Carrying on the e-Learning process with a Workflow Management Engine

Mirko Cesarini
Politecnico di Milano
Dipartimento di Elettronica e Informazione
Piazza Leonardo da Vinci, 32
I 20133 Milano, Italy
cesarini@elet.polimi.it

Mattia Monga
Università degli Studi di Milano
Dip. Informatica e Comunicazione
Via Comelico 39/41
I 20135 Milano, Italy
mattia.monga@unimi.it

Roberto Tedesco
Politecnico di Milano
Dipartimento di Elettronica e Informazione
Piazza Leonardo da Vinci, 32
I 20133 Milano, Italy
tedesco@elet.polimi.it

ABSTRACT

In recent years e-learning systems have promised to change the way people learn. However open issues still remain, in particular actual e-learning environments do not consider learning activities as part of the process of learning. Thus, it is not possible to define structured courses and specify precise learning paths apt to guide learners through learning materials. In our approach, we define courses as workflows. By so doing we can exploit powerful procedural rules in order to define precise while flexible learning paths. In this paper we present Virtual Campus, a research project sponsored by Microsoft Research (UK) and developed at Politecnico di Milano, that exploits a workflow engine to enacts the fruition of structured courses. Our platform provides both an authoring and a fruition environment. The former allows teachers to define and customize learning paths, publishing them as workflows. The fruition environment enacts the workflows and guides learners through the related learning paths. We also describe an experience in using the platform during a Software Engineering course composed by heterogeneous activities (lectures, studying activities, cooperative sessions of work, and exams).

Categories and Subject Descriptors

H.4.1 [**Information Systems Applications**]: Office Automation—*workflow management*; H.5.4 [**Information Interfaces And Presentation**]: Hypertext/Hypermedia—*architectures, navigation*; K.3.1 [**Computers And Education**]: Computer Uses in Education—*computer-managed instruction*

Keywords

Workflow Mgmt Systems, E-Learning, Learning Objects

1. INTRODUCTION

In recent years new ways of teaching and learning that leverage on Internet technologies have been investigated and actively used. The term e-learning is frequently used to address educational activities supported by the use of computers and the Internet communication facilities.

A number of computer-supported learning environments (called Learning Management Systems, LMS) are used by instructors to deliver teaching material to students. Roughly speaking, all of them provide three main features to their customers: (i) A repository of learning materials (sometimes called *learning objects* [8]), organized according to educational needs, in order to foster reuse and information sharing. (ii) Communication facilities, that range from traditional e-mail forums to sophisticated video-conference widgets. (iii) A platform able to deliver content and communication programs over the Internet, preferably exploiting the web browser as the user client.

Nevertheless, computer supported learning environments should help teachers to recover data about on-going learning activities and adapt courses to specific needs of participants. To this end, the availability of a repository of learning data is not enough. The process of learning should be made explicit in the system. Thus, every user carries on her/his own learning process, adapted to her/his profile and controlled by her/his run-time behaviour, and possibly exchanging data with other learners and teachers. In other words the teaching material and activities proposed to Alice, should be different from the ones proposed to Bob, according to their role (student, teacher, etc.), their context (country, undergraduate course, presence of other actors, etc.), their skills and guaranteed effort, their results in previous activities. In addition, the e-learning process as whole should be measurable, in order to be able to asses the performances of every actor involved. In current systems, this is difficult to achieve because the flow of actors activities during the learning process is not explicit stated, nor usually recorded. Moreover, cooperation is often limited to communication: instructors and students exchange messages both in synchronous mode (chatting) and in asynchronous mode (e-mail, bulletin boards). This is quite far from the true cooperation needed to accomplish learning objectives. Ideally, an e-learning platform should be able to share processes and work data, not just messages.

In this paper we propose a novel architecture for e-learning management systems, where the learners' fruition activities are driven by a workflow engine. Workflow engines are programs that drive the activity of people involved in business processes [1]. A set or rules describing the process is coded into a custom programming language and given to the workflow engine, which will force the people involved in the process to remain compliant to the given rules and it will enable users to share artifacts. A business process can be very complicated and consist of several steps whose order is not strictly fixed, as result there could be many valid execution paths, depending of the circumstances. A person involved in a business process has to perform step by step the activities suggested by the workflow engine. When a branch appears in the path, the user selects a way and the workflow will acts consequently. The whole instructional process can be modeled as a business process [3] and enacted by workflow engines. In this paper we discuss how the use of a workflow engine can help in building a e-learning platform. In our system learning data are enriched by static and dynamic dependencies among topics and the system has the notion of the flow of learning activities. The e-learning process becomes a first-class object of the system and we are able to collect data about actors' behaviour, enabling an assessment of the effetciveness of learning activities. Furthermore, processes make explicit the idea of collaboration toward a common goal. As a trivial example consider the collaboration needed among teachers and students during an examination session: students need the work of teachers to begin (the text of the exam), and teachers can perform grading only after students have produced some artifact to evaluate.

With respect to the IEEE LTSA draft standard [7], our work can be seen as a part of the "coach" abstract process, which is in charge to drive the learners' fruition processes. Moreover, by exploiting a workflow as a sequencing engine, we give learners the opportunity to follow different learning paths through the set of learning materials. For this reason our platform could be classified as an "adaptive hypermedia system". In particular, with respect to the classification proposed in [6], our platform has the same properties as an "adaptive link generation" system, since the learning process is web-based and the navigation is driven by the past learners' activities.

This paper describes the design and the problems we faced while building a e-learning management system based on a commercial workflow engine and it is structured as follows: In section 2 we introduce the workflow management systems and show how to exploit them into e-learning platforms. Section 3 shows the architecture of our system. Section 4 illustrates an experiment involving students attending a CS course with our platform. Section 5 presents some related works. Finally, section 6 draws some conclusions and outlines directions for future work.

2. SUPPORTING E-LEARNING WITH WORKFLOW MANAGEMENT SYSTEMS

2.1 Workflows

Workflows are concerned with the automation of procedures where documents, information or tasks are passed between actors according to a defined set of rules to achieve an overall business goal. Workflow management systems (WFMS) [2] are of widespread use in large firms. Suppose there is a company where people travel a lot. Every worker before starting a business journey must have it approved by following some formal steps: (i) A document containing a short description of the journey, its motivation and a estimate of the expenditure must be sent to the worker's boss. (ii) If the boss approves the journey, the boss sends the document to the administrative staff to check if there is enough money in budget to cover the expenses. (iii) If the administrative staff approves the journey, the worker proposing the journey must contact the travel agency to book flights, hotels and what else is needed. (iv) At the end, all the invoices collected during the journey must be delivered to the administrative office. The whole procedure can be automated using a workflow management system. Every employee submit the short description of the journey to the WFMS, the WFMS deliver the journey request to the attention of the employee's boss and to the attention of the administrative staff, collect their approvals or refuses and notify the employee. Finally, if the journey is approved at the end of the travel, as the employee will come back to office, the WFMS will remember to send the invoices collected to the administration.

The process just described constitutes a workflow. The employees, bosses, administrative staff are the participants of the workflow, the rules just described are the procedural rules and they are coded into the process definition given to the WFMS.

WFMS are systems that completely define, manage, and execute workflows through the execution of software whose order of execution is driven by a computer representation of the workflow logic (*process definition*). The *workflow enactment service* is the component of the system that has the responsibility of creating, managing and executing workflow instances, by using one or more *workflow engines* that provide the run time environment. Other components may interface to this service via the WFMS API. These engines are able to interpret descriptions of business processes (written in an ad-hoc process definition language) creating running process instances that can be activated, suspended and terminated. Moreover, they are able to interact with external applications, driving their execution with workflow control data and workflow relevant data. Supervisors can administer and audit the system operating on the workflow enactment service.

Many individual process instances of the same workflow may run concurrently (every one corresponding to a process enactment), as many instances of a class can be present in a program at runtime. Each process instance (or workflow "case") has its own state and is associated with a specific set of participants, documents and resources. That allows a WFMS to manage concurrently different processes of the same type involving different people. In our example the WFMS can manage many employees' requests concurrently, everyone with its own state (there could be processes just started with the proposal to the boss and processes waiting for travel invoices).

Communication among process activities is obtained by

[1]In [2] a business process is defined as: a set of one or more linked procedures or activities which collectively realise a business objective or policy goal, normally within the context of an organisational structure defining functional roles and relationships.

sharing data files. Such data files are manipulated directly (and only) by the invoked applications, although the workflow engines may be responsible for transferring such data between applications (if necessary), as different applications are invoked at different activity points within the workflow process. Where process navigation decisions -or other control operations within the workflow engine- are based on data generated or updated by invoked applications, such data is accessible to the workflow engine. This is the only case of application data accessible to the workflow engine. With the process definition language it is possible to express which *artifacts* will be exchanged among activities and often the creation of artifacts itself drives the progress of execution.

Moreover, each user of the system has a work-list that presents to her/him a set of (manual or automatic) activities that have to be performed in order to satisfy the needs for collaboration that the system made emerge. The workflow enactment service may be considered as a state transition machine where individual process or activity instances change states in response to external events (e.g. completion of an activity) or to specific control decisions taken by a workflow engine (e.g. stepping to the next activity within a process). Transitions between states take place in response to particular commands issued to workflow engines; transitions between certain states will also take place as a result of transition conditions within the process definition being met (e.g. as the result of an external event, or time or data dependent condition, etc).

2.2 The learning process as a workflow

As previously seen a workflow is made by participants, actions, documents and a set of procedural rules that, once coded in a process definition, can be used to drive the workflow enactment. Essentially, the participants in a learning environments are students and teachers (teachers can play an active role within an e-learning process), the actions are the learning activities required to students or the correcting activities required to teachers, the documents are the artifact that students can produce either in laboratories or doing the homeworks or taking an examination. The procedural rules are relationship among the topics of a course (see Fig. 1): (a) Topic B can be studied only after topic A. (b) Topic C must be studied in alternative to topic D. (c) Topic E and F must be studied one after the other, but the order is not specified. (d) Topic K is an optional learning material. After the fruition of topic G, learners decide whether to exploit topic K or not. In both cases the last topic to study is H.

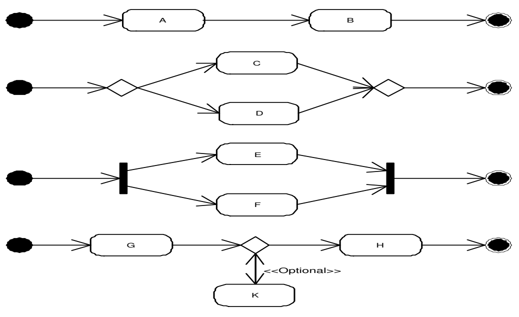

Figure 1: Procedural rules.

To fully apply a WFMS to an educational scenario, there is a synchronization problem to solve. Interaction between people and WFMS is document based. In the example of the travel approbation, the employee starts the approbation process by submitting a document, the same document is sent to the employee's boss to notify him that the travel must be evaluated. But in an educational scenario, the document based synchronization is not enough. A student can be notified about the next topic to study by giving her/him the course notes, or the video record of the lesson. But which document shall the student produce to inform the WFMS that she/he has finished to study a topic? The student can be asked to produce a document that will be evaluated by a test. But non all topics in a course should be tested, and often tests should be taken only after a set of topics has been learned. To address this issue we have enriched the behavior of the WFMS by adding the possibility to consume *events*. When a student finish studying a topic or ends taking a test, an event is produced and the WFMS is notified. Such notification for the WFMS is like receiving a document, and means that an activity has been completed and the student can go on with learning. This issue will be covered with more details in the next section, where the whole architecture will be outlined by showing an implemented prototype.

3. SYSTEM ARCHITECTURE

In this section we will give a short description of our prototype implementing an e-learning system built upon a WFMS. The system follows a client/server architecture: the client is used during an e-learning session by students; the server contains the WFMS, a web server, a collection of Web Services, and other useful tools. The architecture is completed by an authoring environment available only to teachers that can help them in preparing the process descriptions of courses.

3.1 The Client

The client is web based, in order to foster portability and usability of the application. There are no special requirements about the computer to be used to run the web browser (just the availability of a network connection to the server). The students simply connect to an URL and make a login. The bandwidth required is very low and a dial-up connection is enough to use the system, although a broadband connection could help in receiving large multimedia contents. The topics to study are proposed to the students by presenting them study material: course notes, slides and sometimes video-recorded lessons. All the material is presented through dynamic HTML pages. There is no limit about the format of the published material, the only requirement is that a web browser should be able to display it, although in the worst case a material can be downloaded and then viewed off-line with the appropriate application. The latter case is less and less probably because nowadays many web browsers can be customized with plug-ins to display new kind of contents. In our experiments most of the content were MS PowerPoint files, PDF files and video streams.

After a student has downloaded and studied the content, she/he can press a button to get the next learning material. By pressing the button an event is generated and notified to the server, which will create a new web page containing the new topic to study, according to the course described in the process definition that the WFMS behind the server is enacting. When the process definition requires the student to choose among two or more subjects, or studying an optional subject, a question page is generated and presented to the student, which can communicate her/his decision by selecting one of the proposed alternatives and finally pressing a confirm button. Once the button is pressed an event is generated and notified to the server. Accordingly to the student choice the server will present her/him the next topic to study.

3.2 The Server

The server side of the application is a web server coupled with a WFMS. The web server used in the prototype is Microsoft Internet Information Server (hereafter IIS), while for the WFMS we have chosen Microsoft BizTalk [5]. IIS is the web server which BizTalk has been designed to work with. BizTalk is an orchestration engine, a WFMS designed to coordinate computer process activities rather than human activities [2]. A traditional document-centered coordination is not enough for our scenario. Instead the coordination of computational processes, achieved by an orchestration engine, best suit our needs. In fact, in our scenario, the tasks a student performs can be seen as computational processes offered by an e-learning system: to retrieve a file containing a topic to study, to give an examination, to offer an optional topic to study, etc. BizTalk orchestrates the execution of these tasks according to the process definition describing the course: i.e. learn topic A, than learn topic B, than learn either topic C1 or topic C2, than take an examination. In the previous example, BizTalk would allow a student to get material about topic B only after topic A completion. Another advantage of BizTalk is that every service offered by third parties can be orchestrated as an internal service (the only requirement is that the service to be available as Web Service or as URL). For example the process definition of a course can contain a web-based test offered by an external authority (i.e. the TOEFL test), or contain a link to videos hosted by other web sites (i.e. NASA videos).

The obtained e-learning system is a combination of process definitions (*schedules* in the BizTalk jargon), Web Services, BizTalk code plug-ins, static web pages, frames and dynamic web pages as showed in Fig. 2. The internal code plug-ins have been written using the C# [16] language under the Microsoft .NET framework [13], using the provided BizTalk API. The internal code plug-ins have been mainly used to allow the interaction of the students with the WFMS. This issue will be treated later in section 3.5.

3.3 Client/server interaction

As previously said, a student learning session starts with a login operation. As the user press the OK button, the user-id and password are given as parameters to the authen-

[2]An orchestration engine is a particular WFMS born to coordinate services electronically available, offered by one or more companies, in order to obtain another service acting as a composed one. A service electronically available is the external view of a business process offered by a company to others. Another peculiarities of orchestration engines is that the task of coordination is not controlled authoritatively by a single engine, but there are at least as many engines as the number of companies involved in the process and the coordination is performed collaboratively. This is due to the fact that a company doesn't accept to have its own services ruled by others.

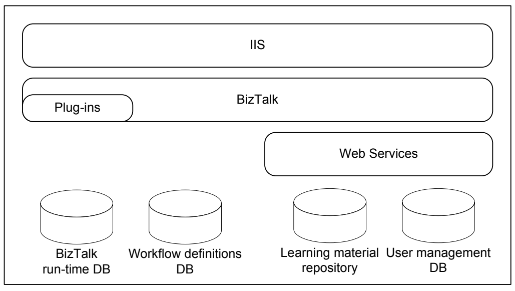

Figure 2: Main architecture.

tication Web Service. The marshaling and unmarshaling of the parameters is done by IIS at server side, but the code executed is a BizTalk code plug-in (this trick is possible because BizTalk is very integrated with IIS). This plug-in will set up the channel for further communications between the student web browser and BizTalk through IIS.

Once the user has been authenticated, the "learning page" is displayed, through which the main learning activities are performed. It contains a list of the previously seen topics, the interaction buttons [3] and a frame hosting a page that is the focus of the student current activity. The frame content could be either the last proposed topic or a page where the student can take some decisions or a quiz page used to take tests. Such a page is generated dynamically and the actual content depends on the state of the workflow instance, namely the state of the student learning activity as tracked by BizTalk. The list of the previously seen topics (history) is useful when a student would like to have a look to an old subject. Every link of the history, when clicked, calls a Web Service whose result is the file containing the interested topic. The history is updated every time the student ends to study a topic and starts a new one.

3.4 Authoring Environment

The activity of coding learning paths within a schedule (a design process) is boring and error prone. The authoring process in our system tries to minimize the effort with many strategies. Teachers do not interact directly with the schedule code, instead they build a course through some visual tools. A great focus on reusability [12] has been given to the authoring environment. Starting from an abstract (and highly reusable) logical representation, a course undergoes a customization process whose goal is to best fit every time the teachers specific requirements. During such a process, three steps can be identified, each of them is increasingly adapting the course representation to teachers needs: (i) The *reusable level* where structured courses are defined aggregating learning materials by means of logical relationships that represent constraints, i.e. "Programming Languages 2 *can be learnt only after* Programming Languages 1", "Seminar on Software Engineering *is alternative to* Software Engineering part 2". A graphical authoring tool enables teachers to define such dependencies and automatically produces the corresponding *didactical level* representation. (ii) The *didactical level* where the workflow composed by learning paths is defined starting from the constraints defined at the higher level. At this level some paths can be explicitly denied, i.e. by denying some topics a teacher can customize a course

[3]The buttons displayed change dynamically basing on the context.

like "Introduction to weather forecasts", originally designed for weather forecast students, to the needs of "Aeronautical engineering" ones. (iii) The *fruition level* where additional details are added, thus leading to the definition of fruition-ready courses, i.e. the records of the students joining the course, the teachers and assistants offering the course. A compiler translates the workflow into an XLANG [20] schedule[4].

Every time teachers need to create a course similar to an existing one, they do not need to create a new *reusable level* course. Instead, they can start from the existing *reusable level* representation leading to a new customized *didactical level* course. Similarly, a *didactical level* course can be customized in different *fruition level* courses. A complete description of the three representation levels is beyond the scope of this paper. For further details see [17].

3.5 Main Issues

One of the main problems encountered during the design of the whole application was how to manage the synchronization between BizTalk and the students. BizTalk is designed to deal with Web Services, code plug-ins, but not to interact directly with human users. Nevertheless in our case the workflow must track the human activity. Thus an event-based infrastructure has been built. When the workflow needs a student input (a decision about which topic to study next, etc.) the workflow starts the execution of an internal plug-in, which will terminate only after receiving an event generated by the student. The event generation is done outside BizTalk: when the student clicks an interaction button, i.e. the next topic button, a Web Service is called which, in turns, generates the event that the BizTalk plug-in is waiting for.

Another problem encountered was how to maintain the students web sessions during the use of the e-learning system. The learning activity required by a course can take weeks and the students do not remain connected all the time. BizTalk interacts to the external world through so-called *ports* (a concept similar to IP ports). In the prototype, BizTalk ports are connected to IIS which manages the students sessions. But when a student does not interact for a while (i.e., because she/he has gone eating) or after a log-out, IIS ends her/his session. As result the connection between BizTalk and the student get lost. The problem was solved by having a redirect to the log-in page every time a student's session ends. As a result when a student does not interact for a while and her/his web session ends, the log-in page is presented whenever the student starts interacting again. The log-in procedure will reconnect the IIS web session to the corresponding BizTalk port.

The last main problem we faced concerns with BizTalk schedules and "sub-schedules". In our prototype, in order to foster reusability, there is the possibility to call from one course another whole one, which in turn can call other courses and so on. A course in BizTalk is described by a *schedule*. To allow a course to call another course as an internal topic, a schedule must call another schedule (i.e., a sub-schedule), wait for its termination and then go on with the next instruction (similar to the call of subroutines). Unfortunately BizTalk was not designed having this behavior in mind. There is no support in the schedule programming language for calling another schedule without terminating the initial one. The problem has been overcome by developing and internal plug-in which launches sub-schedules and wait for their termination.

[4]XLANG is the process definition language used by BizTalk.

4. EXPERIENCE

The developed prototype has been tested with students. A course about Software Engineering named "Design pattern in software engineering" has been developed and some volunteer students have used it to study the proposed subject. The process definition of the course is shown in Fig. 3 (the graphical notation has been introduced in Fig. 1).

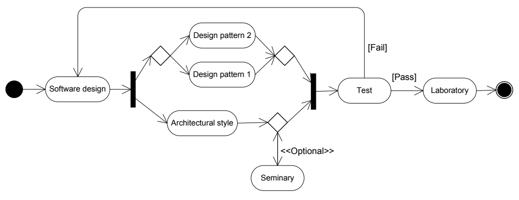

Figure 3: A course on Software Engineering.

The learning material about "Software design", "Design pattern" (1 & 2) and "Architectural style" is in slides format, "Seminary" is a video-recorded seminary available as Windows Media Video stream, "Test" is a web-based application that presents quizes to the students, collects their answers and gives a score, "Laboratory" is a laboratory session where students must practice the just learned topics. During the laboratory activity, students have developed in pairs a Java application. In order to manage versioning of source code, we have exploited PeerVerSy [4], a peer-to-peer configuration management system that makes easy to work when people are disconnected for most of the time.

Due to security reasons it was not possible to let the server be visible outside the university firewall, so the students could use the system only within the campus area. There were some computer rooms available and some areas equipped with wireless connections. The students were able to practice the system during the free time between a lesson and another.

The students missed the possibility to interact with the speaker of the seminar, and missed the presence of a tutor during the labor activity, but the overall assessment was positive: people enjoyed using the e-learning system and the flexibility it was possible in terms of timetable.

5. RELATED WORKS

There are many mature commercial products that can be used to build an e-learning system: Lotus Learning Management System [10], WBT TopClass [18], Oracle iLearning [14], BlackBoard Learning System [1], and WebCT Vista [19]. All of them do not give to teachers the opportunity to define a true course structure. Yet some systems permit to build courses as a group of lessons, it is not possible to define learning paths through the teaching material.

Rule-based systems customize the fruition of courses by mean of rules. CoLab [9] enables collaborative browsing. CoLab, by means of access rules, give teachers the opportunity to define "learning path" through the teaching material. In particular, CoLab allows Coordinated Browsing (learners follow a guide), Coordinated Adaptive Browsing (learners follow a guide but they are allowed to access information on pages depending on access policies that teachers define), Location-based Browsing (there is no guide, access rules allow learners to access pages).

Relationship-based systems systems allow teachers to define a course structure by means of logic relationships among the course components. MediBook [15] is an example of such systems. The basis of MediBook is a formal representation of the medical domain. The formal representation consists of the important medical concepts which are associated with each other by semantic relationships (in other words, an ontology). Courses are associated with concepts. Furthermore, courses are associated with each other by rhetorical relationships (e.g. course A *deepens* course B, course C *is-part-of* course D). Learners can navigate through the structure defined by the rhetorical relationships following the proposed way from a given LO to another) or by the semantic relationships (discovering courses associated to a given concept).

The only application trying to use the power of workflows is Flex-eL [11]. Teachers are provided with a tool that allows the building of a stream of activities that constitutes a learning path. Multiples streams can be associated to a course and students are enforced to learn following one of them. To achieve that Flex-eL uses its embedded workflow functionalities. Nevertheless only sequential activities are allowed, namely constructs like *switch*, *fork* and *join* are not provided. As a result, not all of the potentialities of WFMS can be used in building a learning path. Morever, the sharing capabilities of the workflow engine are not exploited to exchange artifacts among learners and teachers.

6. CONCLUSION AND FUTURE WORK

Several e-learning platforms were designed to give instructors the possibility of publishing material on the web in order to help students in their learning activities. The published material ranges from course notes to ancillary material or even video recorded lectures.

This paper has described a platform we designed leveraging on the power of process management abstractions provided by WFMS. In our approach, when teachers lay out a course, they can benefit of the flexibility, expressiveness and control capabilities usually available for the design of business processes. At the same time a student, having its learning process driven by a workflow designed by a teacher, earns the possibility to customize her/his learning path while maintaining it within the bounds of practicability and usefulness.

Teachers' effort in creating courses for the new platform is eased by some visual tools and by improving reusability. Every created course can be a topic of another course. A new visual designed course is translated into workflow code through intermediate representation levels. These levels allow a teacher to customize the course for different situations by explicitly cutting some branches in the resulting learning paths. The use of an orchestration engine allows to enrich a course not only with static learning material but also with the services provided by third programs or institutions.

Our future research agenda includes further investigations of the interaction among the developed e-learning system and external services, especially automated testing tools. Tutoring and Validation modules are also planned. The Validation tool will track the learners' behaviour within the Virtual Campus environment and define the "user model" in terms of learning attitudes, efficiency of learning strategies, and attitude to cooperation and communication. Derived information will be used to build and make available graphical reports to learners and teachers. The Tutoring tool will exploit the user model to support learners' choices making suggestions which take into account the style, the behaviour and the results obtained in the past.

7. REFERENCES

[1] Blackboard. `http://www.blackboard.com/`.

[2] Workflow management coalition document wfmc-tc-1011., Feb. 1999.

[3] P. Avgeriou, S. Retalis, and N. Papaspyrou. Modelling learning technology systems as business systems. *Software and Systems Modeling*, 2(2):120–133, July 2003.

[4] D. Balzarotti, C. Ghezzi, and M. Monga. Freeing cooperation from servers tyranny. In E. Gregori, L. Cherkasova, G. Cugola, F. Panzieri, and G. P. Picco, editors, *Web Engineering and Peer-to-Peer Computing*, volume 2376 of *LNCS*, pages 235–246. Springer-Verlag, 2002.

[5] Microsoft biztalk. `http://www.biztalk.org/`.

[6] P. Brusilovsky. Adaptive hypermedia. *User Modeling and User-Adapted Interaction*, 11(1):87–110, Jan. 2001.

[7] I. L. T. S. Committee. Draft standard for learning technology systems architecture (ltsa). `http://ltsc.ieee.org/`, Nov. 2001.

[8] I. L. T. S. Committee. Draft standard for learning object metadata. `http://ltsc.ieee.org/`, July 2002.

[9] G. de Jesús Hoyos-Rivera, R. Lima-Gomes, and J. Courtiat. A flexible architecture for collaborative browsing. In *Proc. of the* 11^th *WETICE*, Pittsburgh, Pennsylvania, USA, 2002.

[10] IBM. Lotus learning management system. `http://www.lotus.com/lotus/offering3.nsf`.

[11] j. Lin, C. Ho, W. Sadiq, and M. E. Orlowska. On workflow enabled e-learning services. In *Proc. of the Int. Conf. on Adv. Learning Tech. (ICALT)*, Madison, Wisconsin, USA, 2001.

[12] L. Mainetti, M. Monga, and L. Sbattella. A virtual campus for tethered and untethered scenarios. In *Proc. of FIE 2002*. IEEE, IEEE, Nov. 2002.

[13] Microsoft .net. `http://www.microsoft.com/net/`.

[14] Oracle. ilearning. `http://www.oracle.com`.

[15] A. Steinacker, A. Faatz, C. Seeberg, I. Rimac, S. Hörmann, A. E. Saddik, and R. Steinmetz. Medibook: Combining semantic networks with metadata for learning resources to build a web based learning system. In *Proc. of ED-MEDIA*, Tampere, Finland, 2001.

[16] E. TC39/TG2. Draft c# language specification. Technical report, ECMA, Mar. 2001.

[17] Virtual campus web site. `http://www.elet.polimi.it/res/vcampus`.

[18] WBT. Topclass e-learning suite. `http://www.wbtsystems.com`.

[19] WebCT. Vista. `http://www.webct.com/`.

[20] Microsoft web services for business process design. `http://xml.coverpages.org/xlang.html`.