Journal of Web Engineering, Vol. 0, No. 0 (2003) 000–000

Adding Client-Side Adaptation to the Conceptual Design of e-Learning Web Applications

Stefano Ceri
Dipartimento di Elettronica e Informazione, Politecnico di Milano
Via Ponzio, 34/5, 20133 Milano, Italy
ceri@elet.polimi.it

Peter Dolog
L3S Research Center, University of Hannover
Expo Plaza 1, 30539 Hannover, Germany
dolog@l3s.de

Maristella Matera
Dipartimento di Elettronica e Informazione, Politecnico di Milano
Via Ponzio, 34/5, 20133 Milano, Italy
matera@elet.polimi.it

Wolfgang Nejdl
L3S Research Center, University of Hannover
Expo Plaza 1, 30539 Hannover, Germany
nejdl@l3s.de

Received (received date)
Revised (revised date)

In this paper, we integrate WebML, a high-level model and technology for building server-side Web applications, with UML-Guide, a UML-based system that generates client-side guides for the adaptation of Web applications. The combination of the two systems is shown at work on an e-Learning scenario: WebML is the basis of the specification of a generic e-Learning system, collecting a large number of learning objects, while UML-Guide is used for building company-specific e-Learning curricula. The resulting system can be considered an "adaptive hypermedia generator" in full strength, whose potential expressive power goes beyond the experiments reported in this paper.

Keywords: Personalization, UML, WebML Modeling, Web Engineering.

Communicated by: to be filled by the Editorial

1 Introduction

In recent years, the control of Web applications has moved from the client to the server side, leading to more economical, structured, and well engineered solutions. In particular, the model-driven approach, as advocated in [1, 2, 3, 4], has proved very effective in extending the classical methods and best practices of Software Engineering to the Web. Design methods now concentrate on content, navigation, and presentation design, which are orthogonally developed by means of specialized abstractions and techniques.

While server-side solutions are dominant, yet bringing some intelligence to the client may be highly beneficial in some cases [5, 6]. Client-side solutions can be more dynamic, more adaptive, and protective for sensitive user data. They may be very effective for "remembering" the local context or being aware of the local peculiarities of the interaction. Also, a clear separation of concerns between the client and the server may lead to interesting business opportunities and models.

This paper explores the combination of two existing approaches to the engineering of Web applications. We use the WebML method [1] and its development support environment [7] for generating the application server-side "backbone". We then integrate such a backbone with UML-Guide [8], a client-side personalization engine that dynamically generates additional interfaces and user guides for personalizing the application's fruition, by managing user profiles and context-sensitive data at client side.

The proposed approach capitalizes on the use of two systems that both start from high-level abstractions, and are both capable of automatic deployment of the implementations:

- The WebML method is based on the use of high-level concepts, such as the notions of entity and relationship to denote content, and of page, unit, and link to denote hypertexts. These abstractions are automatically turned into implementation artifacts by means of WebRatio, a tool for the automatic deployment of Web applications [1].

- UML-Guide is based on the use of UML state diagrams, whose nodes and arcs—representing states and transitions—are turned into XMI specifications. A client-side translator, written in XSL, turns such specifications into a user interface facilitating the adaptive use of the application [8].

Coupling WebML and UML-Guide yields the following advantages:

- The use of high-level WebML abstractions in the context of UML-Guide enables the specification of a powerful client-side personalization engine. The resulting application generator can be considered an "adaptive hypermedia generator" in full strength, whose potential expressive power goes well beyond the experiment reported in this paper.

- The tools prove to be highly complementary and easily integrated, as it is sufficient to reuse concepts of WebML inside UML-Guide to provide concept interoperability, and the URL generation technique of the WebML runtime inside the UML-Guide XSL code to provide systems interoperability.

- The use of the UML-driven methods in conjunction with WebML is by itself a very interesting direction of research, aiming at the integration of UML, the most consolidated specification model (and related technology), with WebML as a representative case of new, hypertext-specific models and techniques.

1.1 Driving Scenario

In order to exemplify the integration of the two methods, we refer to an e-Learning scenario, in which a courseware company develops and distributes a vertical application for e-Learning, running on the company's server, specified and developed through WebML.[a] The vertical incorporates learning objects in the format of lessons, exercises, tests, definitions and examples for computer science, arranged

[a] This scenario is suggested by the ProLearn Network of Excellence, whose main focus is the enhancement of professional e-Learning methods and technology; see `http://www.prolearn-project.org`.

according to the ACM categories,[b] and learning paths with checkpoints for the learner. Thus, such a vertical has learning objects as content, and navigation mechanisms, such as guided tours or indexed accesses to pages based on broad categories, enabling a generic user to access such a content though predefined navigation paths.

The vertical is used by Small-Medium Enterprises (SMEs) wishing to build personalized e-Learning curricula, to be used by their employees for focused training activities. We assume that each SME has a clear instruction goal (for example, teaching its employees how to integrate Java programming into Oracle 9i), and that it can use UML-Guide to specify it in UML; we assume that UML state diagrams, together with a vocabulary listing all the learning objects available in the vertical, may be an easy-to-use interface for the SME designer. UML-Guide specifications select the concepts to be covered in the learning paths, as well as the workflow driving the student in the learning process. We also assume that each SME has a clear view of its employees' competencies, and thus is able to constrain possibilities in the learning paths by adaptation rules based on such competencies. These rules enable adaptive content selection from the WebML vertical and also enable to adaptively indicate, show, and hide links in the learning path, and adaptively customize their targets.

1.2 Paper Organization

The paper is organized as follows. Section 2 introduces the WebML component, by providing an overview of the WebML method and the WebML-based specification of the vertical e-Learning application. Section 3 introduces the UML-Guide method, and the specification of the client-side personalization for the vertical e-Learning. Section 4 illustrates the integration of the two methods by means of an architecture where the application server-side code is generated through WebML, while personalization with respect to specific learning goals is managed at client-side through UML-Guide. Section 5 then describes the user interface generated for the integrated application. Finally, Section 6 gives an overview of the processes required for designing and customizing e-Learning applications on the courseware and SME sides. Sections 7 and 8 illustrate some related work and draw our conclusions and future work.

2 WebML Component

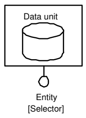
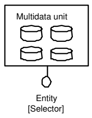
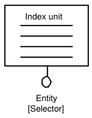
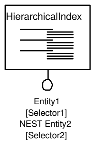
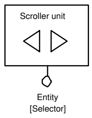
2.1 WebML Component Overview

WebML is a visual language for specifying the content structure of a Web application and the organization and presentation of contents in one or more hypertexts [1]. The design process based on WebML starts with the specification of a data schema, expressing the organization of contents by means of the Entity-Relationship primitives. The *WebML Hypertext Model* allows then describing how contents, previously specified in the data schema, are published into the application hypertext.

The overall structure of the hypertext is defined in terms of *site views*, *areas*, *pages* and *content units*. A *site view* is a hypertext, designed to address a specific set of requirements. It is composed of *areas*, which are the main sections of the hypertext and comprise recursively other sub-areas or pages. *Pages* are the actual containers of information delivered to the user; they are made of *content units* that are elementary pieces of information, extracted from the data sources by means of queries and published within pages. In particular, as described in Table 1, content units denote alternative ways for displaying one or more entity instances.

[b] See `http://www.acm.org/class/1998/`

Table 1. Some basic WebML content units. The whole set of units is described in [1].

Unit name	**Visual Notation**	**Description**
Data unit		It displays a set of attributes for a single entity instance.
Multidata unit		It displays a set of instances for a given entity.
Index unit		It displays a list of properties, also called *descriptive keys*, of a given set of entity instances.
Hierarchical Index unit		A variant of the index unit, which displays list of properties of instances selected from multiple entities, nested in a multi-level tree.
Scroller unit		It represents a scrolling mechanism, based on a block factor, for the elements in a set of instances.

Their specification requires the definition of a *source* (the name of the entity from which the unit's content is extracted) and a *selector* (a condition, used for retrieving the actual objects of the source entity that contribute to the unit's content).

Within site views, links interconnect content units and pages in a variety of configurations yielding to composite navigation mechanisms. Besides representing user navigation, links between units also specify the transportation of some information (called *context*) that the destination unit uses for selecting the data instances to be displayed.

WebML-based development is supported by a CASE tool [7], which offers a visual environment for drawing the WebML conceptual schemas, and then supports the automatic generation of server-side code. The generated applications run in a standard runtime framework on top of Java 2 application servers, and have a flexible, service-based architecture allowing components customization.

2.2 WebML Specification for the Vertical e-Learning

The data schema of the vertical e-Learning application is centered on the concept of Learning Object (LO). As reported in Figure 1, the `LO` entity represents descriptions of learning objects, by means of attributes inspired by the LOM standard.[c] Among them, the attribute `type` expresses the different types of LOs (e.g., lectures, lecture modules, definitions, exercises, tests) published by the vertical application. Each LO has associations with other LOs: for example, a lecture module can be associated with some related definitions, exercises, examples, or tests. The entity `Content` then represents the contents (texts, images, files) LOs consist of. In order to facilitate LO access, the schema also includes

[c] `http://ltsc.ieee.org/`

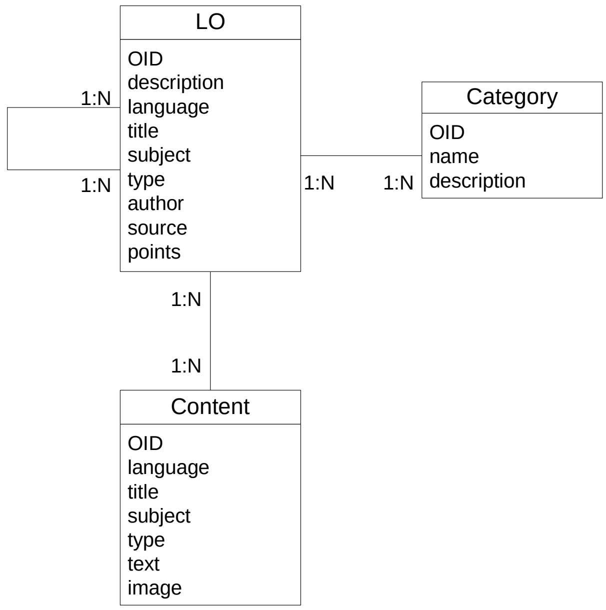


Fig. 1. WebML Data schema for the vertical e-learning application.

the entity `Category`: it stores the ACM categories that classify the LOs published by the e-Learning application.

Figure 2 reports a simplified excerpt of the WebML hypertext schema defined for the vertical e-Learning application; it refers to pages for selecting a lecture module, and accessing its contents as well as associated definitions, exercises, tests and examples. The lecture module selection is operated by means of a navigation chain, in which users progressively select a subject category (`Categories` page), then a course referring to the selected category (`Courses` page), then a lecture (`CourseLectures` page), and finally the lecture module they are interested in (`LectureModules` page). Pages `Categories` and `LectureModules` are marked with an "L" label, which indicates that they are *landmark* pages. This property represents that the two pages will be reachable from any other page of the hypertext, by means of landmark links.

Contents of the selected lecture module are shown in page `LectureContent`. As represented by the `Module Scroller` unit, users can browse lecture modules in a *Guided Tour* navigation that allows moving forward and backward in the (ordered) set of modules available for the currently selected lecture. For each module, the data unit `Module Title` shows the title and a short description of the learning object, the `Contents` multidata unit shows texts and images that compose the module, while the `Definitions` hierarchical index shows titles of the definitions associated with the module and, nested, the corresponding contents. Three index units then show the lists of examples, tests and exercises available for the current lecture module. The selection of one item from such lists leads users in a different page where the corresponding contents are displayed.

The presentation of page `LectureContent`, as produced by the WebML code generator, can be seen in the right frame of the Web page depicted in Figure 7.

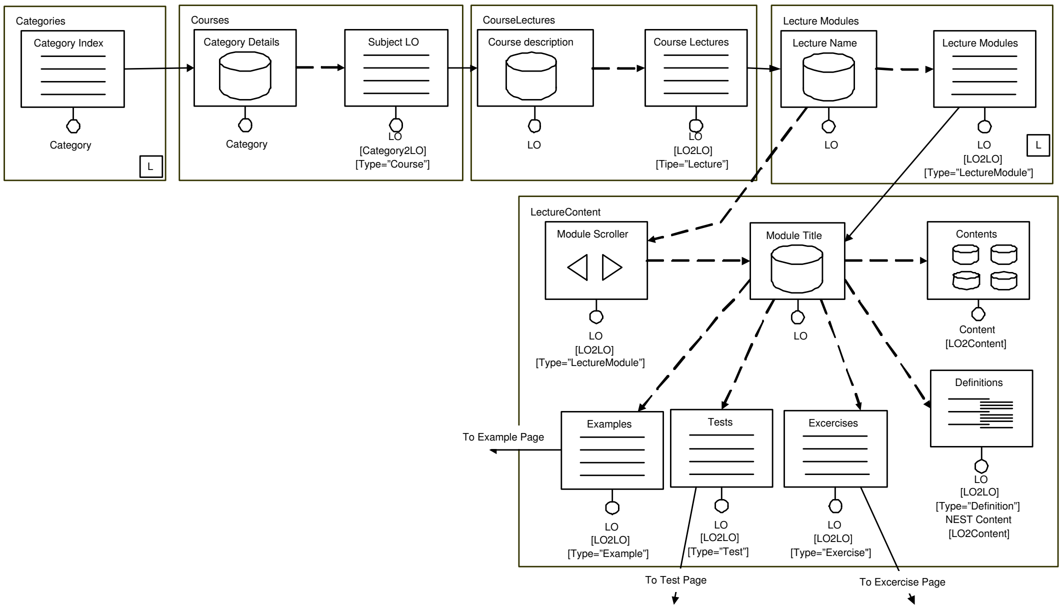


Fig. 2. The WebML specification of the hypertext interface for the vertical e-learning application.

2.3 Authoring Site View

In addition to the site view dedicated to learners, another WebML site view is dedicated to authors; it includes all the content management operations required in order to enter new learning objects and their metadata, or to alter and delete them. Content management site views are available in most Web applications (see e.g. [1] for details). They have a regular hypertext structure with provisions for indexing each entity's elements and then for adding, updating, or modifying each of them. In general, pages of the content management site view enable the management of several related entities, typically in one-to-many relationship (e.g., lessons and the related exercises).

In the vertical e-Learning, the authoring site view constrains authors to enter the mandatory meta-data ”before” entering learning objects; meta-data are indeed essential for the composition of learning objects and the building of curricula. Meta-data themselves can be updated and enriched, by using specific sections of the authoring site view.

3 UML-Guide Component

3.1 UML-Guide Overview

UML State diagrams [9] are used in UML-Guide for modelling the user navigation in a hypertext. Each *state* represents the production of a given information chunk on the device observed by a user, and each state *transition* represents an event caused by user interaction that leads to the production of a new chunk of information. State diagrams therefore provide an abstraction of *hypertext trails*, where each trail can be adapted by taking into account the user background, level of knowledge, preferences and so on [8]. In this way, UML state diagrams are a suitable interface for UML-Guide, whose primary purpose is to build adaptive hypermedia systems.

Atomic states can be grouped into *superstates*. States usually refer to concepts of an application domain; thus, they can correspond to the representation of WebML pages or page units, which enable the viewing of the information entities within the WebML data schema.

Parallel substates represent information chunks to be presented simultaneously. *Fork* and *join* pseudostates are used respectively for splitting and joining computations and enabling parallelism. The *SyncState* pseudostate is used for synchronizing substates of parallel regions.

Transitions represent active interconnections between information chunks, and usually correspond to associations in the application domain model (thus, they can correspond to WebML links, that interconnect pages and units, and in turn depend upon the relationships of the WebML data model). *Events* raise transitions in a state machine; they include user-generated or system-generated events, and the latter include time events. *Guards* can be used to constrain transitions by adaptation rules. Usually, they consist of a predicate over user profile attributes or context information.

Actions can be performed after a transition is raised and before entering a state. Also, transitions can be associated with *side effect actions*, whose effect is, for example, the modification of a user profile, or the choice of presentation styles for a given chunk of information. Actions can also process parameters used in guards of outgoing part of branches. Side effect actions, as well as adaptation rules, can be assigned to *entry*, *exit*, and *do* actions of states.

Tagged values are domain-specific properties used to extend the semantics of elements in UML diagrams. These values can refer, for example, to concepts of the structural model of the application domain, or to specific terminologies which might be useful to identify additional navigation requirements. We will make extensive use of tagged values for linking UML diagrams of UML-Guide to WebML concepts, as illustrated in Section 4.

3.2 UML-Guide State Diagram for e-Learning

The UML-Guide state diagram of Figure 3 illustrates a personalized learning environment for teaching object-oriented programming in JAVA, borrowed from a well-known Sun tutorial .[d] The chosen personalization example focuses on link adaptation; other adaptation aspects are covered in [8].

The tutorial starts with an overview of available lectures, as represented by the `Overview` state, which summarizes the available lectures in the tutorial, as specified by the `Summary` value in the `LearningPresentation` tagged value. It also presents the high level tutorial steps (`Tutorial` value in the `CourseStructure` tagged value). Links from the overview point not only to the first section of the tutorial, but also to the other main sections; all these links, except the first one, are associated with guard conditions that check that the user has enough knowledge to jump directly to the respective lectures.

The next step from the `Overview` is a lecture on the `Object Oriented Programming Concepts`. This state is accessible without any prerequisite on background knowledge; it is a composite state, containing five steps, represented by four substates: `What is an Object`, `What is a Message`, `What is a Class`, `Relations to Code`, and `Questions`. The `Relations to Code` state also shows an `entry procedure` addressing *content level adaptation*. The procedure applies to a learning step about building programs; it states that if the current user does not have sufficient knowledge on basic concepts about object-oriented programming procedures, then learning content on procedures will be added.

The next step from the `Object Oriented Programming Concepts` is the composite state `Language Basics`. The transition between the two states features a `next` event and a guard. The guard specifies a *link level adaptation* rule, saying that the link is recommended when current

[d] See `http://java.sun.com/docs/books/tutorial/java/index.html`.

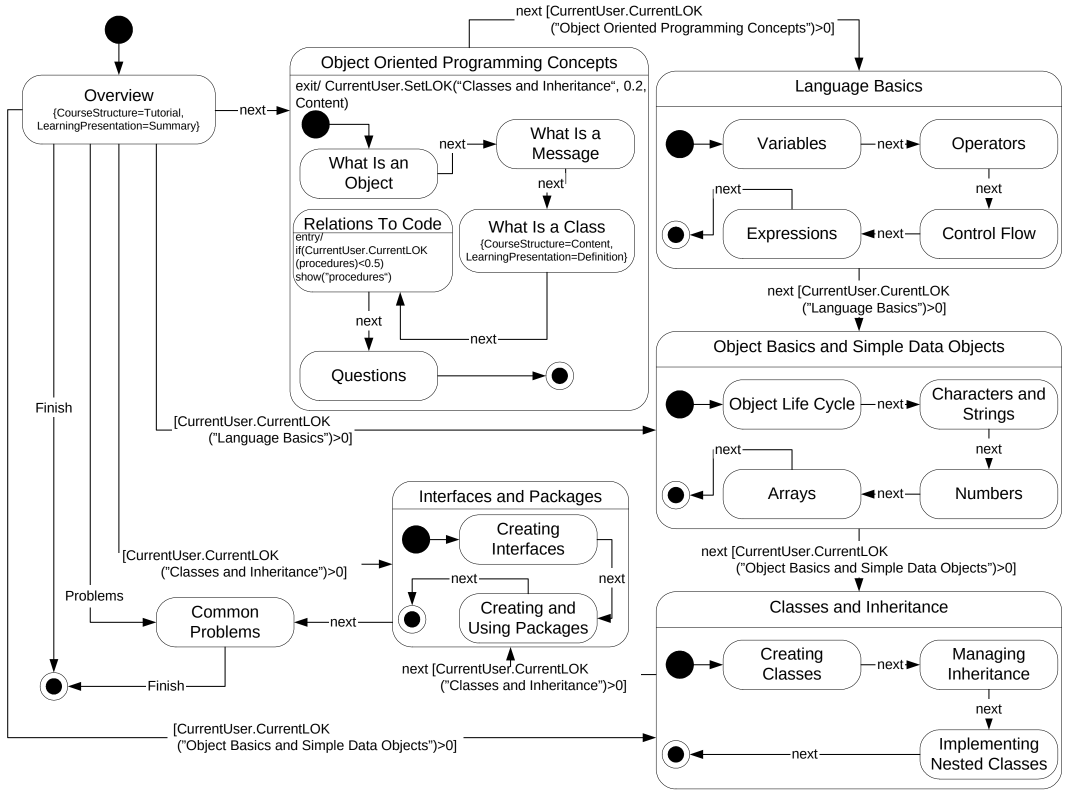


Fig. 3. A navigation model for a Java tutorial in the UML state diagram notation.

user level of knowledge is greater then zero. The other learning steps modelled in the state diagram can be interpreted similarly.

The personalization specification within state diagrams is based on the user model depicted in Figure 4. It is inspired by the LTSC IEEE 1484.2 Learner Model WG Standard proposal for public and private information (PAPI) for learner[e][f] The user model is composed of the classes `User` and `Performance`, plus an association expressing that a learner can have several performance records based on the acquired `LearningExperience` and `Competence`.

The `Performance` class stores the user's level of knowledge about the concepts described by the tutorial. This value is the one used for determining if a transition into a new state is appropriate and must be suggested to a given user. For example, the following condition:

```
[CurrentUser.CurrentLOK(''Classes and Inheritance'')>0]
```

is a guard that in the state diagram determines wether a link can be followed between the `Classes and Inheritance` state and the `Interfaces and Packages` state, based on current user level of knowledge. The `Performance` class maintains as well the value of competence, recorded

[e] `http://ltsc.ieee.org/archive/harvested-2003-10/working_groups/wg2.zip`
[f] For a more detailed learner profile, used e.g. in EU/IST Elena (`http://www.elena-project.org`), the reader is referred to the learner RDF bindings Web site at `http://www.learninglab.de/~dolog/learnerrdfbindings/`.

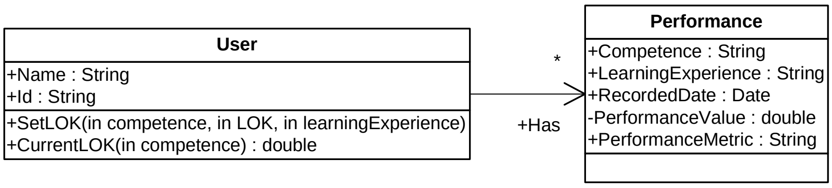


Fig. 4. A user model for the Java tutorial.

date, and metrics used to measure level of competence.

The `User` class provides operations to set and get the acquired level of knowledge or level of competence. These operations are used in guards and actions for adaptivity rules, and for updating learner profile. For example, in the state diagram of Figure 3, the user level of knowledge about "Classes and Inheritance" can be acquired either in the `Object Oriented Programming Concepts` lecture or in the `Classes and Inheritance` lecture. Exit procedures of these states indeed contain similar update operations, as the one which follows:

```
CurrentUser.SetLOK(‘‘Classes and Inheritance’’,0.2,Content).
```

In UML-Guide, state diagrams are used as input for visualizing navigation maps, whose structure is made of documents (nodes), composite nodes (folders), links (arrows), and parallel regions (dashed boxes). State diagrams are edited by means of the commercial tool Poseidon.[g] The navigation map is then generated through a transformation method [8], whose input is the state diagram encoded in XMI (as produced by Poseidon), and whose output is the map.

4 Integration of WebML and UML-Guide

The integration of WebML with UML-Guide proposed in this paper aims at composing a generic "vertical e-Learning" WebML application with a UML-Guide that is focused on a specific learning goal. We offer to the users of the composite system the standard, WebML-generated interface of the vertical, populated by content spawning a large body of knowledge; but we also offer to the focused learners a guide, available on an interface that can be opened "aside" the main one, and that points to pages and contents published by the WebML-generated interface, according to a specific learning objective and user experience.

The integration is loose and preserves the distinctive features of the two systems. In particular, some nodes and links in a UML-Guide state diagram point to content that is managed in the WebML e-Learning vertical; therefore, the integration of UML-Guide with WebML requires UML-Guide adopting few WebML concepts, such as page identifiers and content identifiers. In this way, concepts used as state names or as tagged values within UML-Guide are mapped to learning resources stored in the database generated from the WebML data model.

In the resulting application, the user-specific adaptation occurs in UML-Guide. This separation of concerns represents an extreme solution, as it is possible to support personalization [10] and adaptivity [11] directly in WebML. However, the proposed solution is an example of how client-side computations, specified at high-level in UML, can integrate WebML-designed solutions. As such, this experiment can be replicated for many other applications and the focus on UML-Guide can pursue different

[g] `http://www.gentleware.com/`

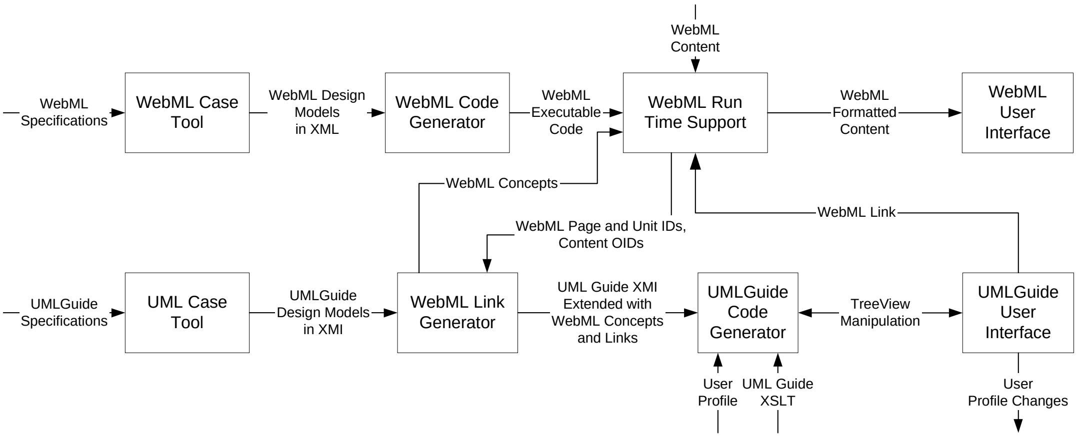


Fig. 5. Architecture of the composed system.

objectives.

Figure 5 describes the system architecture. The high-level WebML and UML-Guide specifications are mapped into XML-based internal representations, respectively built by the Code Generator component of WebRatio [7] and by the XMI [12] Generator of Poseidon.

The WebML run-time component runs JSP templates (also embedding SQL), and uses XSL style sheets for building the application's presentation. The XMI representation of a UML-Guide drives a run-time adaptation engine, written in XSLT, which dynamically changes the content of the profile variables and produces the UML-Guide user interface. The WebML and UML-Guide interfaces are then composed and presented to the user.

In this architecture, the main integration issue is concerned with the generation of WebML links "pointing" to the WebML-controlled portion of the application, to be addressed while building the UML-Guide interface. WebML links take the format:

```
ApplicationURL/page_identifier.do?ParameterList
```

where `page_identifier` denotes a WebML page and `ParameterList` is a list of tag-value pairs, in the form `entity_id.attribute=parameter`. Thus, UML-Guide state diagrams must be extended with tagged values to be used as pointers to WebML concepts. This activity must be performed by UML-Guide designers, typically in the course of the transformations required for "implementing" UML-Guides starting from their high-level descriptions (as illustrated in Figure 3).

It is worth noting that state diagrams augmentation does not require a complete mapping between UML-Guide components and WebML conceptual primitives. UML-Guide designers just need to specify IDs of those WebML elements that are used for URL construction, i.e., pages, source entities from which page contents are extracted, and entity attributes that are used in parametric selectors for retrieving page contents.

Figure 6 depicts an excerpt of state diagram extended with tagged values for WebML concepts needed for computing WebML links. `Object Oriented Programming Concepts` is a lecture. The corresponding page name is `LectureModules` from WebML hypertext model. The entity used to store lectures in the WebML data model is `LO`. The title used as an attribute to identify the

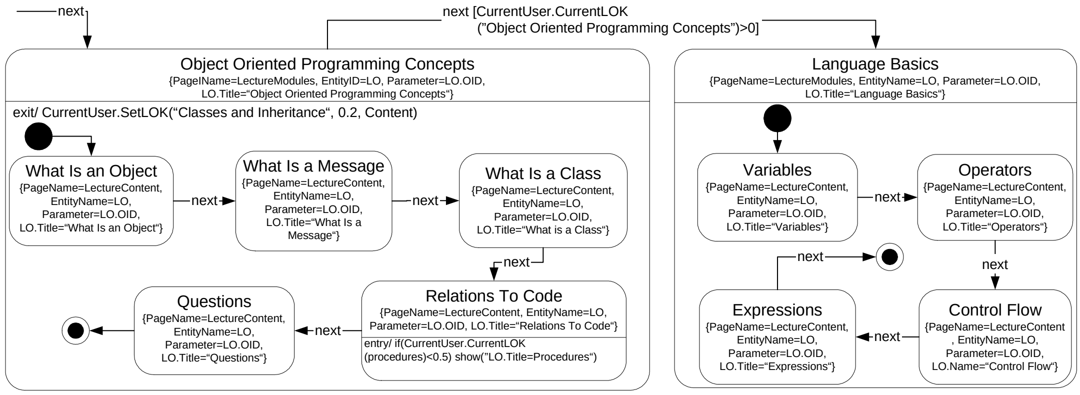


Fig. 6. Excerpt of the UML-Guide state diagram extended with tagged values representing WebML concepts.

lecture is the same as the state name. Entry and exit actions are transformed if they send parameters into WebML links, as it is in the case of `Relations To Code` (where the parameter of the `show` method is replaced by the specific WebML parameter `&LO.Title=Procedures`, used for selecting LOs about Java procedures). Although in our example tagged values for page and entity names are constant values, in more complex cases they can be specified as well as parametric selectors, so as to automatically retrieve their values from the XML WebML specification based on specific conditions.

Queries for retrieving OID's of the WebML concepts and content are submitted through a specifically designed interface to the WebML run-time components. The interface consists of the two functions `GetWebMLConcept(Type, Name)` and `GetWebMLRecordOID(Entity,Attribute, Value)`.

5 User Interface of the Integrated Application

Figure 7 presents the user interface of the integrated application. The UML-Guide generated map, obtained as a transformation of the UML state diagram depicted in Figure 3, is on the left; the WebML application, generated from the specification of Figure 2, is on the right. While the WebML application has an arbitrary interface, which depends on content composition within pages and on the adopted presentation style, the UML-Guide interface has a given structure that includes the following elements:

- *Folder symbol*—represents a composite information fragment composed by other (simple or composite) information fragments and links. The *composition* is visually represented by the plus/minus symbol, for showing/hiding enclosed items, and by the left hand indent of enclosed items. A content can be associated to each symbol.

- *Dashed box symbol*—represents a composite information fragment, which has to be presented concurrently with other composite information fragments (the dashed boxes) depicted on the same level.

- *Document symbol*—represents a simple information fragment; only links can be nested under it.

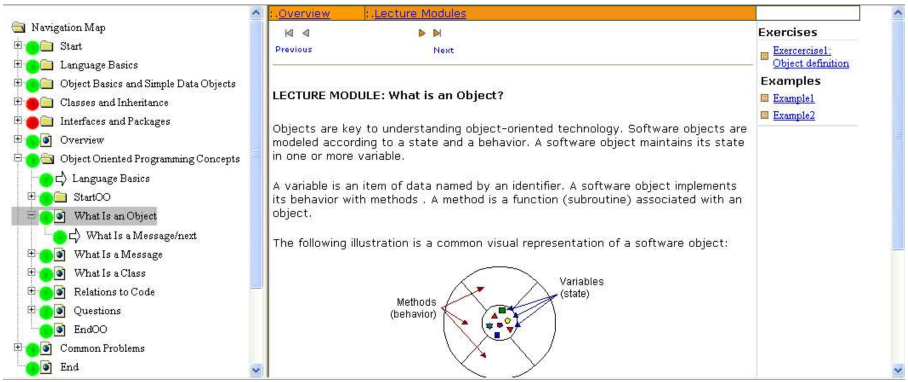


Fig. 7. Visualization of the navigation graph for the Java e-lecture.

- *Arrow symbol*—represents a link to another composite or simple information fragment; the arrow symbols can be nested under the folder when they represent different alternatives of suggested links starting from a particular document. Each arrow is associated with a content and a name of the corresponding target node. Also, the "/next" string is added to names of arrows representing guidance to the next fragment according to the course sequence.

- *Grayed background of nodes*—represents the currently presented node, i.e., the position reached by a user in the navigation map.

Presentation for the adaptive navigation support depends on the generator settings. For example, according to the traffic light metaphor, adaptive recommendations may be represented through different colors (green for nodes appropriate with respect to the current state of the user profile, red for not appropriate nodes, yellow for other situations—e.g. a node that has been already visited). Also, other metaphors might show, hide, or sort the nodes.

Profile records are maintained at the client side. When users begin a new session, their profile is initialized from a client-side XML-based database. The navigation map is manipulated at the client side as well. Javascript is used to implement the user interface control and user profile manipulation. The events generated by user actions on the user interface invoke profile adaptation actions, which possibly process and add new values to the user profile. They also trigger regeneration of the navigation map, according to the newly computed values.

The navigation map responds to changes in user profile by changing recommendation annotations (e.g., changing colors of nodes or showing/hiding nodes). When specific requirements, for example those set by conditions in entry actions of states, are met, the WebML vertical adapts delivered content based on additional parameters that UML-Guide is able to send to the server-side application.

Figure 7 highlights a lecture on "What is an Object". The UML-Guide panel placed on the left shows the position of the user reading the material for the module by the shaded background. The content of the lecture is delivered by the WebML vertical based on the generated link that is assigned to the document symbol at the "What is an Object" entry. The symbol is generated from the simple state with the same name depicted in Figure 6. The state has a transition to the next state "What is a Message", which in the UML-Guide panel is depicted as an outgoing arrow, under the symbol of the current lesson. As the user has sufficient background knowledge needed to understand the next step in

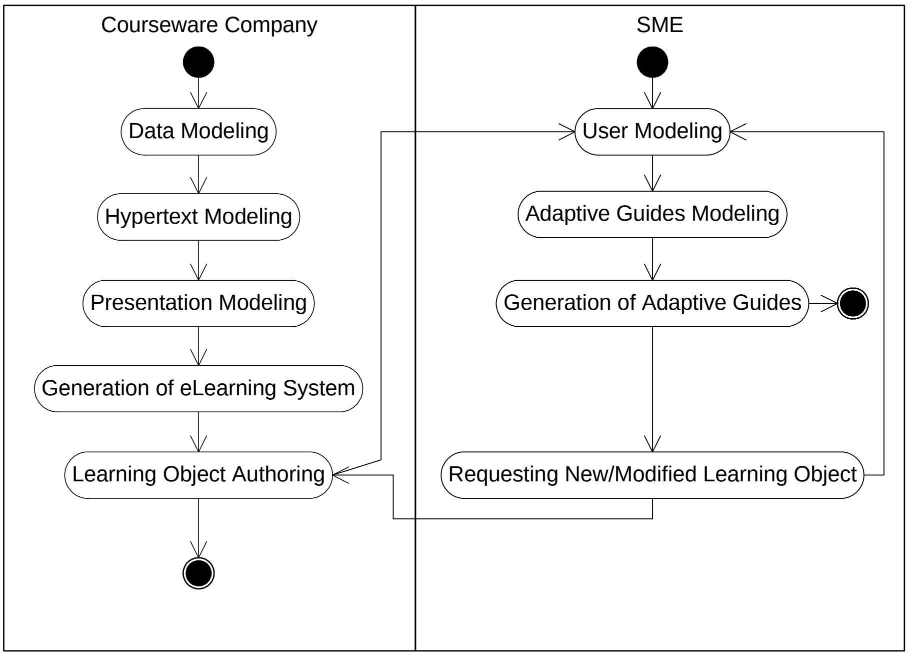


Fig. 8. Adaptive application design process.

his learning path, the direct next steps are annotated by a green ball. Further rules apply for additional entries to hide documents and folders which are not relevant to the user's learning goals.

The simple state "What is an Object" is a substate of the "Object Oriented Programming Concepts" state, which is rendered as a folder symbol in the navigation map. The constraints and side effect actions are transformed into conditions and procedure calls in the UML-Guide which dynamically generates the traffic-light annotations. Other symbols and their grouping under folders are generated similarly from the state diagram according to the method described in [8].

6 Adaptive Application Design Process

Figure 8 depicts the design process for building adaptive applications intermixing server and client side design steps; it applies to the scenario described in Section 1.1, and in general shows the relationship between top-down design on the server side and bottom-up design on the client side. This process is supported by the WebML method and engine on the courseware company side and the UML guide method and engine on the SME side.

The courseware company develops its e-Learning vertical according to well defined top-down steps, as illustrated in [1], consisting in designing the data content first, then the hypertext and finally the presentation. Such a process can be paired with the use of the WebRatio tool [7], which automatically generates the database for storing learning objects and the hypertext composing site views, Web pages, and the content chunks presented to the user. Then, of course, the courseware company also authors the learning objects in particular domains (e.g. modules about "Programming in Java"). The authoring consists of adding, changing or excluding learning objects and also managing their metadata, like classifying the objects according to topic, title, description, difficulty, background required, related learning objects and so on.

The SMEs start the design of the adaptive guide by collecting knowledge and background data of their employees, and then proceed by selecting bottom-up relevant contents from the body of learning objects that are made available. The SME designers gather information about the employee's skills and needs (e.g., the integration of Java programming and Oracle 9i), and build the adaptive user guides to be used throughout the SME; then they select personalized portions of such guides and install each of them on the employee's client application. This application is able to keep trace of the employees' progresses into the correspondent user model while they perform learning activities.

If a SME encounters a need for alteration or additions of new learning objects, the request for that activity is submitted to the courseware company. The courseware company alters/adds the new learning objects and updates their metadata. The database of the e-Learning vertical is updated at the courseware side and the curricula developers are notified about the new learning objects. In case the SMEs' curricula specifications have to be updated, the generation of user guides is repeated.

7 Related Work

Model-driven development of Web applications has been intensely investigated during last years [2, 13, 3, 4]. WebML has been proposed for the model-driven design of "data-intensive" Web applications. Its distinguishing feature is that the proposed design method is also supported by XML- and Java-based technologies, which enable the automatic generation of the application code [7].

During last years, some approaches have been proposed for extending traditional development methods by means of features enabling one-to-one personalization [14, 15, 4]. The aim is to customize the applications contents and the hypertext topology and presentation with respect to user preferences and needs, so as to offer an alternative to the traditional "one-size-fits-all" static approach in the development of Web systems [14, 16]. Some other proposals cover the adaptivity of Web applications with respect to dimensions characterizing the context of use (see [17] for a complete survey). WebML also offers some constructs for personalization. In particular, the application data schema can be extended through some entities representing user profiles and user access rights over the application content [10]. Recently, WebML has also been extended with some primitives for modelling context-aware applications, i.e., mobile, personalized applications that are also able to adapt to the current situation of use [11]. However, WebML, as well as the majority of Web personalization and adaptivity approaches so far proposed, manages personalization at server-side, and does not offer the alternative of managing user profiles and personalization policies at client side.

Conversely, the UML-Guide approach establishes model-driven design for adaptive applications, by considering link level adaptation and content level adaptation at the client side, where adaptation is computed according to the UML design specifications. First, requirements are modelled as variation points with mandatory and optional features in application domain models linked in collaboration diagrams [18]. Guard logical expressions and adaptivity actions are used in navigation specifications [8]. A rule based approach has been also employed in more open environment based on semantic Web models [19].

Although literature proposes works on client-side and server side methods for Web applications adaptivity, there is a lack of contributions towards the integration of the two approaches. One solution comparable to the one proposed in this paper is envisioned in [20], where authors propose a modelling framework in which integrated Web design methods for particular purposes can coexist with the stand alone ones.

Our integrated method pushes towards separation of concerns, proposing adaptivity design as a separate dimension with respect to content and hypertext design. Similarly, the e-Learning community has recognized that the definition of learning paths for navigating within learning objects is orthogonal to the design of learning object content itself. The efforts to establish IMS Learning Design [21] or ADL SCORM [22] sequences for provisioning metadata about learning paths through learning objects also show similar directions although the expensiveness is limited. With respect to this proposals our approach provides more flexibility, as a user can customize paths independently from the complex and expensive learning management systems. The method benefits from the UML graphical representations, which are easier to use in comparison to metadata authoring. One very interesting idea in this context would be to transform UML specifications into the metadata formats prescribed by those standards, to be able to exchange them between e-Learning systems like Blackboard[h] or CLIX[i].

8 Conclusions and Future Work

This paper has shown the integration between WebML and UML-Guide; the proposed approach demonstrates that server-side and client-side technologies can coexist and that it is possible, for both of them, to use model-driven code generation techniques starting from high-level requirements, expressed in graphical form. The proposed application scenario augments an e-Learning vertical so as to make it adaptable and personalisable. The design process is top-down for server-side, generic learning objects, and bottom-up for user-specific client-side adaptation.

The combination of the two methods offers several benefits. Among them the most relevant one is the orthogonality of adaptation design with respect to content and hypertext design. This is particularly useful in the e-Learning domain where, in order to increase the application effectiveness, the provision of LOs by courseware companies must be accompanied by the definition of personalized learning paths or individual curriculum sequences [23], based both on local learning strategies as well as on specific user competencies. These two requirements are not easy to identify by companies offering e-Learning services, while they are generally well-contextualized within organizations exploiting the services. The availability of client-side extensions therefore enables the latter to customize the application locally, according to their learning goals and the knowledge level of their members. With this respect, although WebML offers some constructs for specifying personalization of contents and services, UML-Guide state diagrams constitute an easy to use specification tool, especially due to the popularity of UML. Also, these diagrams allow defining learning paths and conditions for state transitions without mastering the complexity of the server-side application design, keeping the specification at a higher level of abstraction.

A further relevant aspect is that moving user dependent functionalities to the client side allows one to leave to users the full control over user-sensitive data. Users can decide on their own which information will be disclosed and which they will deny access to. This however increases the requirements on client-side tools, that must be able to manage small databases with information about users and process such data for achieving adaptation and personalization. As client machines are usually less powerful, this might result in some lacks of performance. We will further investigate and experiment with this approach in order to find a good balance between client-side and server-side processing. We are also planning an extension of the WebML CASE tool and of UML-Guide for providing automatic support to the integration of the two methods.

We regard this work as the first step of a deeper methodological inspection of the interactions be-

[h] `http://www.blackboard.com/`

[i] `http://www.im-c.de/homepage/index.htm`

tween UML and WebML, and more specifically of the possibility of using state diagrams, which best represent the modeling of dynamic interfaces, for collecting the requirements that can naturally evolve into WebML specifications. The experiments described in this paper, and specifically the mechanisms for rendering state transitions as WebML links, will be extended and reused. While in this paper we focused on UML state diagrams as integration tool, we aim at studying more complex integration scenarios where interaction will be modeled as well by collaboration and sequence diagrams.

References

1. S. Ceri, P. Fraternali, A. Bongio, M. Brambilla, S. Comai, and M. Matera. *Designing Data-Intensive Web Applications*. Morgan Kauffmann, 2002.
2. J. Conallen. *Building Web Applications with UML*. Object Technology Series. Addison Wesley, 2000.
3. F. Garzotto, P. Paolini, and D. Schwabe. HDM - a Model-Based Approach to Hypertext Application Design. *ACM Transactions on Information Systems*, 11(1):1–26, January 1993.
4. D. Schwabe, R. Guimaraes, and G. Rossi. Cohesive Design of Personalized Web Applications. *IEEE Internet Computing*, 6(2):34–43, March-April 2002.
5. K. Marriott, B. Meyer, and L. Tardif. Fast and Efficient client-Side Adaptivity for SVG. In *Proc. of WWW 2002, May 2002, Honolulu, Hawaii, USA*, pages 496–507. ACM Press, 2002.
6. G. South, A.P. Lenaghan, and R.R. Malyan. Using Reflection for Service Adaptation in Mobile Clients. Technical report, Kingston University-UK, 2000.
7. S. Ceri, P. Fraternali, A. Bongio, S. Butti, R. Acerbis, M. Tagliasacchi, G. Toffetti, C. Conserva, R. Elli, F. Ciapessoni, and C. Greppi. Architectural Issues and Solutions in the Development of Data-Intensive Web Applications. In *Proceedings of CIDR 2003, January 2003, Asilomar, CA, USA*, 2003.
8. P. Dolog and W. Nejdl. Using UML and XMI for Generating Adaptive Navigation Sequences in Web-Based Systems. In Perdita Stevens, Jon Whittle, and Grady Booch, editors, *UML 2003 - The Unified Modeling Language. Model Languages and Applications. 6th International Conference, San Francisco, CA, USA, October 2003, Proceedings*, volume 2863 of *LNCS*, pages 205–219. Springer, 2003.
9. Object Management Group. OMG Unified Modelling Language Specification, version 1.3, March 2000. Available at http://www.omg.org/. Accessed on June 1, 2001.
10. S. Ceri, P. Fraternali, and S. Paraboschi. Data-Driven One-to-One Web Site Generation for Data-Intensive Applications. In *Proceedings of Very Large Data Bases, September 1999, Edinburgh, UK*, pages 615–626. IEEE Computer Society, 1999.
11. S. Ceri, F. Daniel, and M. Matera. Extending WebML for Modeling Multi-Channel Context-Aware Web Applications. In *Proceedings of WISE - MMIS'03 Workshop (Mobile Multi-channel Information Systems), Rome, Italy, December 2003*, pages 615–626. IEEE Computer Society, 2003.
12. Object Management Group. OMG XML Metadata Interchange (XMI) Specification, version 1.1, November 2000. Available at http://www.omg.org/. Accessed on June 1, 2002.
13. P. Fraternali. Tools and Approaches for Developing Data-Intensive Web Applications: A survey. *ACM Computing Surveys*, 31(3):227–263, September 1999.
14. P. Brusilovsky. Adaptive Hypermedia. *User Modeling and User-Adapted Interaction*, 11(1-2):87–100, 2001.
15. N. Koch and M. Wirsing. The Munich Reference Model for Adaptive Hypermedia Applications. In P. De Bra, P. Brusilovsky, and R. Conejo, editors, *Adaptive Hypermedia and Adaptive Web-Based Systems, Second International Conference, AH 2002, Malaga, Spain, May 29-31, 2002, Proceedings*, volume 2347 of *Lecture Notes in Computer Science*. Springer, 2002.
16. P. Brusilovsky and W. Nejdl. Adaptive Hypermedia and Adaptive Web. In Munindar P. Singh, editor, *Practical Handbook of Internet Computing*. CRC Press, 2003.
17. G. Kappel, B. Proll, W. Retschitzegger, and W. Schwinger. Customization for Ubiquitous Web Applications — A Comparison of Approaches. *International Journal of Web Engineering and Technology*, 11, January 2003.
18. P. Dolog and W. Nejdl. Using UML-based Feature Models and UML Collaboration Diagrams to Information Modelling for Web-based Applications. In *UML 2004 - The Unified Modeling Language. Model Languages*

and Applications. 7th International Conference, LNCS. Springer, 2004. To appear.
19. P. Dolog, N. Henze, W. Nejdl, and M. Sintek. Personalization in Distributed e-Learning Environments. In *Proc. of WWW2004 — The Thirteen International World Wide Web Conference*, New Yourk, May 2004. ACM Press.
20. P. Dolog and M. Bieliková. Hypermedia Systems Modelling Framework. *Computing and Informatics*, 21(3):221–239, December 2002.
21. IMS. IMS Learning Design - Information Model. http://www.imsglobal.org/learningdesign/index.html, January 2003.
22. ADL. Sharable Content Object Reference Model (SCORM) 2004. http://www.adlnet.org/, February 2004.
23. G. Weber and P. Brusilovsky. Elm-art: An adaptive Versatile System for Web-Based Instruction. *International Journal of Artificial Intelligence in Education*, 12(4):351–384, 2001.