SOFTWARE—PRACTICE AND EXPERIENCE
Softw. Pract. Exper. 2003; **33**:1–32 (DOI: 10.1002/spe.523)

Model-driven design of collaborative Web applications

M. Matera[*,†], A. Maurino, S. Ceri and P. Fraternali

Dipartimento di Elettronica e Informazione, Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133 Milano, Italy

SUMMARY

This paper introduces a model-driven approach to the design of collaborative Web-based applications, i.e. applications in which several users play different roles, in a collaborative way, to pursue a specific goal. The paper illustrates a conference management application (CMA), whose main requirements include: (i) the management of users profiles and access rights based on the role played by users during the conference life cycle; (ii) the delivery of information and services to individual users; (iii) the management of the sequence of activities that lead to the achievement of a common goal. The presented approach is based on WebML, a conceptual modelling language for the Web. The paper also highlights some general properties—as understood by the practical experience of CMA development—that a Web modelling language should feature in order to fully support the development of collaborative applications. Copyright © 2003 John Wiley & Sons, Ltd.

KEY WORDS: Web application design; conceptual modelling; Web-based cooperative applications; CASE tools for Web application development

1. INTRODUCTION

Model-driven design of Web applications is a relatively recent discipline [1–7], which is gaining consensus among application analysts and developers. Its fundamental feature is that the overall organization of Web applications, in the same way as any other complex software system, can be initially specified at a high-level of abstraction, avoiding early commitment to detailed architectural and implementation issues. The abstract specification obtained can then be translated into a running application during the successive development phases.

Most model-driven specifications of Web applications typically apply to read-only Web sites, used primarily for delivering content through a browser; instead, in this paper we focus on *collaborative* Web applications. By this term, we mean applications executed by different groups of users that access

[*]Correspondence to: M. Matera, Dipartimento di Elettronica e Informazione, Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133 Milano, Italy.
[†]E-mail: matera@elet.polimi.it

Contract/grant sponsor: WebSI Project; contract/grant number: IST 2001 35458

Received 4 January 2002
Revised 15 August 2002
Accepted 3 January 2003

Web resources, both in read and write mode, playing different roles in a collaborative way, so as to achieve a common goal.

Collaboration among users complies with a specific workflow, which however does not rely over rigidly structured activities, with multiple levels of approvals, synchronization and monitoring; rather, in such applications each user independently goes through activities in progressive phases and produces results that altogether guarantee the achievement of common goals. Each single activity in a given phase is not further regulated: users can behave autonomously while performing a phase, with the only constraint that they must produce a result within the end of the phase.

Although not following rigid constrains, at certain times cooperative users must communicate and synchronize. In a Web application, which is asynchronously executed by each user according to the HTTP protocol, communication and synchronization have to occur by means of shared data. We assume that users of the same user group or of different groups may sometimes cooperatively modify the same documents. However, the main form of communication occurs in users of different groups sharing (parts of) documents, produced by one group during one phase of the process and consumed by another group during the successive phase, according to the classical 'producer–consumer' paradigm.

There is a broad range of applications that can be considered cooperative, e.g. appointments scheduling, document production within organizations and so on. Perhaps the most typical example of cooperative Web applications is the management of open source software development, like www.mozilla.org or www.sourceforge.org. These Web applications enforce a well-defined distinction of user roles and a sharp organization of activities to avoid problems like code overwriting or wrong versioning. For example, in the Mozilla Web application different groups of users (programmers, beta testers, bug reporters and managers) have distinct activities to accomplish and work together to create a freeware Internet browser. Similar features and design problems are also found in auction Web applications (see, for example, www.ebay.com), in which a number of users play different roles, such as potential buyer, sale proponent, etc.

In this paper we discuss a model-driven approach to the design of collaborative Web-based applications with the intent of showing how conceptual models, originally born for describing general purpose Web applications, can be adopted as they are, or at least with only a few extensions, for modelling Web-based collaborative applications. The discussion is based on our experience in designing and developing a conference management application (CMA), able to support conference review processes on the Web. The main requirements of this application include the management of users profiles and access rights based on the role played by users during the conference life cycle, the delivery of information and services personalized to individual users, and the management of the sequence of activities that lead to the achievement of a common goal—the selection of conference papers. The impulse to developing this application came to us from a 'competition' among different model-driven approaches to Web design, proposed by the organizers of the IWWOST 2001 Workshop [8]. Conference management case studies have been largely described in the literature (see, for example, [9]). The intent of the competition was however to discuss the new challenges posed by the deployment of such applications over the Web, which are also the challenges posed by the current evolution from read-only Web sites to complex and dynamic Web applications [10]. CMA falls into the category of collaborative Web applications, because it is performed by users who play different roles in a collaborative way in order to pursue the specific goal of selecting the (hopefully) best papers for the conference in a (hopefully) fair way.

Throughout this paper we use WebML (Web Modelling Language) [11,12], a language for the conceptual modelling of Web applications, developed in Academia and currently employed by a CASE tool for the development of data-intensive Web sites, called WebRatio [13]. However, our observations also apply to other conceptual models like, for example, OOHDM [7] and Araneus [1].

The paper is organized as follows. Section 2 gives a brief introduction on the WebML method and the visual notation it is based on. Section 3 reports on the analysis of the CMA requirements. Sections 4 and 5 show the adopted WebML solution for CMA design. Section 6 briefly describes the implementation through the CASE tool supporting the WebML-based design. Section 7 includes reports on the lessons learned by the CMA case study. Finally, Section 8 draws the conclusions.

2. THE WebML METHOD AT A GLANCE

WebML is a visual language for specifying the content structure of a Web application and the organization and presentation of such content in a hypertext [11,12]. In addition to having a visual representation, conveying the essential features of its primitives, WebML is also provided with an XML-based textual representation, used to specify additional detailed properties, not conveniently expressible in the graphical notation. WebML specifications can, therefore, be represented as visual diagrams as well as XML documents and this makes the automatic processing of specifications by CASE tools possible. In particular, WebML automatic processing is supported by the WebRatio CASE tool [8], which is able to translate visual specifications into concrete page templates and also map them onto data sources, thus making it easier to master the complexity of Web application development.

As reported in Figure 1, the WebML approach to the development of Web applications consists of different phases. Inspired by Boehm's spiral model [14], and in line with modern methods for Web and software applications development [10,15,16], the WebML process must be applied in an iterative and incremental manner, in which the various phases are repeated and refined until results meet the application requirements. The product lifecycle therefore undergoes several cycles, each producing a prototype or a partial version of the application. At each iteration, the current version of the application is tested and evaluated and then extended or modified to cope with the previously collected requirements, as well as the newly emerged requirements. Such an iterative and incremental lifecycle appears particularly appropriate for the Web context, where applications must be deployed quickly (in 'Internet time') and requirements are likely to change during the development time.

Out of the entire process illustrated in Figure 1, the 'upper' phases of analysis and conceptual modelling are those most influenced by the adoption of a conceptual model. This paper will focus on them, by showing how they specialize for the development of collaborative Web applications. Some issues about implementation will be also discussed, by showing how conceptual schemas derived from the design phases can be translated into a running application through the WebML CASE tool.

In the rest of this section the different activities in the WebML development process will be introduced. The WebML notations for the definition of conceptual schemas are also briefly illustrated.

2.1. Requirements analysis

Requirements analysis focuses on collecting information about the application domain and the expected functions, and specifying them through easy-to-understand descriptions. The input to this

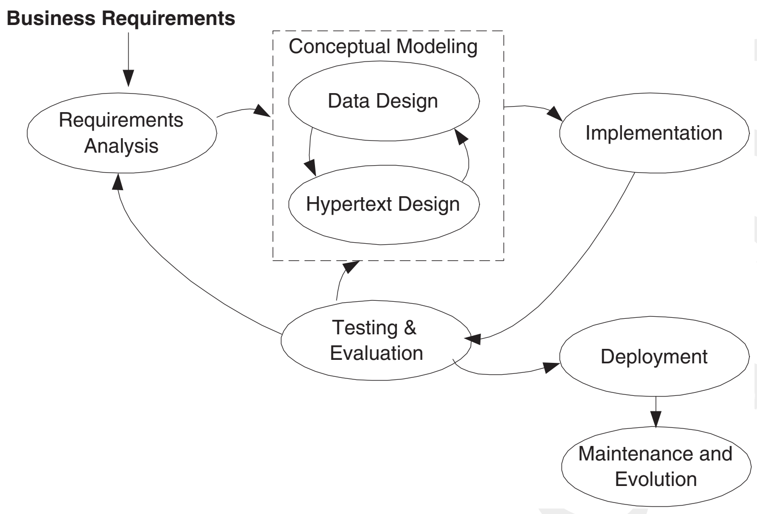


Figure 1. Phases in the WebML development process.

activity is the set of business requirements that motivate the application development. The main results of this phase are as follows.

- The identification of the *groups of users* addressed by the application. In cooperative applications, each group represents users playing the same role within the process; more precisely, users in the same group perform the same activities with the same access rights over the same information classes. It is worth noting that the same physical user may also play different roles, thus belonging to different groups.
- The specification of *functional requirements* that address the functions to be provided to users. For each group of users, the relevant activities to be performed are identified and specified; each activity is a cohesive set of elementary tasks [17].
- The identification of *core information objects*, i.e. the main information assets to be accessed and/or manipulated by users.
- The decomposition of the Web application into *site views*, i.e. different hypertexts designed to meet a well-defined set of functional and user requirements. Each user group will be provided with at least one site view supporting the functions identified for the group. When a group performs several phases of the cooperative process, it may be provided with several site views.

The WebML method does not prescribe any specific format for requirements specification. However, as also shown later on in this paper, table formats are suggested for capturing the informal requirements (such as the group description table or the site views description table). UML use case diagrams and activity diagrams [18] can also be used as standard representations of usage scenarios and activity synchronization, respectively. In particular, functional requirements are captured by activity flow, showing sequentiality, parallelism and synchronization among the activities to be performed by different users groups.

2.2. Conceptual modelling

Conceptual modelling consists of defining WebML-based conceptual schemas, which express the organization of the application at a high level of abstraction, independently from implementation details. According to the WebML approach, conceptual modelling consists of *data design* and *hypertext design*.

- *Data design* corresponds to organizing core information objects previously identified during requirements analysis into a comprehensive and coherent data schema, possibly enriched through derived objects. In the case of collaborative Web applications, prominent data design activities also include:
    - *group modelling*, whose aim is to translate the requirements of user groups into entities and relationships in the data schema;
    - *identification of synchronization objects*, whose aim is to determine entities or relationships that must be produced by users of one or more groups in a given phase and consumed by users of different groups in the following phase;
    - *identification of shared objects*, whose aim is to determine which objects within a given phase support communication among the users of the same group.
- *Hypertext design* produces site view schemas on top of the data schema previously defined. Site views express the composition of the content and services within hypertext pages, as well as the navigation between units and pages. In the case of process-oriented applications, where different user groups take part in the process, hypertext design consists of defining multiple site views for the user groups involved. As highlighted in the case study discussion, for collaborative Web applications the hypertext design activities have the following objectives:
    - *supporting the association of users with their activities depending on the roles played*, which consists of mapping users onto appropriate groups and site views, depending on the phase of the collaborative process reached;
    - *supporting profile management*, enabling the creation of profile information (done either by application administrators or by users themselves);
    - *supporting personalization*, for providing users with contents and services personalized according to their needs;
    - *supporting phase transition*, for making the process evolve along different phases.

The models provided by the WebML language for data and hypertext design are briefly described in the following. A broader description of the language and its formal definition can be found in [11,12,19] and at http://webml.org.

2.2.1. WebML data model

Data modelling is one of the most traditional and consolidated disciplines of information technology, for which well-established modelling languages and guidelines exist. For this reason, WebML does not yet propose another data modelling language; rather, it exploits the successful and popular notation of the entity-relationship data model. The fundamental elements of the WebML data model are therefore *entities*, defined as containers of data elements, and *relationships*, defined as semantic connections between entities. Entities have named properties, called *attributes*, with an associated type. Entities can be organized in *generalization hierarchies* and relationships can be restricted by means of *cardinality constraints*.

In the design of Web applications it is often required to calculate the value of some attributes or relationships of an entity from the value of some other elements of the schema. Attributes and relationships so obtained are called *derived*. The entity-relationship model does not include any standard notation for characterizing derived attributes and relationships, neither is it a language for expressing their computation rules. However, the specification of an attribute or of a relationship can be easily extended. For example, borrowing the notation and the OCL language from UML [18], derived attributes and relationships can be denoted by adding a slash character '/' in front of the attribute or relationship name and their computation rule can be specified as an OCL expression added to the declaration of the attribute or relationship.

2.2.2. WebML hypertext model

The hypertext model enables the definition of the hypertext, which is shown to a user in the browser. It enables the definition of pages and their internal organization in terms of elementary units for displaying content. It also supports the definition of links between pages and content units that facilitate information location and browsing. Some elementary units then specify operations, such as content management or user's login/logout procedures.

The overall structure of a hypertext is defined in terms of site views, areas, pages and content units. A *site view* is a particular hypertext, designed to address a specific set of requirements. It consists of *areas,* which are the main sections of the hypertext and comprises recursively other sub-areas or pages. *Pages* are the actual containers of information delivered to the user.

Several site views can be defined on top of the same data schema, for serving the needs of different user communities or even for arranging the composition of pages to meet the requirements of different access devices like PDAs, smart phones and similar appliances.

Figure 2 gives an example of the organization of pages and areas in a site view, which in a conference management system allows the conference chair to set-up the conference. The site view is composed by a home page, which is the first page accessed when the conference chair logs into the application. This page gathers access to two areas: the *Conference Set-up* area, including only one page through which the conference chair enters data describing the conference, and the *Manage PC Members* area, including three pages that show the list of already nominated program committee members and support the modification of their data or the insertion of new members.

Pages and areas are characterized by some distinguishing properties, which highlight their 'importance' in the Web site. In particular, pages inside an area or site view can be of three types.

- The *home page* (denoted with a small 'H' inside the page icon) is the page at the default address of the site, or the one presented after the user logs into the application; it must be unique.
- The *default page* (denoted with a small 'D' inside the page icon) is the one presented by default when its enclosing area is accessed; it must be unique within an area.

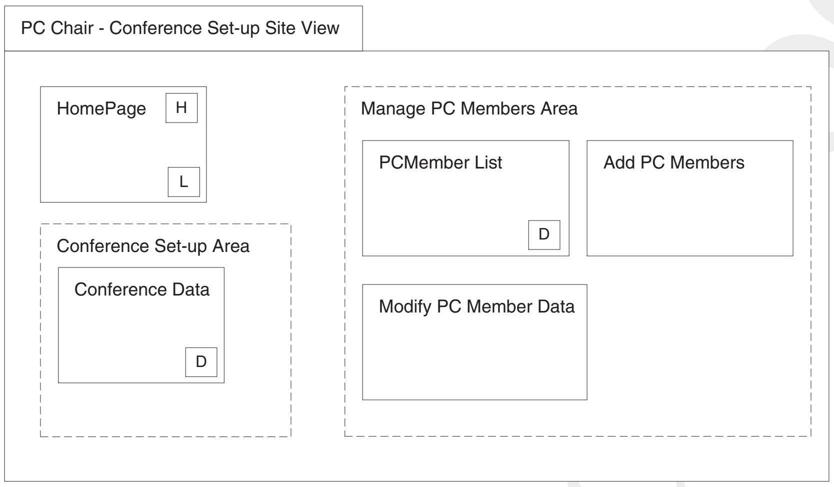


Figure 2. Example of site view composition based on areas and pages.

- A *landmark page* (denoted with a small 'L' inside the page icon) is reachable from all the other pages or areas within its enclosing site view.

For example, in Figure 2 the home page is also a landmark page, meaning that a link to it will be available from any other page of the site view. Also, the *Conference Data* page and the *PCMember List* page are default pages for their enclosing areas. This implies that the two pages are entry points for the two areas.

2.2.2.1. Page composition. Once areas and pages have been identified for a given site view, hypertext design proceeds by composing pages. Pages are made of *content units*, which are the elementary pieces of information extracted from data sources and published within pages. Table I reports the five WebML core *content units*, representing the elementary information elements that may appear in the hypertext pages. Units represent one or more instances of entities of the structural schema, typically selected by means of queries over the entity attributes or over relationships. In particular, *data units* represent some of the attributes of a given entity instance; *multidata units* represents some of the attributes of a set of entity instances; *index units* present a list of descriptive keys of a list of entity instances and enable the selection of one of them; *scroller units* display the elements of a sequence of entity instances one by one. Finally, *data entry* units represent forms for collecting input values into fields.

Data, multidata, index and scroller units include a source and a selector. The *source* is the name of the entity from which the unit's content is queried. The *selector* is a predicate, used for determining the *actual objects* of the source entity that contribute to the unit's content.

The previous collection of units is deemed to be sufficient to logically represent content of arbitrary nature on a Web interface [12]. However, some extensions are also available, for example the *multi-*

Table I. The five content units in the WebML composition model.

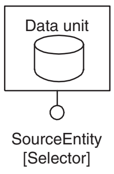
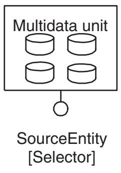
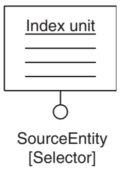
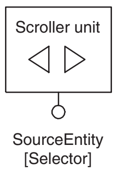
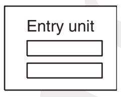
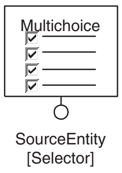
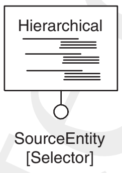
Data unit	Multidata unit	Index unit	Scroller unit	Entry unit
				

Table II. Two index unit variants.

Multi-choice index unit	Hierarchical index unit
	

choice and the *hierarchical indexes* reported in Table II. These are two variants of the index unit that, respectively, allow the choosing of multiple objects and organizing the list of index entries defined over two or more entities hierarchically (see [12] for greater detail).

2.2.2.2. Link definition. Units and pages are interconnected by links, thus forming a hypertext. Links between units are called *contextual*, because they carry some information from the source unit to the destination unit, which at least corresponds to the identifier of one of the objects displayed by the source unit. In contrast, links between pages are called *non-contextual*. In contextual links, the binding between the source unit and the destination unit of the link is formally represented by *link parameters*, defined over the link, and by *parametric selectors*, defined in the destination unit. A link parameter is a value associated with a link between units, which is transported, as an effect of the link navigation, from the source unit to the destination unit. A parametric selector is, instead, a unit selector whose condition mentions one or more parameters.

As an example of WebML specification, Figure 3 reports a simple hypertext, consisting of two pages. The page *SubmittedPapersList* includes an index unit (Paper Index) and a data unit (Paper Details), defined over the entity Paper. The former shows the list of all the Paper instances; the latter shows details about one single instance, selected from the index by following a contextual link. This link transports as a parameter the identifier of the selected item (CurrPaper) and the data unit uses

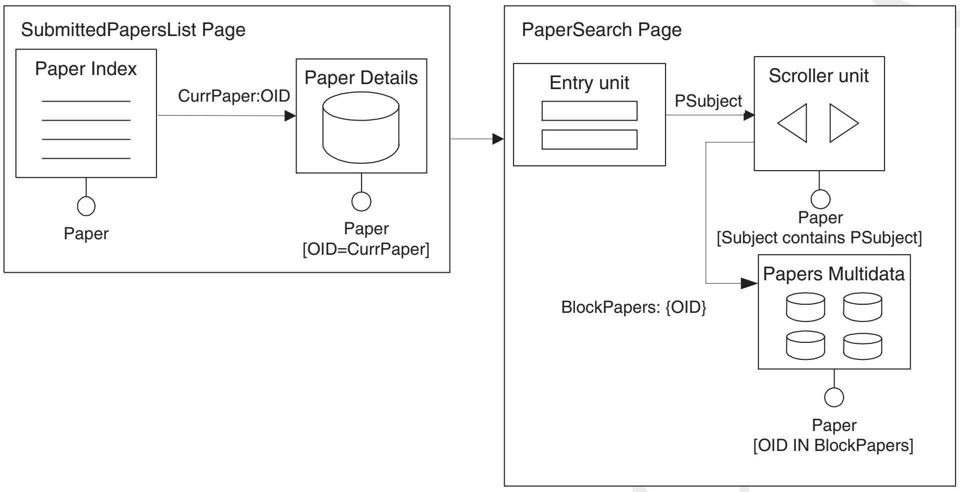


Figure 3. Example of page composition and contextual and non-contextual navigation.

this parameter for displaying the instance details. A non-contextual link allows instead navigation from the *SubmittedPapersList* page to the *PaperSearch* page where, independently from the content displayed in the first page, it is possible to perform a search about papers submitted for a given subject. An entry unit allows a subject name to be entered as a search keyword. The link in the output from the entry unit transports information such as link parameters (PSubject) to the scroller unit, which retrieves all the matching papers. Then, a multidata unit (Papers Multidata) displays the attributes of the retrieved papers one block at a time and the user may access the first, last, previous or next paper block in the sequence by means of scrolling commands. In contextual link definition, source unit object identifiers (OID) can be considered the default context transported by the links. Therefore, the explicit specification of the corresponding parameter over the link and of the parametric selector in the destination unit can be omitted.

In some applications, it may be necessary to specify a different and more sophisticated link behaviour, whereby the content of some unit is displayed as soon as the page is accessed, even if the user has not navigated its incoming link. This effect can be achieved by using *automatic links*. An automatic link, graphically represented by putting a label ‘A’ over the link, is ‘navigated’ in the absence of a user’s interaction when the page that contains the source unit of the link is accessed. Also, there are cases in which a link is used only for passing context information from one unit to another and thus is not rendered as an anchor. This type of link is called a *transport link*, to highlight that the link enables only context-information passing and not user navigation. Transport links are graphically represented through dashed arrows.

2.2.2.3. Context parameters. In some cases, context information is not transferred point to point during navigation, but can be set as globally available to all the pages of the site view. This is possible

Table III. The WebML context parameter units.

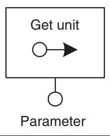
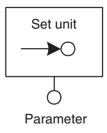
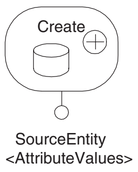
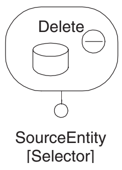
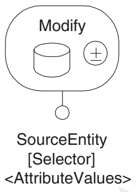
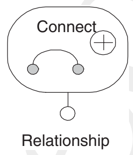
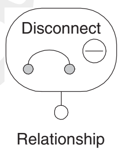
Get unit	Set unit
	

Table IV. The WebML operation units.

Create unit	Delete unit	Modify unit	Connect unit	Disconnect unit
				

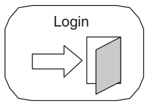
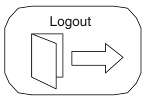
through the WebML notion of context parameters. Parameters can be set through the *set* unit and consumed within a page through a *get* unit. The visual representation of such two units is reported in Table III.

2.2.2.4. Content management operations. In addition to the specification of read-only Web sites, where the user interaction is limited to information browsing, WebML also supports the specification of services and content management operations requiring write access over the information hosted in a site (e.g. the filling of a shopping trolley or an update of the users' personal information).

WebML offers additional primitives for expressing built-in update operations, such as creating, deleting or modifying an instance of an entity (represented through the *create*, *delete* and *modify* units, respectively), or adding or dropping a relationship between two instances (represented through the *connect* and *disconnect* unit, respectively). The visual representation of such units is reported in Table IV.

Other utility operations extend the previous set. For example, *login* and *logout* (see Table V), are, respectively, used for managing access control and verifying the identity of a user accessing the application site views and for closing the session of a logged user. Other arbitrary application-dependent services and business components, like e-payment services, can be represented using the

Table V. Login and logout operations, supporting site view access control.

Login unit	Logout unit
	

general concept of *generic operation*, i.e. 'black boxes', whose internal details are not specified in WebML, which can, however, be linked to WebML units or pages.

Operation units do not publish the content to be displayed to the user, but execute some processing as a side effect of the navigation of a link. Like content units, operations may have a source object (either an entity or a relationship) and selectors, may receive parameters from their input links, and may provide values to be used as parameters of their output links. The result of executing an operation can be displayed in a page by using an appropriate content unit, for example a data or multidata unit, defined over the objects updated by the operation.

Regardless of their type, WebML operations may have multiple incoming contextual links, which provide the context necessary for executing the operation. One of the incoming links is the activating link (the one followed by the user for triggering the operation), while the others just transport contextual information and parameters, for example the identifiers of some objects involved in the operation. Two or more operations can be linked to form a chain, which is activated by firing the first operation.

Each operation can have two types of output links: *one OK link* and *one KO link*. The former is followed when the operation succeeds; the latter when the operation fails. The selection of the link to follow (OK or KO) is based on the outcome of operation execution and is under the responsibility of the operation implementation.

2.3. Implementation

The WebRatio CASE tool largely assists designers in the implementation of the database and of the Web hypertext. First of all, it offers a visual environment for drawing the data and hypertext conceptual schemas. Such visual specifications are then stored as XML documents and these are the inputs for the WebML code generator that, on their basis, supports data and hypertext implementation. In particular, it provides an interface to the data layer that assists designers in automatically mapping the conceptual-level entities, attributes and relationships to physical data structures in the data sources, where the actual data will be stored. It also automatically produces a set of dynamic page templates and unit descriptors, which enable the execution of the application in the runtime layer. A page template is a template file (e.g. a JSP file), which expresses the content and mark-up of a page in the mark-up language of choice (e.g. in HTML, WML, etc.). A unit descriptor is an XML file that expresses the dependencies of a WebML unit from the data layer (e.g. the name of the database and the code of the SQL query from which the population of an index must be computed). Both the templates and the unit descriptors are produced by a set of translators coded in XSL and executed on the XML specification by a standard XSL processor.

2.4. Further development phases

The remaining phases in the development lifecycle consist of *testing* and *evaluating* the application in order to improve its internal and external quality, *deploying* the application on top of a selected architecture and *maintaining* and possibly *evolving* the application once it is deployed. Such phases are not specifically covered in this paper. However, we want to point out that, within the WebML method, great support for testing and evaluation comes from the availability of a CASE tool. In addition to offering automatic support to data and hypertext implementation, the tool assists the population of data sources with randomly generated data, taken either from a default population or from attribute-specific populations; therefore, testing and evaluating a 'realistic' prototype is possible and the iteration in the process lifecycle can be fully realized.

Furthermore, in a model-driven process like that based on WebML, maintenance and evolution benefits from the existence of a conceptual model of the application. Requests for changes can in fact be turned into changes at the conceptual level, either to the data model or to the hypertext model. Then, changes at the conceptual level are propagated to the implementation. This approach smoothly incorporates change management into the mainstream production lifecycle and greatly reduces the risk of breaking the software engineering process due to the application of changes solely at the implementation level.

3. CMA REQUIREMENTS ANALYSIS

The CMA requirement analysis has been carried out in conformance with the activity prescribed by the WebML methodology. This section illustrates the business requirements from which the analysis was started and then shows the most relevant analysis artifacts directly related to user and process management analysis—these are the dimensions that most characterize the development of collaborative applications. A complete description of the CMA development can be found in [20].

3.1. CMA business requirements

CMA aims at offering support for the process of paper submission, evaluation and selection within a conference. The application covers the entire process of managing paper selection, from the conference definition, to paper submissions, paper assignments to PC members, paper reviewing, paper selection and conference program publishing.

Different actors perform their assigned activities through the CMA as follows.

1. The **Program Committee Chair** (PC Chair) is in charge of managing the whole conference.
    - They are responsible for creating a new conference instance and defining the important conference dates (e.g. the deadline for paper submission, the end of the review process and the author notification date).

- They determine the conference tracks and subjects—the conference is supposed to have a set of tracks and, optionally, a set of subjects.
- They establish the Program Committee, by pre-registering PC Members.
- They assign papers to be reviewed by PC Members. This assignment must take into account possible conflicts, which can arise if the PC Member affiliation coincides with the paper author affiliation, or if the PC Member explicitly declares to be in conflict with some papers. The assignment must also be based on possible preferences expressed by PC Members with respect to tracks, subjects and papers.
- Advised by PC Members, they define the final list of accepted and rejected papers.

2. **PC Members** are in charge of reviewing the papers assigned to them by the PC Chair. They are pre-registered by the PC Chair, but must confirm their registration personally, including entering affiliation and contact information. PC Members perform the following activities.

   - During paper submission, they may indicate their preferences for conference tracks and subjects.
   - After paper submission, they may indicate preferences for some of the submitted papers.
   - Once they have been assigned with papers to review, they may designate other people as reviewers of some papers.
   - They take part to the final discussion about paper acceptance, advising the PC Chair in determining the final list of accepted and rejected papers.

3. **Reviewers** are designated and pre-registered by PC Members for the evaluation of some papers and use the CMA for entering their reviews.
4. **Authors** can submit one or more papers. In order to do this, they first need to register with the application.

3.2. Identification of groups

Most CMA user groups directly correspond to the actors described by business requirements, namely *PC Chair*, *PC Member*, *Reviewer* and *Author*[‡]. In order to access the system, the users in these groups must be registered. In particular the PC Chair personal data are pre-inserted by a system administrator before the beginning of the conference management. PC Members are pre-registered by the PC Chair during conference set-up. Similarly, Reviewers are pre-registered by PC Members. Authors are instead required to personally register before submitting a paper.

In addition to these user groups, a further group, *Anonymous*, is needed to represent those non-registered users who do not play any special role in the conference management (e.g. prospect authors, casual readers, etc.), but that could need access to a public application front-end. It is worth noting

[‡]The group Author includes only one author per paper, that is the one who registers with CMA for submitting the paper. Since coauthors are not involved in the conference management process, they are not represented by a user group. Data about possible coauthors for each paper needs, however, to be managed by the application, for example for determining if a paper is in conflict with a given PC Member. For this reason, an entity in the data schema will be devoted to representing information about coauthors (see Section 4).

that even registered users (i.e. PC Chair, PC Members, Reviewers and already registered Authors) are considered anonymous until they do not authenticate themselves.

Once the list of user groups has been identified, properties characterizing each group must be defined. Each registered user may be described by the following attributes: *UserName, Password, Name, Surname, Email, Affiliation*. This last attribute is fundamental for PC Members and Authors, because during paper assignment to PC Members, it allows possible conflicts between PC Members and Authors' affiliations to be checked. Additionally, PC Members and Reviewers are also characterized by the *Number of Papers to Review*, which allows the number of papers already assigned for review by the PC Chair during paper assignment to be monitored.

A compact description for the above-mentioned user groups is reported in Table VI. For each group, some preliminary indications are also provided about the information objects that the group will manage in access or write mode, as well as the most relevant usage scenarios the group will be involved with. Note that some of these requirements might not be known during the first analysis session. However, since the WebML process is iterative, requirements could even be discovered or refined during the following phases, so generating process iteration.

3.3. Functional requirements analysis

Functional requirements address the essential functions that the application should deliver to its users. A practical way of gathering functional requirements is to identify and examine a number of representative usage scenarios (or *use cases*, in the UML terminology). The identification of groups is preliminary to the study of functional requirements, because it is more natural to examine the application usage scenarios by considering the requirements of each group separately.

In CMA, usage scenarios change or evolve according to a specific workflow; therefore, it is particularly useful to conduct the functional analysis by identifying and representing interactions among different activities, such as sequentiality, synchronization or parallelism. A UML activity diagram could serve the purpose of representing the activities flow.

The CMA activity diagram is shown in Figure 4. The synchronization bars in the diagram represent the points where some data have been 'produced' by certain activities and are ready to be 'consumed' by successive activities. We can say that each synchronization bar represents a phase transition. It therefore results that the activities of CMA user groups can be organized in seven different phases, as follows.

- *Conference Set-up*, during which the PC Chair defines the conference relevant data and sets up the program committee by designating PC Members.
- *Paper Submission*, during which Authors submit papers (after registering) and PC Members express their preferences about conference tracks and subjects.
- *Paper Bidding*, during which PC Members express preferences about the submitted papers.
- *Paper Assignment*, during which the PC Chair, based on the PC Member preferences about tracks, subjects and papers, assigns papers to PC Members, taking care to avoid conflicts.
- *Paper Review*, during which PC Members can insert reviews for their assigned papers and can designate other Reviewers. Once designated, Reviewers can also enter their reviews.
- *Final Discussion*, during which reviews are finalized and on their basis the final program is prepared.

Table VI. Group description table.

PC Chair	
Description	It represents the PC Chair, who is in charge of managing the whole conference management process. The PC Chair is pre-registered by the Web application administrator.
Profile data	*UserName, Password, Name, Surname, Email, Affiliation.*
Objects accessed in read mode	Conference data, user data, PC Member preferences, papers, paper reviews.
Objects accessed in write mode	Conference data (including Tracks and Subjects), user data (for PC Member creation and personal profile modification).
Relevant usage scenarios	Conference Set-up, Paper Assignment, Review Monitoring, Final Discussion, Conference Management Closing.
PC Member	
Description	It clusters all the PC Members designated by the PC Chair. The PC Chair is in charge of registering PC Members as users of the CMA application. They must however confirm their registration.
Profile data	*UserName, Password, Name, Surname, Email, Affiliation, Number of papers to review*.
Objects accessed in read mode	Conference data (including Tracks and Subjects), Papers, Reviews.
Objects accessed in write mode	User data (for reviewers' creation and for personal profile modification), preferences about tracks, subjects and papers, paper reviews.
Relevant usage scenarios	Paper Bidding, Reviewer Creation, Review Insertion, Final Discussion.
Reviewer	
Description	It clusters all the Reviewers designated by PC Members for reviewing their assigned papers. PC Members are in charge of pre-registering their reviewers.
Profile data	*UserName, Password, Name, Surname, Email, Affiliation, Number of Papers to Review*.
Objects accessed in read mode	Conference data (including Tracks and Subjects), Paper.
Objects accessed in write mode	Review.
Relevant usage scenarios	Review Insertion.
Author	
Description	It clusters the paper contact authors that register to the application for submitting a paper.
Profile data	*UserName, Password, Name, Surname, Email, Affiliation.*
Objects accessed in read mode	Conference data (including Tracks and Subjects)
Objects accessed in write mode	User data (for registering themselves as a user), Papers.
Relevant usage scenarios	Paper Submission.
Anonymous	
Description	It represents all the users that do not play a role in the conference management, but that might need to access the public site view of the CMA as, for example, for prospective authors or casual users.
Profile data	No profile required.
Objects accessed in read mode	Conference data (including Tracks and Topics).
Objects accessed in write mode	None.
Relevant usage scenarios	Author Registration, Conference Program Browsing.

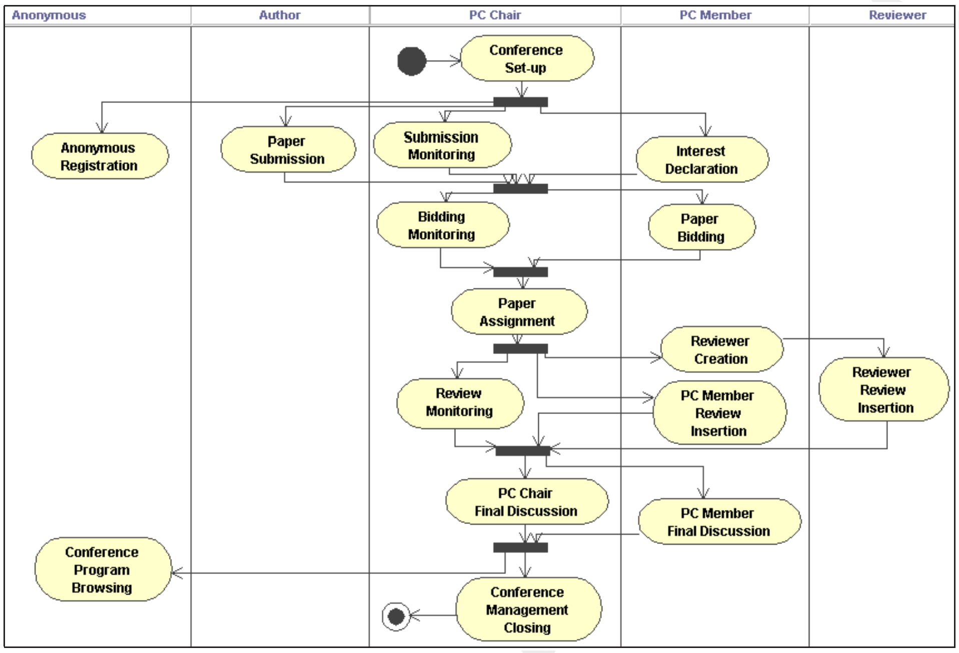


Figure 4. The CMA activity diagram.

- *Final Conference Program*, in which the PC Chair closes all the conference management activities and the conference program is made available to every (including anonymous) user.

It is worth noting that phase transitions are explicitly determined by the PC Chair, who monitors the process and checks if the state reached by some synchronization data is such that they can close the current phase. For example, the PC Chair closes the review phase only when all the reviews for the submitted papers have been inserted.

From the CMA activity diagram shown in Figure 4, some further interesting properties emerge, such as the fact that in some cases different users execute different tasks autonomously (e.g. paper submission by Authors, interest declarations by the PC Members), while in several other cases, tasks are executed collaboratively. For instance, while PC Members and Reviewers insert paper reviews, the PC Chair monitors the reviews' status (open or finished) to assess whether the review can be declared as closed. Moreover, in the final discussion phase, the PC Chair and the PC Members access the inserted reviews and together define the final conference program. This confers a collaborative nature to the CMA process.

Usage scenarios could also be further detailed through UML use case diagrams, which are able to emphasize interactions among different usage scenarios and between user groups and usage scenarios better. For the sake of brevity, further descriptions are not reported here. More details about the CMA functional analysis can, however, be found in [20].

3.4. Identification of core information objects

From the user group analysis and the functional analysis performed previously, the following information concepts emerge as central to conference management.

- *User* and *Group* are fundamental for representing data about the application users, their roles, which can be derived from the group they belong to and, consequently, their access rights over information objects and services.
- *Conference* data needs to be represented, with properties referring to the conference venue and important dates. The conference has also two components[§]: *Tracks* and *Subjects*.
- *Papers* represent the main information objects accessed and managed in the central phases of paper submission and review.
- Besides having a contact author, represented as an instance of the User entity, each paper can have one or more *Coauthors*, whose data must be represented for determining the occurrence of possible conflicts during paper assignment to PC Members.
- For each submitted paper, one or more *Reviews* are created during the Paper Review phase.

3.5. Identification of site views

As already discussed above, in the conference management process user group activities change along the whole process. In particular, the functional analysis has highlighted that different phases exist and that the transition between phases corresponds to an activity change for some group. Therefore, for each phase and for each involved user group, one site view must be defined, able to support the group activities in that phase. If a group activity consists of distinct and self-contained tasks, then the site view associated with this activity can be organized in areas, each one supporting a specific task. For example, the site view dedicated to PC Members during the review phase can be organized in two areas, one devoted to review insertion, the other one to reviewers creation.

Even groups not involved in a given phase must be provided with a site view. In CMA, all groups not active in a given phase are provided with a site view consisting of a single page with a message informing users that login is temporarily suspended. Also, a site view must be associated to the Anonymous user group, which allows both non-registered users to get general information and registered users to authenticate themselves and be redirected to the site view currently associated to their group. In CMA, a 'public' site view provides non-authenticated users with general information about the conference and with a login form for entering authentication data in case they are registered.

The association between user groups and the identified site views in CMA is summarized in Table VII.

[§]Components of information concepts correspond to complex and possibly multivalued properties, which are described in the data schema by means of entities.

Table VII. Association between user groups and site views in the different conference phases.

	PC Chair	PC Member	Author	Reviewer	Anonymous
Conference Set-up	Conference Set-up	—	—	—	—
Submission	Submission Monitoring	Interest Declaration	Paper Submission	—	Registration
Bidding	Bidding Monitoring	Paper Bidding	Login Suspended	—	Anonymous Login
Assignment	Paper Assignment	Login Suspended	Login Suspended	—	Login
Review	Review Monitoring	Review Insertion	Login Suspended	Insert Review	Login
Discussion	Final Discussion	Final Discussion	Login Suspended	Login Suspended	Login
Conference Program	Conference Management Closing	Login Suspended	Login Suspended	Login Suspended	Conference Program Browsing

4. CMA DATA DESIGN

Figure 5 shows the structural schema defined for the CMA case study. It is obtained by representing the core information objects identified during requirements analysis and associating them through relationships. Some derived attributes and relationships have been added to the schema, with the aim of enriching it with redundant entity properties and relationships, which can be computed from other elements in the schema.

The entity *Conference* is associated with one or more *Tracks* and one or more *Subjects*.

The entity *Paper*, being a core object managed and accessed by users of different groups for different purposes, has several relationships with the entity *User*. In particular:

- the *Submission* relationship associates each paper with a user representing the author that has submitted it;
- the *Review* relationship also associates each paper with one or more users representing the PC Members and the Reviewers that must review it;
- the relationship *ExplicitConflict* represents the situation in which, during the Bidding phase, a PC Member user or the PC Chair have raised an explicit conflict for one or more papers;
- the derived relationship *NonConflictualPaper* associates PC Member users with those papers that are not in conflict with them. It is derived according to a query which evaluates the coupling between instances of the two entities *User* and *Paper*, by checking the following conditions:

```
[User.Affiliation <> Paper.Submission.Affiliation]
[User.Affiliation <> Paper.Paper_Coauthor.Affiliation]
```

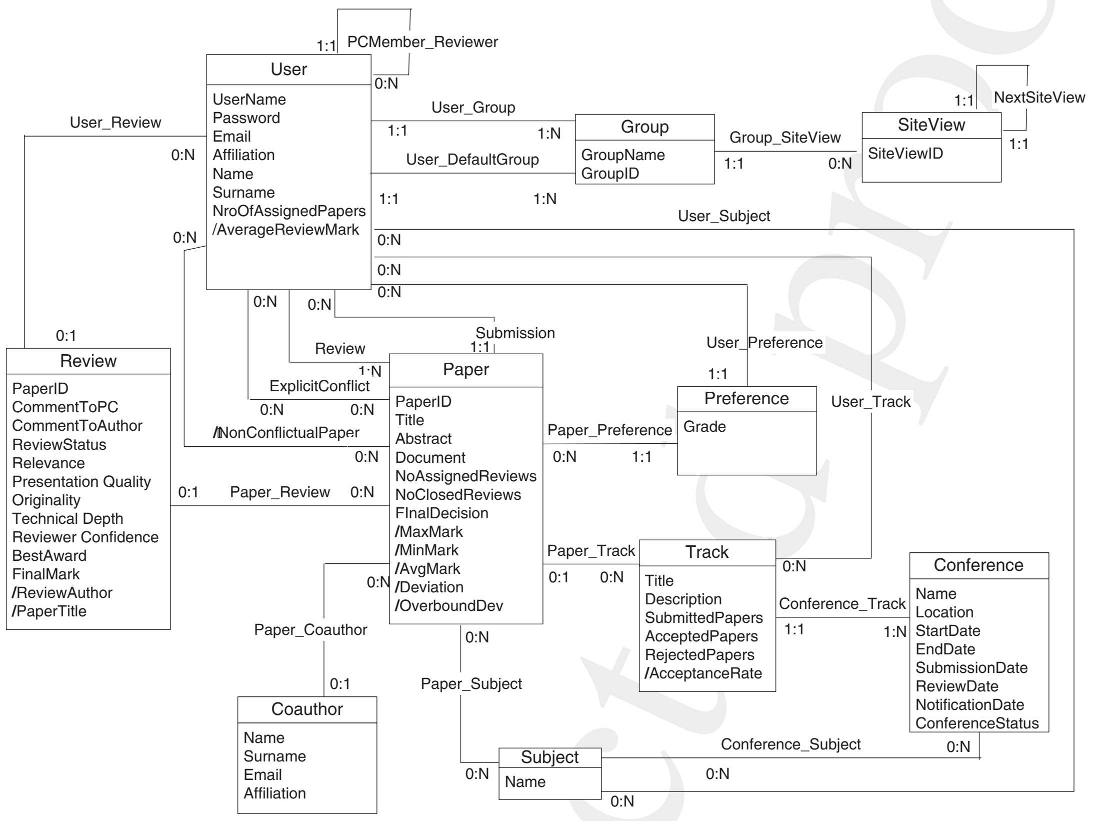


Figure 5. The CMA data schema. Slash characters in attribute or relationship names denotes derived data.

```
[User.Oid NOT IN Paper.ExplicitConflict.User.Oid]
[User.Oid NOT IN Paper.Review.User.Oid]
```

Each paper is also associated with a single *Track*, one or more *Subjects*, one or more *Reviews* entered by both Reviewers and PC Members and one or more *Coauthors*. Also, each paper is associated with one or more *Preferences*, expressed by PC Members during the bidding phase.

Two relationships associate each PC Member User with one or more tracks and to one or more subjects, thus representing preferences about tracks and subjects expressed by PC Members.

4.1. Group modelling

The user model, the activities to be performed by user groups in the different phases, and the activity evolution along the process are translated into three entities, *Group*, *User* and *SiteView*, and four relationships as follows.

- *Group* specifies the groups to which CMA users may belong.
- *User* specifies the single users that can access the application, independently from the group they belong to. Its properties constitute the user profile, i.e. the set of psychographic data necessary to manage CMA users, which emerged in the requirement analysis phase.
- Two relationships connect *User* and *Group*: the *User_Group* relationship specifies all the *Groups* a *User* can belong to; the *User_DefaultGroup* relationship specifies the default *Group* of each user, which is necessary to pick-up a group and serve the corresponding hypertext when a user logging into the application belongs to different groups.
- The relationship *PCMember_Reviewer*, recursively defined over the entity User, associates PC Member users with their Reviewers.
- *SiteView* specifies the site views accessible by the different groups. Each *SiteView* instance represents one distinct hypertext, which serves the activity of a given group in a specific phase.
- The relationship *Group_SiteView* maps a group to the specific site view that will be presented to users when they access the application as members of the group.
- The recursive relationship *NextSiteView*, defined over the entity *Site View*, associates a site view of a given group with the next site view to be presented to the same group, according to the sequence of associations between user groups and site views reported in Table VII. For instance, the PC Chair's *Conference Set-up* site view precedes the PC Chair's *SubmissionMonitoring* site view, which in turn precedes the PC Chair's *BiddingMonitoring* site view, and so on. This relationship is needed in order to control the site view change during phase transitions.

Once the data schema is defined and before proceeding with site view design, it is fundamental to identify entities and relationships in the data schema that correspond to synchronization objects in the different phase transitions across the whole process, and to shared objects supporting communication among users within different phases.

4.2. Identification of synchronization objects

In CMA, synchronization between user activities in different phases is determined by the availability of some information objects, to be produced by one or more user groups in a given phase. Based on the availability of such objects, the PC Chair can declare the current phase closed; thus the following phase starts and some other groups can consume the information objects previously produced for performing their activities. Therefore, the CMA process goes on along cycles of production–consumption of synchronization data by different groups in different phases.

Going back to the activity diagram depicted in Figure 4, synchronization bars represent phase transitions in CMA and also indicate that some data have been produced, thus are available for the following phase. Table VIII details which information objects (whether entities, entity attributes or relationships) are produced at each synchronization point, as well as their Producer and Consumer groups.

4.3. Identification of shared objects

Besides synchronization, some form of communication must be ensured within a same phase among users belonging to a same group or to different groups. This is made possible by some *shared* data, i.e. data at least accessible (and in some cases modifiable) by all such users.

Table VIII. Synchronization objects.

Phase transition	Producer	Synchronization data	Consumer
Conference Set-up ⇒ Paper Submission	PC Chair	*Track* *Subject*	PC Member PC Member
Paper Submission ⇒ Paper Bidding	Author PC Member PC Member	*Paper* *Paper_Track* *Paper_Subject*	PC Member PC Chair PC Chair
Paper Bidding ⇒ Paper Assignment	PC Member	*Preference*	PC Chair
Paper Assignment ⇒ Paper Review	PC Chair	• *NoOfAssignedReviews* (attribute of the entity Paper). • *NoOfAssignedPapers* (attribute of the entity User). (These attributes help in determining, respectively, if each paper has reached the right number of PC Members who will review it and if each PC Member has been assigned with the right number of papers to review).	PC Member
Paper Review ⇒ Final Discussion	PC Member	*Review*	PC Chair, PC Member
Final Discussion ⇒ Conference Program Browsing	PC Chair	*Final Decision* (attribute of the entity Paper). (This attribute can assume values [*accepted*, *rejected*] and is necessary in order to determine the final conference program).	Anonymous User

In every conference management phase, the PC Chair needs to monitor the status of some objects (i.e. the synchronization objects reported in Table VIII), so as to assess if they have been completely and correctly produced and declare the phase transition. Therefore, the synchronization objects highlighted in Table VIII are also objects *shared* between their producer group and the PC Chair.

Some other shares occur in other specific phases:

- During the Paper Review phase, PC Members can access the reviews created by reviewers they have designated at any moment. They can also modify such reviews, once reviewers declare them as finished. Also, when a PC Member declares the review for a given paper as finished, they can also access the reviews authored by other PC Members for the same paper. Therefore, *Review* is an object shared between PC Members and Reviewers, as well as between different PC Members.
- In the Final Discussion phase, the participating users, i.e. PC Chair and PC Members, access all the reviews and discuss paper acceptance and rejection. Again, *Review* is a shared object.

5. CMA HYPERTEXT DESIGN

Once the structural schema has been defined, the design proceeds with hypertext specification. In the following, we describe a few pages taken from two site views, with the aim of showing how the adopted model-driven approach allows us to reach the following goals.

- *Supporting the association of users with their activities, based on the role played.* The association between user groups and current site views is represented at the data level, through a relationship between the entities *Group* and *SiteView*. This relationship is exploited at login time, for redirecting users to the right site view, depending on the group they belong to.
- *Supporting profile management.* The system must allow the creation of new users and their profiles to be entered or modified. For instance, the PC Chair creates the program committee and the PC Member creates their Reviewers. Also, Authors can register, thus creating their own profile.
- *Supporting personalization of contents and services.* Several application requirements impose that users access a subset of the application data, according to their profile. For instance, during the bidding phase PC Members see only those papers which refer to tracks and subjects they have marked as preferred and that do not feature any conflict of interest with their affiliation.
- *Supporting phase transition along the process and the correspondent variation of users activities and access rights.* Since both site views and users groups are represented as data, at the end of each phase the PC Chair can express the change of activities and access rights by modifying the binding between user groups and site views, i.e. by modifying a relationship within the data schema.

5.1. Supporting the association of users with their activities

CMA site views can be distinguished in two classes: *public* and *restricted*. The former (i.e. Anonymous' Registration, Anonymous' Login and Anonymous' Conference Program Browsing) are accessible by all users, without any control. All the other site views require the user to provide credentials for gaining access permission through a login operation.

When registered users log into the application using a public site view, the problem arises of deciding which restricted site view they must be forwarded to. A simple policy may route users to a default site view, based on the users' default group. This is possible thanks to the explicit representation of the mapping between groups and site views in the data schema. When a user logs in, the *User_DefaultGroup* relationship is used to determine the user's default group; then the relationship *Group_SiteView* identifies the current site view associated with the default group and the user is forwarded to the home page of this site view.

Figure 6 schematically represents the login process management in WebML: the Login page, included in the public site view, contains an entry unit whereby users can insert their credentials (for example, username and password). From the entry unit, a link activates the login operation, which looks up the user's default group and determines the destination site view. Then, the login operation redirects the user to the home page of the selected site view. If the credentials of the user are invalid, the login operation may route the user to an error page that may include an error message and a non-contextual link to the home page of the public site view.

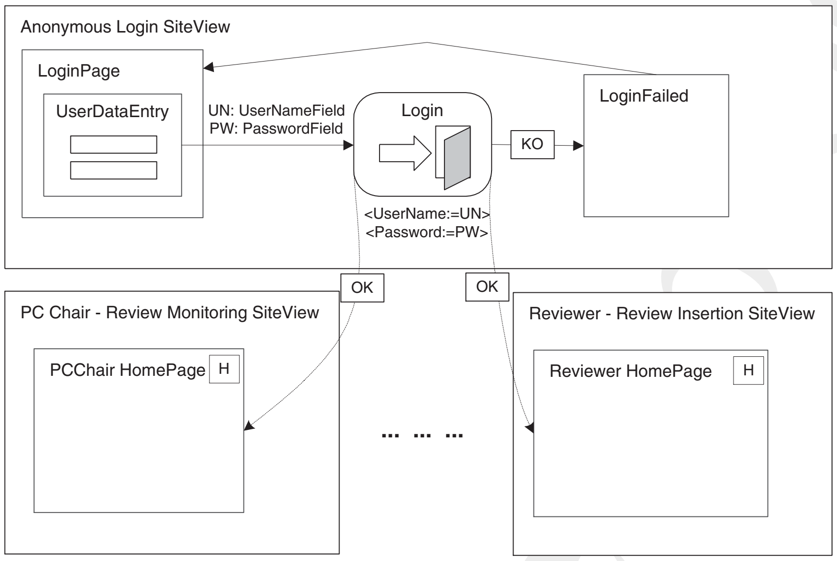


Figure 6. Login process, redirecting users to the site view currently associated with the group they belong to.

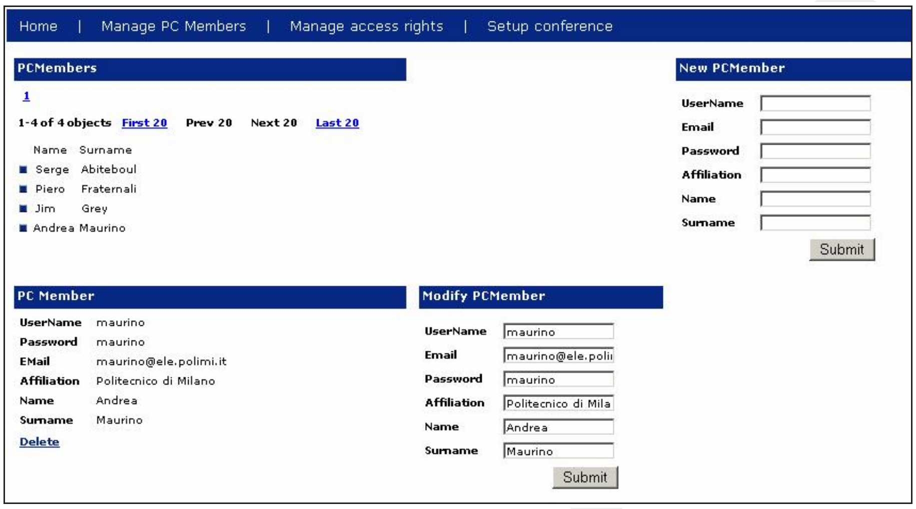
5.2. Supporting user profile management

Figure 7(a) represents the *Manage PCMembers* page, taken from the PC Chair's *Conference Set-up* site view. In this page, the PC Chair can pre-register new PC Members. Figure 7(b) shows the corresponding WebML schema. In this page, the OID of the 'PC Member' group is retrieved through the get unit *PCMember*. This OID has been set as a global parameter in another page.

The *PCMembers* index unit presents the list of all the PC Members (i.e. all the users belonging to the PC Member group). From it, a single PC Member can be selected and the corresponding data displayed through the *PCMember* data unit. The PC Chair can thus delete the selected PC Member by activating the *Delete PCMember* operation, or modify their data by collecting the new data (through the *Modify PCMember* entry unit) and activating a modify operation (*Modify PCMember* unit).

A second entry unit (*New PCMember*) allows the PC Chair to insert data about a new PC Member. These data feed an operation chain that creates a new user instance (*Create PCMember* operation) and then associates it with the PC Member Group (*UserToGroup* operation).

It is worth noting that a similar hypertext composition, based on the create and connect operation chain for creating a new user and connecting it to a group, is also adopted in the Home Page of the Anonymous' *Registration* site view, in which anonymous users can create their own profile and become registered Authors.



Home | Manage PC Members | Manage access rights | Setup conference

PCMembers

1

1-4 of 4 objects First 20 Prev 20 Next 20 Last 20

Name	Surname
Serge	Abiteboul
Piero	Fraternali
Jim	Grey
Andrea	Maurino

New PCMember

UserName	
Email	
Password	
Affiliation	
Name	
Surname	

Submit

PC Member

UserName	maurino
Password	maurino
EMail	maurino@ele.polimi.it
Affiliation	Politecnico di Milano
Name	Andrea
Surname	Maurino

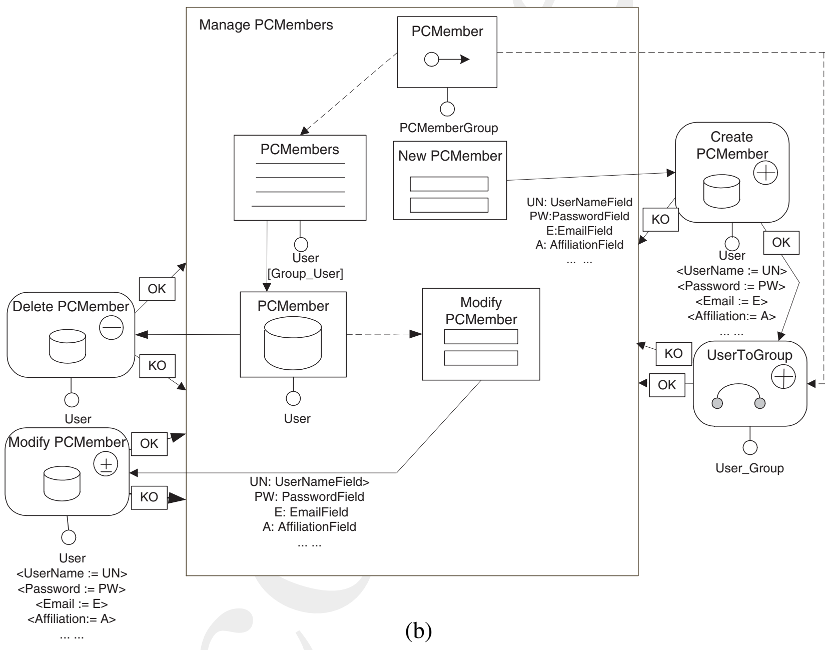
Delete

Modify PCMember

UserName	maurino
Email	maurino@ele.polii
Password	maurino
Affiliation	Politecnico di Mila
Name	Andrea
Surname	Maurino

Submit

(a)



(b)

Figure 7. The page Manage PCMembers (a) and its WebML specification (b).

5.3. Supporting personalization of contents and services

During the Paper Bidding phase, PC Members are asked to express their preferences on a subset of the submitted papers. The list of papers presented to each PC Member is personalized, taking into account whether the paper subjects and track match the preferences expressed by the PC Members and that papers are not in conflict with the PC Members affiliation.

Figure 8(a) shows the page *Declare interest or conflict of interest*, taken from the PC Members' *Paper Bidding* site view. It allows PC Members to explicitly indicate their interest on a personalized list of papers. Before expressing their interest, PC Members may also indicate the existence of a conflict for some of these papers; for example, although the authors' and PC Member's affiliations are different, the PC Member is somehow involved in the research work described in the paper. This action further personalizes the list of the papers.

Figure 8(b) shows the WebML specification of the same page. The get unit *Current User* supplies the OID of the user currently logged in. This OID is a global parameter, which is automatically set when a user logs into the application and enters the site view. The hierarchical index *Preferences for non conflictual papers* shows the preferences grades (i.e. instances of the entity *Preference*) assigned by the current PC Member to the set of papers (i.e. instances of the entity *Paper*) selected through the derived relationship *NonConflictualPaper*. Initially, the preference grades shown assume a default value (zero).

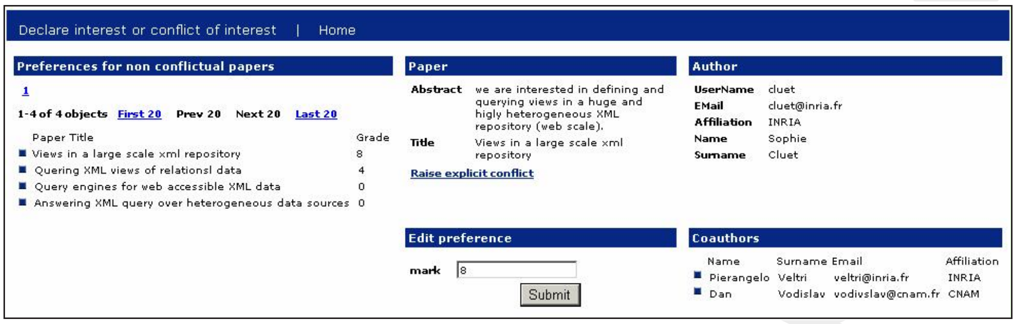
The PC Member can select a paper, getting more details about it (*Paper* data unit), data about its contact Author (*Author* data unit) and data about all its coauthors (*Coauthors* multidata unit); for the selected paper the PC Member can input a new preference grade by inserting data through a form (*Edit Preference* entry unit). The form 'Submit' button then activates the *Modify Preference* operation, which updates the preference value in the database.

Alternatively, to express a preference grade for the selected paper, the PC Member can also raise an explicit conflict. This is done by following a link that activates a connect unit (*Explicit Conflict*) populating the relationship *ExplictConflict*. If the operation is successfully executed, the content of the *Preferences for non conflictual papers* index unit is updated, so that it no longer shows papers in conflict.

5.4. Managing phase transitions

Figure 9 illustrates the *Manage access rights* page, through which the PC Chair makes the conference status evolve. This page, which is available in all the PC Chair site views, allows the PC Chair to manage phase transition, by changing the association between groups and current site views. In this way, in each phase groups are provided with a site view supporting their current activity.

The page includes the list of all the groups (*Group Index* unit), from which the PC Chair can select a group and see its details (*Selected Group* data unit) and its current site view (*Current SiteView* data unit). On the basis of the relationship *NextSiteView*, the *Next SiteView* data unit then shows the successive site view to be associated with the selected group, in accordance with the associations between groups and site views and site view transitions described in Table VII. At this point, the PC Chair can trigger the site view transition for the selected group, by activating the connect operation *Change SiteView*, that overrides the previous relationship instance, expressing a new association between the selected group and site views. As a result, the next login by a user in the group leads to the home page of the new associated site view.



Declare interest or conflict of interest | Home

Preferences for non conflictual papers

1

1-4 of 4 objects First 20 **Prev 20** **Next 20** Last 20

Paper Title	Grade
Views in a large scale xml repository	8
Quering XML views of relationsl data	4
Query engines for web accessible XML data	0
Answering XML query over heterogeneous data sources	0

Paper

Abstract	we are interested in defining and querying views in a huge and higly heterogeneous XML repository (web scale).
Title	Views in a large scale xml repository

Raise explicit conflict

Author

UserName	cluet
EMail	cluet@inria.fr
Affiliation	INRIA
Name	Sophie
Surname	Cluet

Edit preference

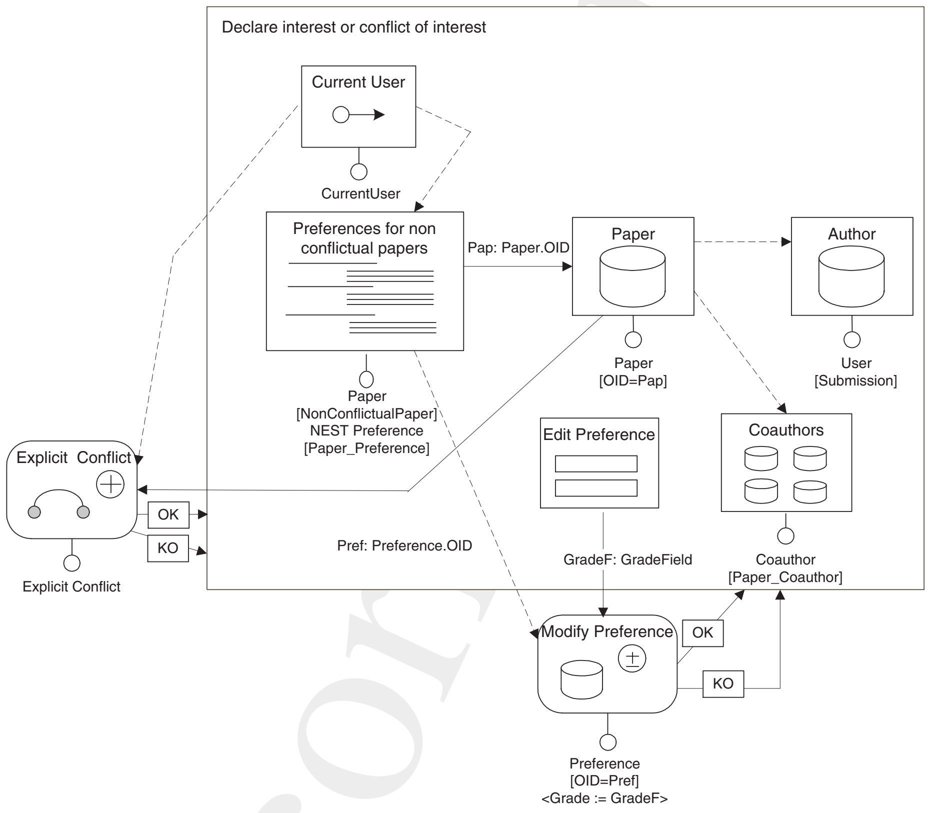
mark 8

Submit

Coauthors

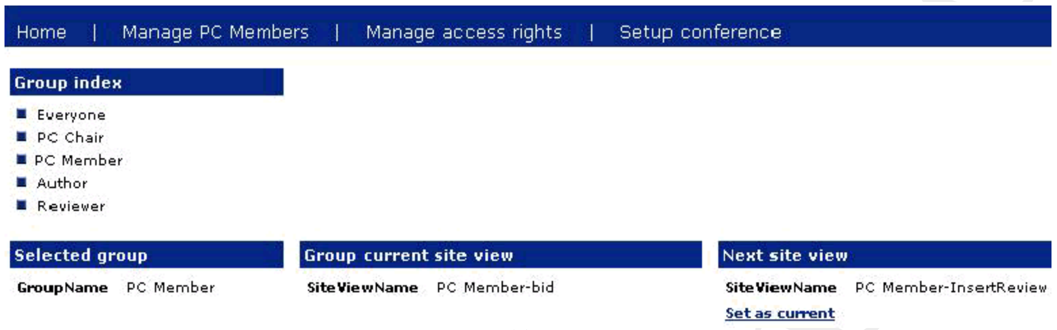
Name	Surname	Email	Affiliation
Pierangelo	Veltri	veltri@inria.fr	INRIA
Dan	Vodislav	vodivslav@cnam.fr	CNAM

(a)


Declare interest or conflict of interest
Current User
CurrentUser
Preferences for non conflictual papers
Pap: Paper.OID
Paper
Author
Paper [OID=Pap]
User [Submission]
Paper [NonConflictualPaper] NEST Preference [Paper_Preference]
Explicit Conflict
OK
KO
Explicit Conflict
Edit Preference
Coauthors
Pref: Preference.OID
GradeF: GradeField
Coauthor [Paper_Coauthor]
Modify Preference
OK
KO
Preference [OID=Pref] <Grade := GradeF>

(b)

Figure 8. The page Declare interest or conflict of interest (a) and its WebML specification (b).



Home | Manage PC Members | Manage access rights | Setup conference

Group index

- Everyone
- PC Chair
- PC Member
- Author
- Reviewer

Selected group

GroupName PC Member

Group current site view

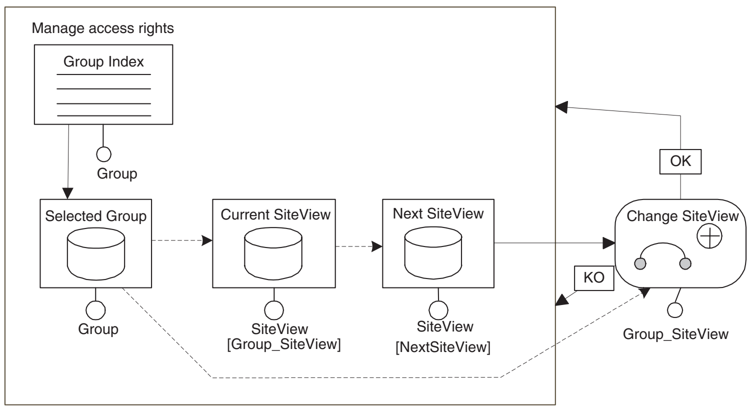
SiteViewName PC Member-bid

Next site view

SiteViewName PC Member-InsertReview

Set as current

(a)



(b)

Figure 9. Page for modifying the mapping of groups to site views (a) and its WebML specification (b).

6. CMA IMPLEMENTATION

The full implementation of CMA, covering all the application requirements, has been carried out by using WebRatio, a tool suite that offers automatic support for translating WebML conceptual schemas into a running application. The entities and relationships in the high-level data schema have been automatically mapped on top of a SQL server data source and JSP 1.1 templates have been generated, thus obtaining an application deployable on a Java2EE platform. Table IX summarizes the application size, in terms of required site views, number of Web pages, number of relational tables and views, and size of the JSP code automatically generated by the tool.

As extensively discussed in [12], the architecture of the generated Web application complies with the MVC 2 architecture [21–23], in which a partitioning of responsibility is operated among the Model, View and Controller components. According to this architecture, each JSP template is a

Table IX. Dimensions of the CMA implementation.

Number of site views	16
Number of JSP pages	148
Number of relational tables	16
Number of relational views for derived data	30
Approximate dimension of generated JSP code (kbytes)	950
Number of SQL views manually retouched*	6

*Hand coding of SQL view is only due to optimization reasons.

View component, which generates the rendition of a page in HTML, on the basis of the content computed through *page actions* and *page services*. The former are instances of Java classes, able to extract the input from the HTTP request of a page and call the corresponding page service, passing to it the required parameters. The page service is a business function, which computes the page and updates the objects that represent the state of the application in the Model. The binding among user's HTTP requests, page actions and page views is obtained through an *action mapping*, declared in a configuration file implementing the Controller.

Similarly to the partitioning operated for pages, WebML content units map into *unit services*, i.e. Java classes responsible for unit content computation, and *custom tags*, transforming the content produced by unit services into HTML. Also, operations map into *operation services* and into an action mapping in the Controller configuration file.

7. LESSONS LEARNED

The development of the CMA application reported here demonstrates the applicability to the development of collaborative applications of the model-driven design methods advocated for general-purpose Web applications.

As our experience shows, model-driven design captures all the essential ingredients of a collaborative application at the conceptual level:

- *user-oriented requirements*, in terms of the process participants, their roles and their access rights over application objects and activities;
- *functional requirements*, i.e. the Web interfaces and the business logic allowing different users to perform their activities, while cooperating for pursuing a common goal;
- *cooperation requirements*, i.e. the constraints over the accomplishment of the different activities in the process, which guarantee collaboration and synchronization among users.

User-related requirements are explicitly modelled in the structure schema, which permits the expression of access rights and content ownership rules in a declarative manner.

Functional requirements are conveyed by the hypertext modelling language, which allows the declaration of pages that constitute the front-end whereby users perform the individual activities.

Collaboration requirements are captured using both the data and the hypertext model in a coordinated manner as follows.

- The data model permits the declaration of shared data structures and of state information necessary for coordinating the execution of activities by different users (or user groups).
- The hypertext model expresses the interfaces for executing individual activities as sets of correlated pages and exploits links to specify the required navigation among the sequence of pages supporting an activity. By conditioning the content and linkage of pages based on these control data, it is possible to express a variety of collaboration requirements at the hypertext level, even in the absence of an *ad hoc* workflow management system.

As the CMA application shows, a modelling language for Web applications is able to express both push-based and pull-based collaboration models.

- In a pull-based model, the user is the source of events that make the collaboration progress. Pull-based systems are the natural target of Web-oriented conceptual models, since Web-based applications essentially obey to the pull paradigm dictated by the HTTP protocol.
- In a push-based model, the system proactively pushes events to the user, who responds to them and makes the collaboration evolve. WebML is able to express a limited form of push requirements, in which the system reacts to some events caused by the pull-based interaction of some user (e.g. the PC chair) and produces other events that are pushed to other users (e.g. to the PC members, which are notified that a phase of the conference management is terminated and thus access to a different set of pages).

In essence, Web application conceptual modelling has proven effective in formally capturing a substantial fraction of the requirements of a typical collaborative application. This fact has some important consequences on the development process of this class of systems, because all the benefits of conceptual modelling, already sufficiently proven in the fields of data and hypertext modelling, carry over to the realm of collaborative Web-enabled applications, without a significant loss of application features.

Conceptual modelling can therefore offer an alternative solution to developers with respect to the adoption of dedicated collaborative application platforms or workflow management systems (WfMSs). Very often, such systems lack a global perspective over process definition and operate at the implementation level. In some *ad hoc* groupware systems (see, for example, IBM Lotus Notes [24] or Microsoft Sharepoint [25]) the definition of the activities to be tackled by process participants and possible collaboration requirements are obtained through the concatenation of some components supporting typical collaboration activities (e.g. chat, message boards, document sharing and so on). It follows that the logic underlying the process is defined through parameter setting and component composition and it is implicitly represented through scripts embedded in an unstructured source code. At the other extreme, WfMSs greatly support the process design, by offering high-level visual formalisms (see, for example, [26]). However, process specification mainly captures the workflow enactment; functional requirements and the corresponding user activities are often treated as black boxes and are supported by external resources, which are invoked and integrated within the WfMS through programming interfaces. The workflow engine just monitors and coordinates the process execution.

With respect to the above specialized systems, conceptual modelling introduces a different design paradigm, which does not require dedicated software support (generally conceptual specifications are automatically translated into template-based languages, such as JSP or ASP.NET) and offers the following advantages.

- Differently from WfMSs, conceptual modelling not only captures the enactment of workflows, but also the application data and the interfaces for performing the activities. In doing so, it offers a uniform high-level specification notation, spanning all the aspects of a collaborative application. From these high-level specifications, the application code for enacting the workflow, the data structures for storing the application data and the interfaces for performing the activities can be generated automatically or semi-automatically, as already possible with commercial products [13,27,28].
- Differently from development platforms and collaborative software development kits, which mostly focus on the implementation-level productivity, conceptual modelling leverages a platform-independent and abstract view of requirements and design, in which all the aspects of a collaborative application are expressed in a declarative manner. This discipline is not only beneficial in conjunction with automatic code generation, but also improves the quality of application design and documentation and facilitates maintenance, because changes in user requirements, functional requirements and collaboration requirements can be applied in an orthogonal way to the affected part of the conceptual model and then automatically propagated to the implementation.

8. CONCLUSIONS

In this paper, we have illustrated a model-driven approach to the development of collaborative applications on the Web, by specifically focusing on those design activities that are most relevant to cooperative applications, including user and user group modelling, user profile management, identification of synchronization and shared objects, association between users and their activities, content and service personalization, and phase transition management.

Although we have demonstrated that a conceptual model for Web applications can be used, as it is, for designing collaborative Web-centric applications, other ingredients may be required for designing a full-fledge WfMS on the Web. In particular, this paper has shown that collaborative activities within the conference management can be delivered on the Web just relying on the HTTP protocol, which is based on a 'pull' paradigm, where the application execution is driven by the user intervention through specific actions (basically navigation of links). We have also shown that, under certain assumptions, WebML is able to express simple push requirements. However, many WfMSs support event notification according to a more complex paradigm: for example, alert messages are automatically sent to users when given events occur. In WebML, such a proactive behaviour would be possible to model by adding specialized operations (for example, a *SendMail* operation, which could receive as its input a list of messages to be sent to users). We are currently modelling such an extension for WebML in the context of the development of a complex workflow application, aimed at supporting the workflow among a large computer company and its resellers and partners.

Another functional requirement typically covered by WfMSs is the ability to respond to asynchronous, external events (such as deadlines, errors, exceptions and so on). Generally, the Web paradigm does not support this requirement, because the only event that can be trapped is the navigation along a link. A possible way to work around this is to create 'guards' in the system (such as event monitors or time managers) or in the database (such as triggers), devoted to detect asynchronous events and react to them. In general, existing Web-design models and methods could be easily extended with a few primitives (e.g. conceptual-level business rules, asynchronous messaging, exception handling, conditional constructs for expressing fork or join conditions), making the modelling of Web-based workflows more effective. We are currently working on extending WebML in this direction [29].

A further extension required by WfMS is the ability to deal with software agents. CMA and, in general, collaborative applications on the Web are accessed by human agents only; instead, a general WfMS must be able to manage both human and software agents. This is a new frontier for the research of the Web community. Recently the W3C consortium has promoted a 'Semantic Web' workgroup, whose goal is 'an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation' [30]. At the same time, new research is dedicated to Web services, aimed at facilitating the communication of human and software agents with remote computing systems. Our current work is also devoted to modelling the integration of Web services within Web applications through primitives for interacting with operations provided by Web services, data transfer capabilities between the application data sources and messages to be exchanged with services, and mechanisms supporting synchronous and asynchronous bi-directional messaging [29].

REFERENCES

1. Atzeni P, Mecca G, Merialdo P. Data-intensive Web sites: Design and maintenance. *World Wide Web* 2001; **4**:21–47.
2. De Troyer O, Leune C. WSDM: A user centered design method for Web sites. *Proceedings of the WWW7 Conference*, 1998; 85–94.
3. Garzotto F, Paolini P, Schwabe D. HDM—a model-based approach to hypertext application design. *ACM Transactions on Information Systems* 1993; **11**(1):1–26.
4. Fernandez M, Florescu D, Kang J, Levy A, Suciu D. Catching the boat with Strudel: Experiences with a Web-site management system. *SIGMOD Records* 1998; **27**(2):414–425.
5. Fraternali P, Paolini P. A conceptual model and a tool environment for developing more scalable and dynamic Web applications. *Proceedings of EDBT 1998*, Schek HJ, Saltor F, Ramos I, Alonso G (eds.). Springer: Berlin, 1998; 421–435.
6. Isakowitz T, Stohr EA, Balasubramanian P. RMM: A methodology for structured hypermedia design. *Communications of the ACM* 1995; **38**(8):34–43.
7. Schwabe D, Rossi G. The object-oriented hypermedia design model. *Communications of the ACM* 1995; **38**(8).
8. Pastor O (ed.). *Proceedings of the IWWOST'01 Workshop—First International Workshop on Web-Oriented Software Technologies*, Spain, 2001.
9. Nierstrasz O. Identify the Champion. *Pattern Languages of Program Design*, vol. 4, Harrison N, Foote B, Tohnert H (eds.). Addison-Wesley: Reading, MA, 2000; 539–556.
10. Conallen J. *Building Web Applications with UML* (*Object Technology Series*). Addison-Wesley: Reading, MA, 2000.
11. Ceri S, Fraternali P, Bongio A. Web Modeling Language (WebML): A modeling language for designing Web sites. *Computer Networks* 2000; **3**(1–6):137–157.
12. Ceri S, Fraternali P, Bongio A, Brambilla M, Comai S, Matera M. *Designing Data-Intensive Web Applications*. Morgan Kaufmann, 2002.
13. WebRatio—Site Development Studio. http://www.webratio.com [20 December 2002].
14. Boehm B. A spiral model of software development and enhancement. *IEEE Computer* 1988; **21**(5):61–72.
15. Beck K. Embracing change with extreme programming. *IEEE Computer* 1999; **32**(10):70–77.

16. Kruchten P. *The Rational Unified Process: An Introduction*. Addison-Wesley: Reading, MA, 1999.
17. Workflow Management Coalition. The Workflow Reference Model. http://www.wfmc.org [20 December 2002].
18. Booch G, Rumbaugh J, Jacobson I. *The Unified Modeling Language User Guide* (*Object Technology Series*). Addison-Wesley: Reading, MA, 1999.
19. Bongio A, Ceri S, Fraternali P, Maurino A. Modeling data entry and operations in WebML. *Proceedings of WebDB'00 (Selected Papers)* (*Lecture Notes in Computer Science*), Vossen G, Suciu D (eds.). Springer: Berlin, 2000; 201–214.
20. Ceri S, Fraternali P, Matera M, Maurino A. Designing multi-role, collaborative Web Sites with WebML: A conference management system case study. *Proceedings of the IWWOST'01 Workshop—First International Workshop on Web-Oriented Software Technologies*, Spain, 2001; 130–152.
21. Gamma E, Helm R, Johnson R, Vlissedes J. *Design Patterns—Elements of Reusable Object Oriented Software*. Addison-Wesley: Reading, MA, 1995.
22. Alur D, Crupi J, Malks D. *Core J2EE Patterns: Best Practices and Design Strategies*. Prentice-Hall: Englewood Cliffs, NJ, 2001.
23. Davis M. Struts—an open-source MVC implementation. http://www-106ibm.com/developerworks/library/j-struts/?n-j-2151 [20 December 2002].
24. IBM. Lotus Notes. http://www.lotus.com/products/ [20 December 2002].
25. Microsoft Corporation. Sharepoint Technologies. http://www.microsoft.com/sharepoint/ [20 December 2002].
26. Casati F, Shan MC. Process automation as the foundation for e-business. *Proceedings of VLDB 2000*, Cairo, Egypt, September 2000; 688–691.
27. CodeCharge. http://www.codecharge.com [20 December 2002].
28. Oracle. Oracle9i Designer: Technical Overview. http://www.oracle.com [20 December 2002].
29. Brambilla M, Ceri S, Comai S, Fraternali P, Manolescu I. Model-driven specification of Web services composition and integration with traditional Web applications. *IEEE Data Engineering Bulletin* 2002; **25**(4).
30. Berners-Lee T, Hendler J, Lassila O. The semantic Web. *Scientific American* 2001; **5**(1).

Annotations from spe523.pdf

Page 14

Annotation 1
Au: line 2
Please clarify sense?

Page 31

Annotation 1
Au:
Please supply publisher details for references [2,8,20,26].
Please supply page numbers for references [7,29,30].
Please supply volume number
for reference [19].