated. Koch and Hennicker use UML Class and Sequence diagrams to describe and model navigational information between web pages. However, business logic and data are not explicitly analysed.

*Model-Based Code Generation for the web.* The most important commercial proposal in this

Meta-level	MOF terms	Instances
M3	Meta-metamodel	MOF model
M2	Meta-metadata, Metamodel	UML 1.5, UML web Specific Data Modelling Profile
M1	Metadata, Model	UML Data Models
M0	Data	Generated Systems (SQL, HTML/JavaScript, PHP), Application Database, etc.

Table 3: OMG Metadata Architecture

category is Oracle Designer. It represents a database-centric approach which relies on high-level models for the development and generation of web applications. The design process is based on augmented ER diagrams. The Designer is integrated with software generators, which, starting from database design diagrams, modules and links between them and user parameters, generate web pages.

A. Kraus and N. Koch [28], show how the UML design models produced in [25] can be automatically mapped to XML documents. XML documents for the conceptual model are then automatically mapped to conceptual DOM objects and logical presentation objects (representing the user interface) are transformed to physical presentation objects (e.g., HTML or WAP pages) through XSLT transformations.

Araneus [34], AutoWeb [18], WebML [4] represent the more interesting model-based approaches for the web. Due to space limitation, we analyse only Araneus, since it appears one of the most representative approaches and presents many commonalities with the other ones. For a description of web site development tools and projects, you may refer to [17, 34]. A complete overview on web applications methodologies is presented in [38]. Araneus is a model-based environment for managing web content in an integrated way. High-level models are used to design and develop each aspect of a web site. These models are integrated in a software process which, in five steps, analyses the system requirements, develops the site contents, designs both databases, presentation and site structure and finally implements it. Our approach shares with Araneus the idea to realise a model-based web site development. In this paper we have shown how to model data using UML notations. Differently from Araneus, our goal is to use a unique, UML-based notation, to model each aspect of a web application. Araneus, instead, uses three different models (entity-relationship and relational tables for data structure modelling, the AMD-d model for navigation, and ADM-d scheme attributes for presentation) and the Homer algebra to relate them together.

## 6 Conclusions and Future Work

This paper introduces *Webile*, a UML profile for modelling data-intensive web applications. The work starts from the data modelling capabilities of the IBM Rational Software's Data Modelling Profile and extends it in order to capture abstractions and features which are typical of the hyper-media domain. A *Webile* model consists of the specification of the aspects of a web application, i.e. the data content, the page structures, and the navigation. These notions are captured by stereotyped

UML elements and define most of the aspects of a data-intensive Website.

Once a web application is described, its model can be used to automatically generate the application by using XSL transformation stylesheets. The generated application consists of a collection of programs which allow the data management of the application (typically implement the functionality of the back office) and the server-side programs which will implement the system which has been modeled. The main achievement of the work is the uniform formulation of all the aspects of a web applications, since the authors believe that the adoptions of different formalisms for different aspects (as in Araneus and WebML) is a major drawback. If on one hand, the separation of concerns is important, on the other hand it is not necessary to use different formalisms for different concern domains. UML not only provides with uniformity and flexibility, but also with a set of tools and formalisms which are standard, and complaint to open standards and formats, such as OMG Metadata Architecture and XML-based Metadata Interchange.

Our short-term effort will be devoted to the completion of the implementation of the *Webile* system. The system is still prototypical and many features require to be undertaken, although the current results are encouraging. Convenient mechanisms for user authentication are at the moment under investigation and especially new modelling elements will be soon included in the model in order to capture the notion of authenticated service, i.e. a page structure which will be available only after authentication. Formal semantics for *Webile* is a sound idea and will provide us with better understanding. Especially, we intend to investigate on the definition of a calculus able to perform some static analysis of the class diagram to prevent erroneous scenarios. In fact, wrong tagged values, such as the cardinality of the associations, for instance, may lead to wrong configurations which may be detected earlier in the design process. Once this is defined, it should be integrated in some existing UML-based tools (e.g., Rational Rose or Argo/UML). What we envision is a tool-assisted framework where UML diagrams are drawn, specialised profiles can be applied in order to define data, applications or presentation elements for a web application. A well defined software development process is also in need. With the proposed approach we do not claim that all the required information (for web pages generation, for example) are known a priori or has to be provided during the initial design steps. We plan a more incremental approach where the main entities and relationships are modeled using a light *Webile* initial model and successively improved (during architectural analysis and system design) with information regarding the presentation and navigation.

## 7 Acknowledgements

This work is partially supported by the Italian M.I.U.R. project SAHARA.

## References

- [1] K. Bhatnagar, M. Gupta, Niit Corporation, Niit Contributor, and A.D. Wilfred. *PHP Professional Projects*. Premier Press, 2002.
- [2] J. Blank, W. Choi, A. Kent, G. Prasad, and C. Ullman. *Beginning PHP4*. pub-WROX, 2000.
- [3] S.J. Cannan and G.A.M. Otten. *SQL - The Standard Handbook*. McGraw-Hill, 1992.

- [4] S. Ceri, P. Fraternali, and A. Bongio. Web Modeling Language (WebML): a modeling language for designing Web sites. *Computer Networks (Amsterdam, Netherlands: 1999)*, 33(1–6):137–157, june 2000.
- [5] S. Ceri, P. Fraternali, A. Bongio, M. Brambilla, S. Comai, and M. Matera. Designing Data-Intensive web Applications. *The Morgan-Kaufmann Series in Data Management Systems, Jim Gray, Series Editor*, 2002.
- [6] P.P. Chen. The Entity-Relationship Model: Toward a Unifying View of Data. *ACM Trans. on Data Base Systems*, 1(1):9–36, 1976.
- [7] EIA / CDIF Technical Committee. CDIF / CASE Data Interchange Format, 1994. EIA Interim Std. EIS / IS-106-112.
- [8] J. Conallen. Modelling web Application Architectures with UML. *Comm. ACM*, 42(10):63–70, 1999.
- [9] J. Conallen. *Building web Applications with UML*. The Addison-Wesley Object Technology Series, 2000.
- [10] Rational Software Corporation. Mapping Object to Data Models with the UML, 2000. Rational Software White Paper, TP-185 3/00.
- [11] Rational Software Corporation. The UML and Data Modelling, 2000. Rational Software White Paper.
- [12] Rational Software Corporation. Using Rose Data Modeler, 2000. Rational Rose.
- [13] A. van Deursen, P. Klint, and J. Visser. *Domain-Specific Languages*. Marcel Dekker, 2002.
- [14] H.-E. Eriksson and M. Penker. *UML Toolkit*. John Wiley & Sons, 1997.
- [15] M. Fernandez, D. Florescu, A. Levy, and D. Suciu. Declarative Specification of Web Sites with Strudel. *VLDB Journal: Very Large Data Bases*, 9(1):38–55, 2000.
- [16] D. Florescu, A. Levy, and A. Mendelzon. Database Techniques for the World-Wide web: A Survey. *IACM SIGMOD Record*, 27(3), 1998.
- [17] P. Fraternali. Tools and approaches for developing data-intensive web applications: A survey. *ACM Computing Surveys (CSUR)*, 31(3):227–263, 1999.
- [18] P. Fraternali and P. Paolini. Model-driven development of Web applications: the AutoWeb system. *ACM Transactions on Information Systems*, 18(4):323–382, 2000.
- [19] D. Gornik. UML Data Modelling Profile, 2002. Rational Software White Paper, TP162.
- [20] Object Management Group. OMG/Meta Object Facility(MOF) V1.4, 2001.
- [21] Object Management Group. OMG/Unified Modelling Language(UML) V1.4, 2001.

- [22] Object Management Group. OMG/XML Metadata Inter-change(XMI) V1.2, 2002.
- [23] J. Heering and M. Mernik. Domain-specific languages for software engineering. In *Proc. Minitrack part of the Software Technology Track of HICSS-35*, 2002.
- [24] R. Hennicker and N. Koch. A UML-based methodology for hypermedia design. In Andy Evans, Stuart Kent, and Bran Selic, editors, *UML 2000 - The Unified Modeling Language. Advancing the Standard. Third International Conference, York, UK, October 2000, Proceedings*, volume 1939 of LNCS, pages 410–424. Springer, 2000.
- [25] R. Hennicker and N. Koch. Systematic design of web applications with UML. In Keng Siau and Terry Halpin, editors, *Unified Modeling Language: Systems Analysis, Design and Development Issues*, chapter 1, pages 1–20. Idea Publishing Group, 2001.
- [26] Ardent Software Inc. ODMG OQL User Manual. Release 5.0, 1998.
- [27] J. Jrjens. UMLsec: Extending UML for secure systems development. In Jean-Marc Jézéquel, Heinrich Hussmann, and Stephen Cook, editors, *UML 2002 - The Unified Modeling Language. Model Engineering, Languages, Concepts, and Tools. 5th International Conference, Dresden, Germany, September/October 2002, Proceedings*, volume 2460 of LNCS, pages 412–425. Springer, 2002.
- [28] A. Kraus and N. Koch. Generation of web applications from uml models using an xml publishing framework. In H. Ehrig, B. Krmer, and A. Ertas, editors, *6th World Conference on Integrated Design and Process Technology (IDPT)*, volume 1, 2002.
- [29] P. Kruchten. Architectural Blueprints - The "4+1" View Model of Software Architecture. *IEEE Software*, 12(6):42–50, 1995.
- [30] P. Kruchten. *The Rational Unified Process: An Introduction*. Addison-Wesley Pub. Co., second edition, 2000.
- [31] P.M. Lewis, A. Bernstein, and M. Kifer. *Databases and Transaction Processing: An Application-Oriented Approach*. Addison-Wesley, 2002.
- [32] D. Lowe and W. Hall. *Hypermedia & the web: An Engineering Approach*. John Wiley & Sons, 1999.
- [33] M. Macdonald and R. Standefer. *ASP.NET: The Complete Reference*. McGraw-Hill Osborne Media, 2002.
- [34] P. Merialdo, P. Atzeni, and G. Mecca. Design and development of data-intensive web sites: The Araneus approach. *ACM Transactions on Internet Technology (TOIT)*, 3(1):49–92, 2003.
- [35] R.J. Muller. *Database Design for Smarties: Using UML for Data Modelling*. Morgan Kaufmann, 1999.
- [36] E.J. Naiburg and R.A. Maksimchuk. *UML for Database Design*. Addison-Wesley Object Technology Series, 2001.

- [37] H. Phil and H. Philip. *JSP 2.0: The Complete Reference*. McGraw-Hill Osborne Media, second edition, 2002.
- [38] D. Schwabe. A conference review system. In *Proc. 1st Workshop on web-oriented Software Technology*, 2001.
- [39] B. Thalheim. *Entity-Relationship Modeling: Foundations of Database Technology*. Springer-Verlag New York, Inc., 2000.