An approach for managing the evolution of web-based educational applications

Yannis Psaromiligkos*

Technological Education Institute of Piraeus,
General Department of Mathematics, Computer Science Laboratory,
250 Thivon & P. Ralli, 122 44 Athens, Greece
Fax. ++30 210 5381351 E-mail: jpsa@teipir.gr
*Corresponding author

Symeon Retalis

University of Piraeus, Department of Technology Education and
Digital Systems, 80 Karaoli & Dimitriou, 185 34 Piraeus, Greece
E-mail: retal@unipi.gr

Abstract: The management of the infrastructure and content of the World Wide Web is a growing problem for the development of web-based educational applications (WbEA) and is hidden beneath the tremendous benefits of the web. Moreover, such systems undergo more frequent and quite extensive changes in their development and operational period because the client's perception (either the learner, the teacher or institutional organisation) of the complete end product is a 'moving target'. So, there is a need for a disciplined approach to managing the evolution of web-based educational applications. Such a disciplined approach as applied throughout the software engineering process is called software configuration management (SCM). This paper begins a discussion of how the development of web-based educational applications could benefit from SCM techniques and tools and proposes a model for the management of the evolution of such systems. The model couples the development and evolution process of WbEA configurations into one framework, which seems promising for the implementation of a configuration management process. Also, the model focuses on the development of the web-based educational content, defining the configurations of the items involved in the underlying process. Finally, an example is given to demonstrate the benefits of the use of this approach.

Keywords: web-based educational applications; software configuration management; version control; learning technology systems; e-learning; learning resources; learning management systems; instructional design; evaluation.

Reference to this paper should be made as follows: Psaromiligkos, Y. and Retalis, S (2004) 'An approach for managing the evolution of web-based educational applications', *Int. J. Web Engineering and Technology*, Vol. 1, No. 2, pp.148–167.

Biographical notes: Dr. Yannis Psaromiligkos is an Associate Professor at the General Department of Mathematics, Technological Education Institute of Piraeus, Greece. He holds a BSc in mathematics, and an MSc in computer science and operation research from the University of Athens and a PhD in computer science from the Department of Electrical and Computer

Engineering, National Technical University of Athens, Greece. His research interests are in web engineering and software configuration management, web-based learning environments, evaluation methodologies for web-based systems, object-oriented programming and modelling. He is the author of several papers and serves on the editorial board of international journals.

Dr. Symeon Retalis is an Associate Professor at the Department of Technology Education & Digital Systems, University of Piraeus. He holds a diploma in electrical and computer engineering from the Department of Electrical and Computer Engineering Studies, National Technical University of Athens, Greece, an MSc in information technology-knowledge based systems from the Department of Artificial Intelligence, University of Edinburgh, Scotland, and a PhD from the Department of Electrical and Computer Engineering, National Technical University of Athens, Greece. His research interests lie in the development of web-based learning systems, the design of adaptive hypermedia systems, web engineering and human computer interaction. He is the author of several papers, serves in the editorial board of international journals and participates to special interest groups on web engineering.

1 Introduction

The vision of a knowledge-based future where acquiring and acting on information and knowledge is the primary operation of all learners is close to coming true [1]. In order to facilitate the realisation of this vision, learning technology systems are being extensively employed. Learning technology systems (LTS) are learning and training systems that are supported by information technology [2]. Examples of such systems are computer-based training systems, intelligent tutoring systems and web-based educational systems. Special kinds of LTSs are the web-based educational applications (WbEA) which are based on the state-of-the-art internet and WWW technologies in order to provide education and training following the open and distance learning paradigm. WbEAs aim to support and partially automate the instructional process on a subject field, which might concern, for example, a course, a seminar or even a series of lectures [3]. From a different perspective, these systems intend to satisfy certain instructional needs for a subject domain, which have surfaced mainly because of the advances in research and technology, the emergence of the information society and the globalisation of markets.

It is not odd, that WbEAs, which make extensive use of new technologies, have been called upon to solve the vast demand for life-long learning, caused by the advances in technology itself. In particular, the advantages gained from employing web-based educational applications are derived from their potential to: promote advanced interactivity between learners and tutors; offer flexibility concerning the way, time and place of learning; grant multiple media delivery methods through hypermedia; allow several synchronous and asynchronous communication facilities; and provide easy, one-stop maintenance and reusability of resources [4,5]. The design and implementation of such systems though is not an easy task, since they are complex systems that incorporate a variety of organisational, administrative, instructional and technological components [6,7]. A WbEA should be seen as a system comprising three interrelated subsystems:

- The human subsystem, which describes the roles, in as much detail as possible, for each kind of human agent (teacher, learner, tutor, network administrator) involved in the instructional process [8].
- The web-based learning resources subsystem, which is perceived as a mosaic of online learning resources. Such learning resources can be course notes, slideware, study guides, self-assessment questionnaires, communication archives, learning material used for communication purposes, etc.
- The technical infrastructure subsystem, which is divided into common and special. An instructional system basically makes use of services from common infrastructures, which is a set of learning places, that support student learning in general (e.g. laboratories, networking facilities, etc.). However, in order to best support the instructional process, special infrastructures should be created (e.g. multimedia conferencing systems, state-of-the-art hardware components, a specific learning management system, etc.), which will provide services unique to a particular instructional problem [9,10].

Systematic, disciplined development approaches must be devised in order to leverage the complexity and assortment of WbEAs and achieve overall quality within specific time and fund limits. Additionally, the size and complexity of modern WbEAs bring about great intricacy in their crafting as they evolve, as there is not enough knowledge or experience in this field. This also imposes the use of new disciplined approaches to managing their evolution. Such a disciplined approach can be adopted/adapted from the software engineering field.

Software configuration management (SCM) is the discipline of managing the evolution of software systems both during the initial stages of development and during all stages of maintenance. SCM constitutes a key element of the software engineering process and includes many activities that must be carried out consistently. Moreover, it involves many different individuals, such as customers, managers and software engineers as well as many different products such as management plans, specifications (requirements, design, test), code (source and executable), user's manuals, etc. SCM has four coordinating functions [11–13]:

- *configuration identification*: the definition of the software life cycle products that will be under control, their baselines and how they will change
- *configuration control*: the technical and administrative procedures in order to control the changes to products
- *configuration audit*: the function that makes the current status of any software product visible to management
- *configuration status accounting*: the function that provides the development history of any software product, recording the activities of the previous SCM functions.

In large-scale software systems the management of these activities is an extremely difficult job, but essential for effective and reliable evolution of such software. Computer assistance for the representation and evolution of the history of software system development is of great importance. It can avoid the confusion caused by interaction among the different individuals, improving productivity.

In this paper we try to illustrate the need to incorporate (via adaptation) SCM techniques into the development process of a WbEA.

The structure of the paper is as follows: Section 2 presents the SCM process that is being applied in general, (i.e. non-educational) software systems. We should note that as far as we know, this is the first time that ideas on how to apply SCM functions to WbEA development are being recorded. While Section 3 describes the overall development process of a WbEA, Section 4 will analyse the way that SCM can be incorporated into the WbEA development process. Finally, concluding remarks and future plans will be mentioned in Section 5.

2 The software configuration management

When a large number of people work on a software project, some of the problems that can arise are as follows:

- *simultaneous update*: when two or more programmers work separately on the same program, incompatibility of the changes they make can be disastrous
- *shared or common code*: when a change is made to a module shared by several subsystems, all the programmers involved in the development of these subsystems must be informed of the potential effects this change can have on their work
- *versions*: changes must be applied in a controlled manner if regression is to be avoided; moreover, they result in a large number of versions even of a single component and therefore the structure of the system becomes extremely complex.

All the items that are produced as part of the software engineering process are collectively called a software configuration. SCM is the process of identifying, organising and controlling changes to software configurations made in all stages of a software project's life cycle. Its purpose is to maximise the productivity of the development team members by controlling their interaction and minimising mistakes.

Because change can occur at any time, SCM is an 'umbrella' activity that is applied throughout the software engineering process. SCM activities provide means for:

- identifying configurations and changes
- controlling the application of changes
- ensuring the proper implementation of changes
- reporting changes to all the interested team members.

Artefacts that can be divided into several classes are the output of the software engineering process. Moreover, we can view and manage these artefacts as composite or atomic entities. The term configuration item (CI) is used to denote such a controllable (under configuration management control) artefact or a part of it. If we view an item as a collection of other items, then it is called composite or otherwise atomic. The CIs may form hierarchies according to the relation 'is composed of' between composite and their parts (atomic or composite). A CI can be any of the parts of a whole system as long as it is treated and managed as a single unit. Of course, management is simpler if the item belongs to the lower levels of the system's hierarchy.

Because changes are inevitable, CIs may have many versions. The versions of atomic CIs imply the versions of composite CIs and so on. Because of versions the structure of a software system becomes extremely complex. There are two kinds of versions: revisions and variations. Variations represent independent lines of development and each one consists of a sequence of revisions. Variations coexist in time while revisions replace each other because their purpose is primarily to fix bugs. When a CI version is distributed outside, the development organisation is called release. Version control of CIs is one of the fundamental tasks of every software configuration management tool and up till now it has constituted the basis of many research efforts [14–19].

Changes also need a key mechanism to bring them under control because they can rapidly lead to chaos. The key mechanism for managing changes is the baseline. A baseline is a milestone in the software engineering process that marks the completion of a phase accompanied by the delivery of a number of approved (validated) CIs. A baseline is the foundation of configuration management as it provides the official standard, on which subsequent work is based and, therefore, it should be established at an early development point. All the SCM processes revolve around the baselines. Once an initial product level has stabilised, a first baseline is established. Every successive set of validated enhancements establishes a new baseline that provides the cornerstone for further development.

Once a CI becomes a baseline, it is placed in a project database (also called a project library or software repository). The baselines must be protected against unauthorised change while at the same time enabling the programmers to modify and test their code. This flexibility is accomplished by providing the programmers with private working copies of any part of a baseline. Thus they can try out any changes without interfering with the work of anyone else. When their work is ready to be incorporated into a new baseline, change management ensures that each change introduced is compatible with every other change and also maintains system integrity.

3 The development process of a WbEA

Nowadays, educational applications make extensive use of networked technologies. A consequence of this trend is that developers build complex educational systems that incorporate a variety of organisational, administrative, instructional and technological components [7]. The need for a systematic and disciplined way to develop such systems is obvious. In this section we briefly describe the CADMOS methodology (web-based courseware development methodology for open-learning systems), which is a development methodology for WbEAs [20].

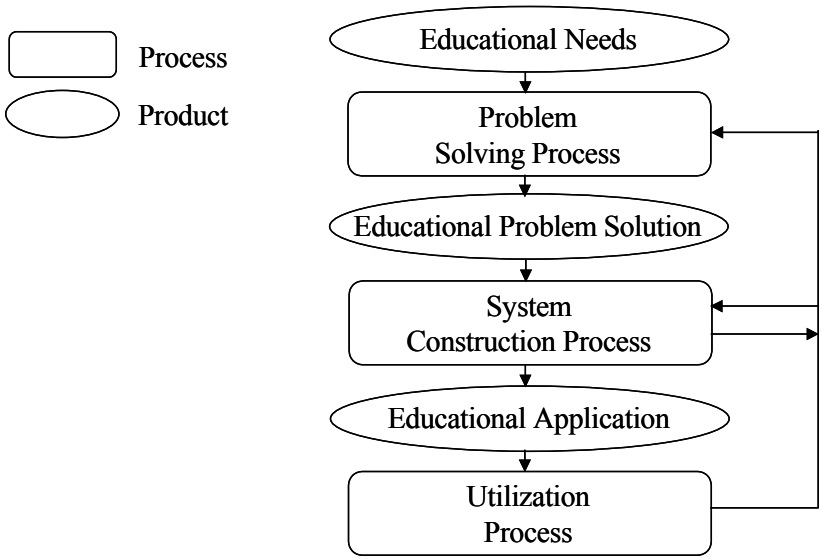
CADMOS supports the evolutionary model for the development of a WbEA [21]. This is because the evolutionary model is iterative and is characterised in a manner that enables developers to construct increasingly more complete versions of the end product. Thus, according to CADMOS, a WbEA should be developed as a series of fully functional builds (working versions of the instructional system). A build satisfies the current set of the requirements of the product under development. The underlying essence of the evolutionary development is that the client's perception (either learner, teacher or institutional organisation) of the complete end product is a 'moving target'. While the builds are tested and summatively evaluated, the user's opinion regarding the

characteristics of the system might change, resulting in changing the requirements and consequently the design and development of the future builds.

The instructional development should contain three main processes, each one subdivided into sub-processes as illustrated in Figure 1:

- the problem solving process
- the system construction process
- the system utilisation process.

Figure 1 Development stages of a WbEA according to CADMOS methodology



The aim of the problem solution finding sub-process is to construct a desired solution to a given instructional problem already defined by the situational evaluation sub-process. The solution to the instructional problem emerges from blending five interrelated sets of learning elements: the learning objectives, the didactic events, the syllabus, the assessment procedure and other issues like prerequisites, fees, technical constraints and so forth. This solution is the main product of this process. It must be a well written, highly detailed document following specific guidelines on how to formulate the learning objectives, the syllabus, the didactic events, etc. (for an overview, see [3,22,23]). Thus, such a 'non-technical' solution, that is the instructional problem abstract solution, plays the role of requirements specification for the WbEA under construction.

The construction process is the actual implementation of an instructional problem solution for a specific learning environment. The construction phase receives as input the solution from the problem finding process. This solution is then divided into three parts:

- human part
- webware part
- specific infrastructure part.

This division is done within the system's engineering sub-process. Each part of the solution specifies how the solution will be realised and supported into a real learning environment. Having specified the solution parts, each one of them is developed according to related methodologies, (i.e. cognitive engineering for training the human actors to play their roles, courseware engineering for developing the web-based learning resources and software engineering for building the specific software infrastructure) finally resulting in three subsystems that are all integrated into one 'whole'.

The last stage of the process is concerned with the utilisation of the system. It is within this process that the developed system will be summatively evaluated and reviewed. The utilisation process consists of:

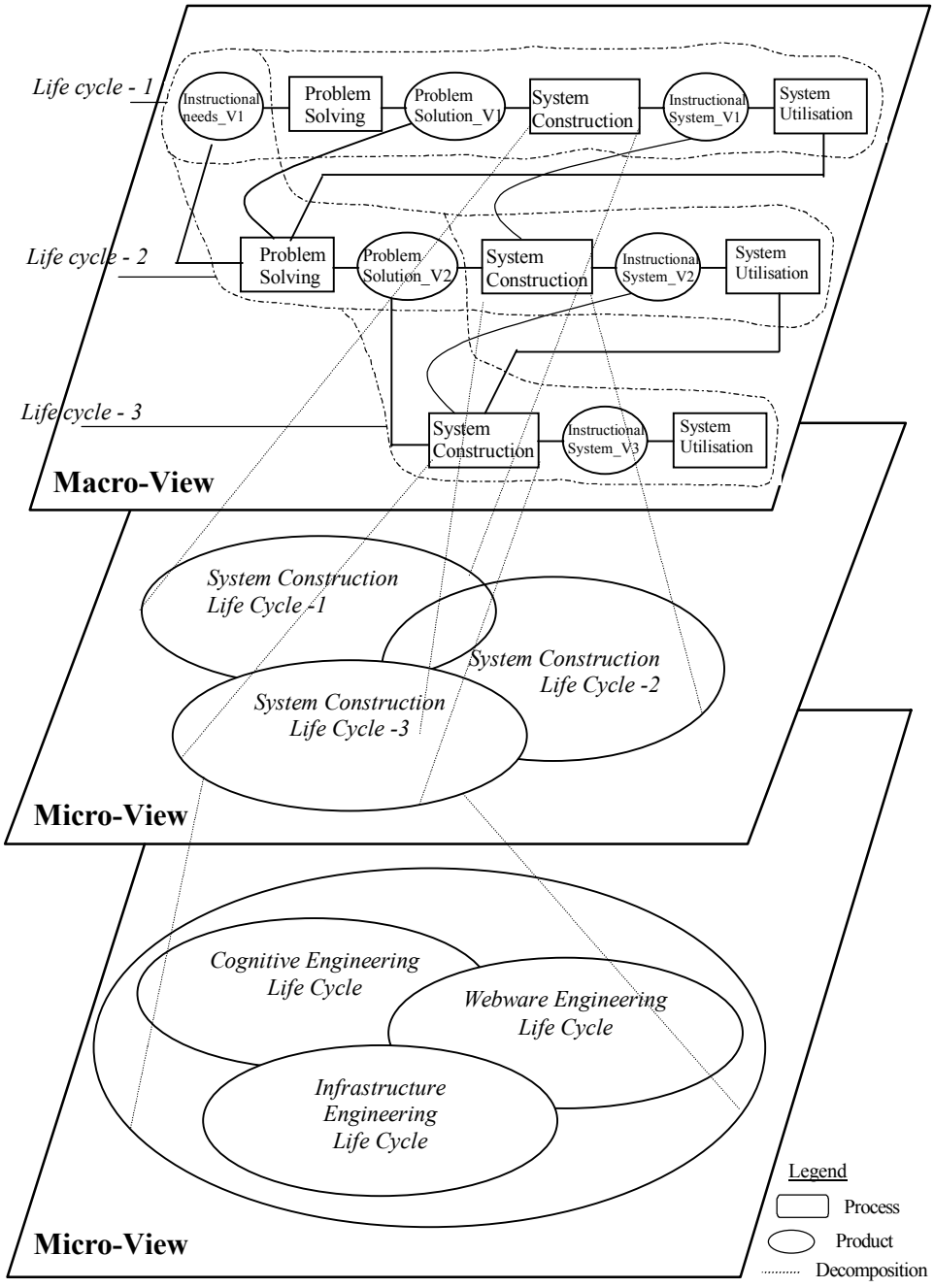
- the instructional process when the instructional system developed is being used, tested and validated in a real learning environment
- the summative evaluation of the instructional problem solution sub-process which provides feedback from both learners and teachers
- the customisation and distillation of the outputs of the instructional process, which can be used for the evolution of the instructional system.

4 Modelling the development and evolution of WbEA configurations

As mentioned in Section 3, there are a number of interrelated subsystems (life cycles) involved in the development of a WbEA and so the configuration of the system becomes extremely complex. A configuration management model should represent the development and evolution of WbEA configurations so that the various life cycles expressing the organisations' specific needs could be supported while at the same time it should ensure and maintain the consistency of the underlying configurations. In this section we try to define such a model for the representation of the WbEA configurations and their evolution.

At first, we consider a macro and a micro view of the WbEA development process in order to implement configuration management (see Figure 2). The macro view involves the top-level processes (phases) of the WbEA development process, that is, 'problem solving', 'system construction' and 'system utilisation' processes as well as the underlying output products. For the 'system utilisation' process there are only intermediate products that might be used as input to other processes during evolution with the necessary adaptation (e.g. student assignments, students' frequently asked questions, etc.). At the micro view analysis is the 'system construction' process because it involves three interrelated subsystems (life cycles) that produce the underlying complexity.

Figure 2 Development and evolution of a WbEA



By looking through the above subsystems we can notice the following: the specific infrastructure subsystem that will provide services unique to the particular instructional problem is a software engineering task. So, the development organisation could follow

current software configuration management techniques and tools for the development and evolution of such a system. As it concerns the human subsystem it is a task of the cognitive engineering field. However, when a configuration management process is introduced into the life cycle of a WbEA, new roles and competencies will be generated that need to be identified and clearly specified:

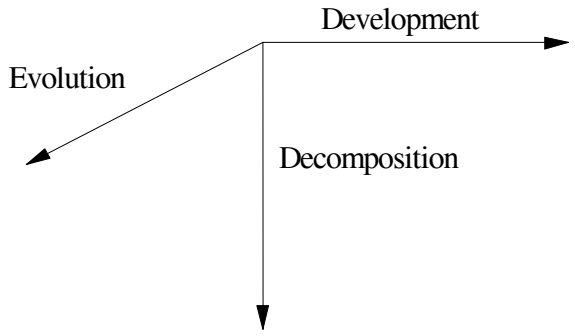
> "Competency refers to a state of being well qualified to perform an activity, task or job function. When a person is competent to do something, he or she has achieved a state of competency that is recognisable and verifiable to a particular community of practitioners." [24]

Typically, a competency is divided into specific indicators describing the requisite knowledge, skills, attitudes and context of performance [25–27].

The webware subsystem deals with the content of a WbEA, which inevitably generates the most interest and it needs further analysis (see next section). Of course, a configuration management model should take into account the interrelations of all the subsystems in order to control the evolution of the WbEA.

We model the WbEA configurations as a three-dimensional space, as shown in Figure 3. The first dimension represents the configuration development, the second dimension represents the configuration evolution and the third dimension represents the decomposition of a configuration.

Figure 3 A model for WbEA configurations



The dimension of *development* represents the way each configuration is developed. At the macro view each configuration involves the top-level phases of the WbEA development process as shown in Figure 1. Such a configuration defines a concrete version of the WbEA. The configurations of the micro view represent how the top-level phases of the WbEA system development process are implemented.

The underlying micro views are subject of the *decomposition* process. During this process we analyse how each composite configuration management entity is constructed. If such an entity is composed of other composites we continue until we define all the elementary (atomic) items. In fact, the definition of atomic items depends on the granularity level of the underlying development model. We recall that one of the most important configuration management processes is the identification of the entities that will be under control, i.e. the identification of configuration management items.

The dimension of *evolution* represents how each configuration evolves over time. We must note that each configuration, once it is created and placed under configuration control (baseline), becomes immutable; i.e we must create a new version of it each time we have to change it. We consider the evolution process as actually being selective iterations of the development process. Such iterations at the macro view generate the versions of the WbEA system; while at the micro view they generate the versions of the various products (configuration management items) comprising the WbEA. Each evolution cycle normally begins after the system utilisation process when the WbEA system is summatively evaluated. However, a formative evaluation process may initiate intermediate evolution cycles, (i.e. evolution cycles at the micro view) before the start-up of the utilisation process.

4.1 Defining configurations of the webware subsystem

The development process of the webware subsystem according to CADMOS follows the principles of the object oriented hypermedia design method (OOHDM) [28,29], which has provided systematic ways to design generic hypermedia applications and not especially educational ones. In this section we deal with the definition of the entities that will be under control, i.e. the identification of configuration management items as well as the management of their changes.

In order to achieve a fine-grained granularity level we propose a model that captures all the entities involved in the underlying process. This model follows the UML notation and utilises its extension mechanisms [30] suitable for a WbEA.

We view each WbEA as a (composite) configuration management item that is composed of learning resources, as shown in Figure 4. Learning resources are organisational units of learning content, each with specified learning objectives. Each learning resource is also a composite item and contains either a number of web assets or other learning resources, hence organising the WbEA in a hierarchical structure (see Figure 5). This structure conforms to learning content packaging standards like [31,32], so it is possible to automate the process of mapping the model artefacts to one of the existing learning technology standards. While learning resources are conceptual elements that allow the structure of the WbEA, they can have visual presentations by means of access assets (see below) that contain links to actual content (content assets).

Figure 4 Web-based learning resources

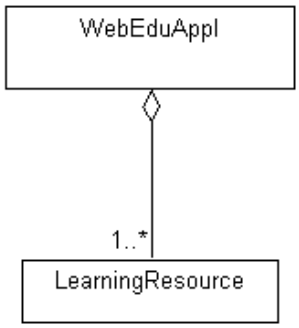
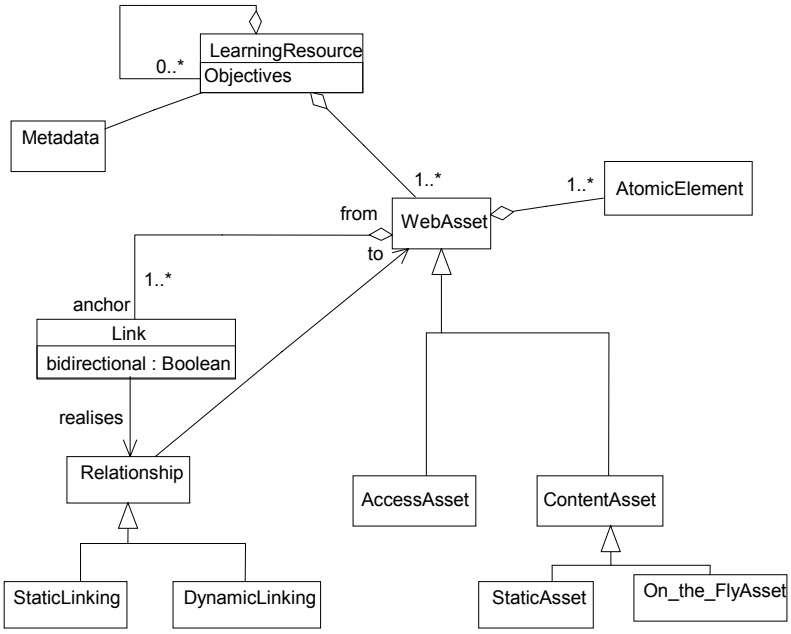


Figure 5 Learning resources detailed model



Furthermore, each web asset is a composite item that includes atomic elements, which are multimedia elements (audio, video, image or simple text), active elements (applets, ActiveX controls, Flash objects), client scripting code and form elements, as illustrated in Figure 6. It can also contain navigational (structural) links (e.g. forward, backward, back to the contents, etc.). It should be mentioned that the navigational links implement structural relationships between the assets of the assets of the courseware and are subject matter independent. For each link, its name and the linked web asset should be noted.

The web assets of the learning resources are divided into the following types, as presented in Figure 5: access assets, whose learning content is minimal, used for navigational purposes since they provide access to web assets with learning material. From one access asset the user can also navigate to another access asset (as often happens with the contents of the hypermedia book). As mentioned above, access assets are usually visual instances of learning resources containing links to the learning resources and other assets contained in the corresponding learning resource.

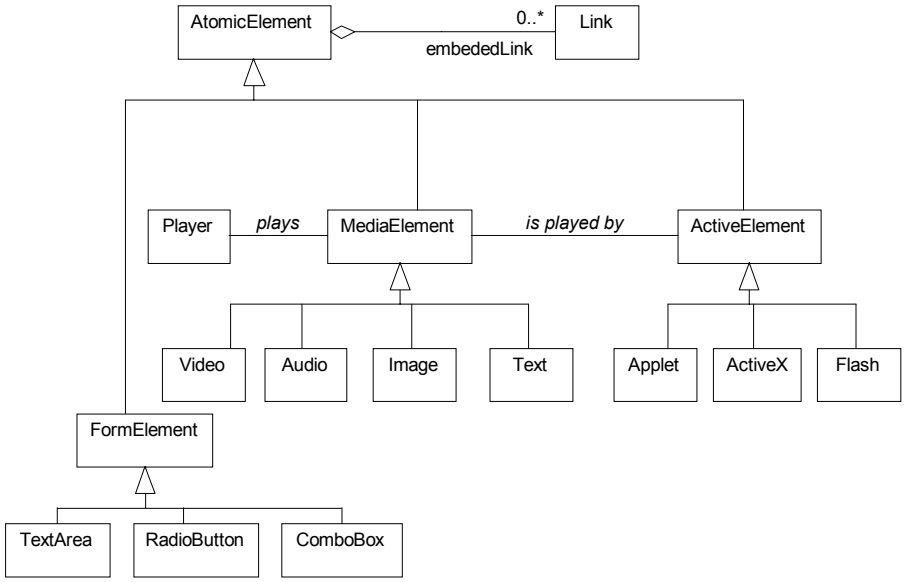
Content assets include informative or learning material. These assets are either 'hard coded' (e.g. HTML files) or created 'on-the-fly', as in the case of .asp pages. So a specification at a lower level must be made.

A web asset contains anchors, which are realisations of relationships among the web assets of learning resources. Anchors can be either static navigations links or dynamic links that activate a process that creates on-the-fly web assets, as is the case of submitting queries to a database, which is followed by the dynamic publishing of the results. Both anchors and relationships are items too.

Finally, a web asset includes atomic elements, namely multimedia elements, active elements, links and code in a mark-up language (e.g. HTML). Moreover, each one of

these elements contains embedded links which are subject specific and do not depend on the general web-based courseware navigational schema. For some kinds of multimedia elements, the designer must specify the software system (player) that will handle its presentation, such as RealAudio or RealVideo, or a special plug-in which must be defined as items, too. In Figure 6, the abstract structure of atomic elements, where media elements is illustrated using a class diagram.

Figure 6 Atomic elements detailed model



4.2 Applying configurations to the webware subsystem development process

4.2.1 The conceptual level

At this level, the description of the WbEA is transformed into a 'conceptual schema' following the underlying object-oriented conceptual model. We represent the 'conceptual schema' (the deliverable of this process) as a composite item containing designs illustrating the webware as a mosaic of learning resources as well as web assets. For example, an electronic book is a learning resource comprising of hierarchically arranged sets of web assets, on-the-fly assets, etc. Apart from the learning resources and the web assets the items involved at the conceptual level also include relationships and the anchors that realise these relationships.

4.2.2 The navigational level

At this level the 'navigational schema' of the WbEA is constructed as an official deliverable. This should be considered as a composite item that complements the 'conceptual schema' giving more details about how web assets are interconnected with hyperlinks. It contains state transition diagrams where the states are the web assets

identified at the conceptual level and transitions, which are the anchors that realise the relationships between the web assets. A series of such diagrams comprise the navigational schema, according to the navigational contexts that can be created according to the various realisations of the relationships among the web assets and the learning resources. So, the classes of items involved at this level are not new. In case new items are needed for the better presentation of the navigational structures (e.g. new access assets, new anchors and relationships) revisions to the conceptual schema must be made for adding these new items.

4.2.3 The abstract interface level

At this level, the graphical user interface (GUI) deliverable of the hypermedia application is constructed, that is the structure of the content, layout and 'look and feel' of the web assets. This composite item complements the previous schemata with the necessary atomic elements of each web asset. So, the items defined in this level are all the atomic elements defined by the model described in Section 4.1, e.g. form elements, media elements and active elements.

The interface design is ruled by the principles of the page metaphor, a practice taken from multimedia engineering where it has been extensively adopted and used. Page metaphor is used to specify the page components with graphic symbols and to deploy them on the screen showing their layout. Therefore, with the use of graphical semantics, the design depicts the page form just as it will be implemented. The designs made are actually reusable page templates. For instance, if we design the page template of one paragraph of an online book in a hypermedia application, then all the other paragraphs of the book might have the same look, using the same components with the same layout, having the same frames etc.

4.2.4 The implementation level

At this level the actual implementation of the previous designs will take place. All the information needed is already defined and constructed in such a way as to be independent of the implementation environment. However, at this level the particular runtime and development environment must be taken into account. So, the items appearing at this level are runtime specific and they depend on the data model of the language that will be used as well as on how they will be stored. Such items may be stored in various files such as html, asp, jsp, gifs, etc.

During implementation we define certain metadata on them according to the principles of learning objects standardisation bodies like IMS, IEEE LOM, SCORM, etc. All atomic elements must have metadata that describe various aspects of them, like author details, type or format, size, etc. The definition of metadata during the implementation phase is of paramount importance as it facilitates the management of the web asset resources and their accessibility and reusability.

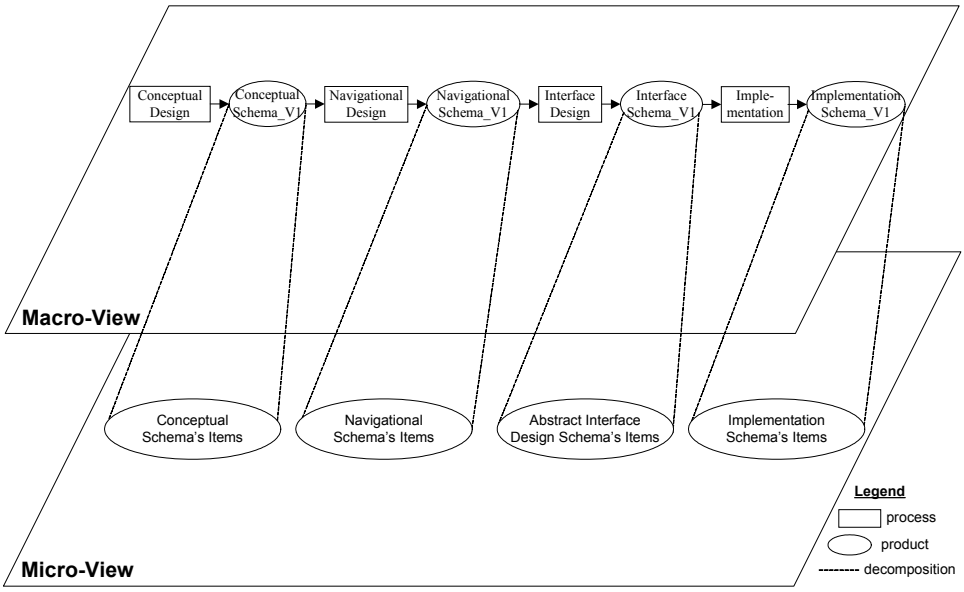
4.3 An example

In this section we describe some parts of a WbEA that have been developed and evolved for the course 'An introduction to compilers' offered by the software engineering laboratory of the National Technical University of Athens. At the macro view, the first

version of the application can be represented by the development process labelled as «Life cycle-1» in Figure 2. The cornerstone of the special technological infrastructure was the WebCT course management system [33]. This system hosted the web-based learning resources, the details about students and instructors (personal data and records) and the data used for administration (course management).

A simplified view of the webware subsystem development process is shown in Figure 7. In each top-level step a number of configuration items are produced. These items define a schema, which becomes a baseline in order to proceed to the next step. Each successive schema complements its precedent schema with new configuration items. When the process terminates, a new version of the webware subsystem is created. However, the underlying process is iterative (each step is validated according to guidelines for formative evaluation of the instructional design, checking structural, navigational, aesthetics, usability and functional issues) and may contain many iterations before an actual version of the webware subsystem is constructed. So, there may be many versions of the above baselines before the system proceeds to the utilisation process.

Figure 7 The webware subsystem development process



The webware part of the WbEA includes a variety of learning resources such as: a hypermedia didactic book, two case studies, self-assessment web-based questionnaires, a collection of past exam papers, a course description and a study guide. Because of space limitations, at the micro view we will concentrate on one of the case studies. This case study contains learning material for giving the students a complete example on how to create a compiler for an educational language called 'Russel'. This case study has the structure of a hypermedia didactic book. It also contains active elements that are being used by learners to test the validity of already prepared examples as well as their own intermediate products during the development process of a compiler (e.g. lexical analysis). The resources have been developed using HTML and CGI scripts and have

been stored on a web server that hosts the WebCT learning management system [33]. The language of the learning material in the current implementation is Greek and so we avoid showing the real interfaces (it will look Greek to the reader).

Figure 8 shows the items comprising the first version of the 'conceptual schema'. Four learning resources, each one containing a set of web assets, both access and content assets have been developed. For presentation purposes we show only the relationship 'SemAnalysisRel' and the anchor (SemAnalysis) that realises it. The underlying relationship relates the web asset 'SemanticAnalysis' of the case study with the corresponding section of the didactic book.

Figure 8 The first version of the 'conceptual schema'

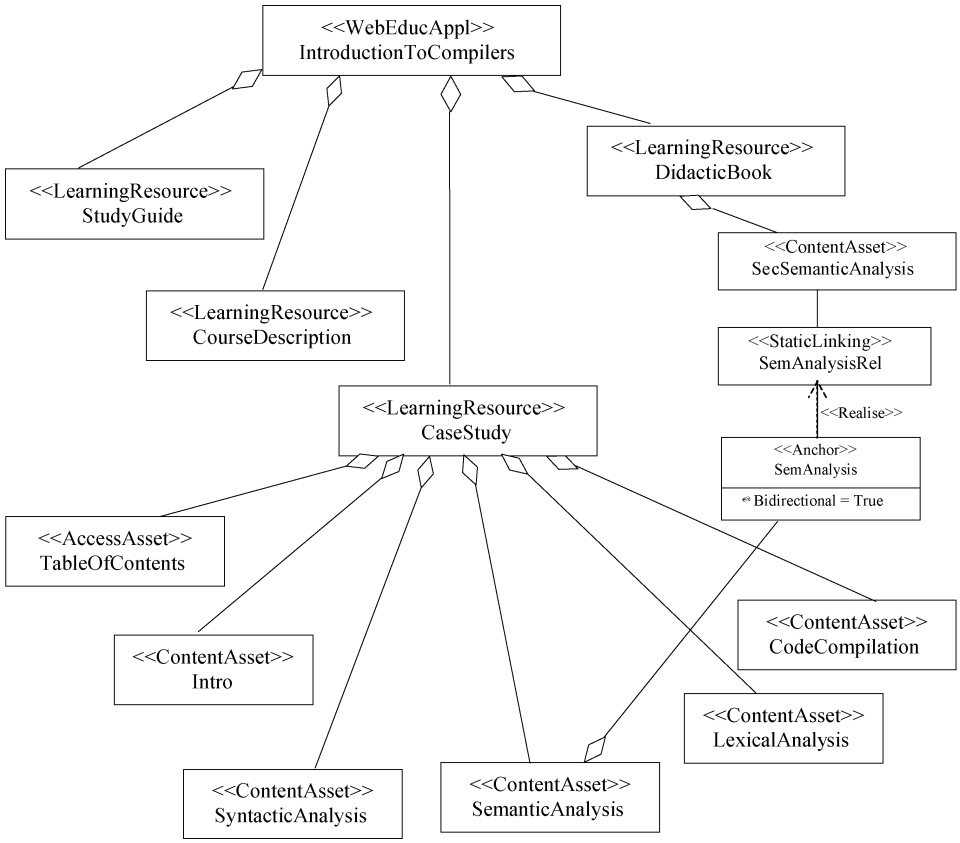


Figure 9 shows the items comprising the first version of the deliverable of the navigational phase. We show only the resource *Case Study* and for reasons of simplicity, only four of the actual 27 content assets contained into the current learning resource are shown. An access asset named *Table Of Contents* is shown containing links to the content assets of the current learning resource. As Figure 9 shows, the *Table Of Contents* contains anchors that realise both static and dynamic linking.

Figure 9 The first version of the 'navigational schema'

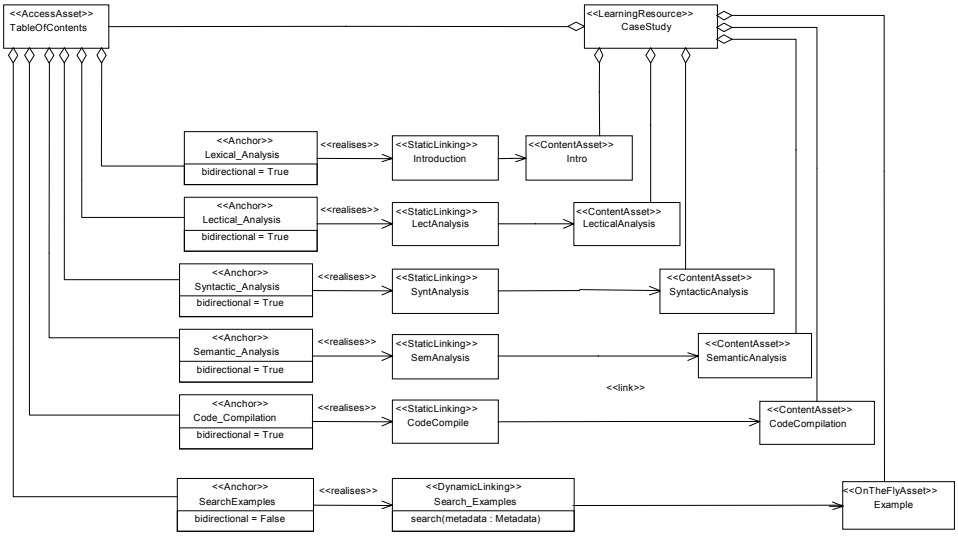
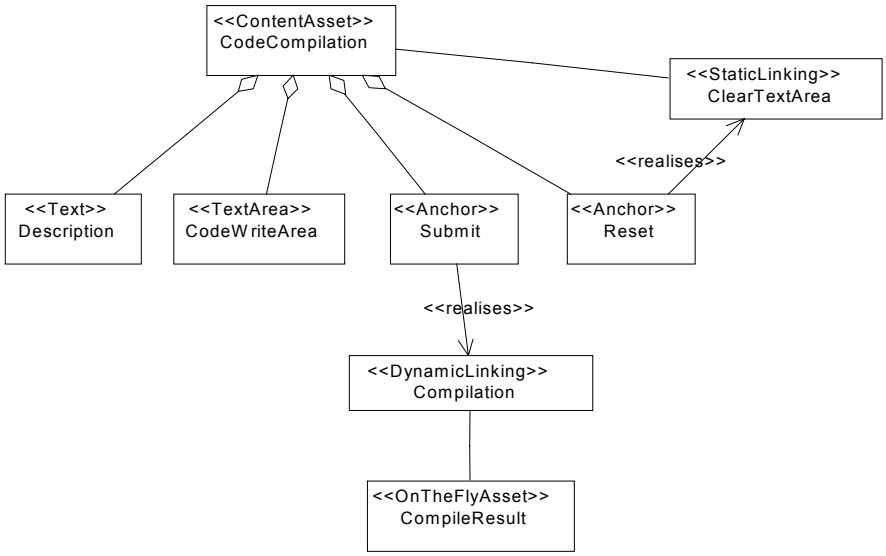


Figure 10 shows the items (atomic elements) comprising a particular asset of the resource *Case Study* of the abstract interface design phase. This asset has text and a form element and it is used for validating the intermediate products during the development process of the 'Russel' compiler.

Figure 10 The abstract interface design phase of a web asset version



Though it is not obvious from the simple previous example, there may exist nesting structures of learning resources of an arbitrary depth. We propose a recursive process of identifying the learning resources of a particular WbEA until the definition of the web assets and their contained atomic elements, which are the finer granularity elements in this model.

After the utilisation process, the WbEA is summatively evaluated leading to new evolution cycles in order to implement the necessary changes as suggested by the results from quantitative and qualitative analysis of students' feedback. Finally, the distillation and customisation process of the outputs of the instructional process can also be used for the evolution of the WbEA. However, all these evolution cycles are normally captured by our model in such a way that it is easy for the designer to find out all the consequences of a given change. A similar approach from the configuration management domain is called change-based versioning [17].

5 Related work

Although there is still no consensus on a general life cycle model of a web application, one can identify a set of typical activities involved in the development of a web application by analysing the life cycle models of traditional information systems and the proposals for structured hypermedia design [28,34–36]. Presently, it seems that no specific tool or class of tools is fully addressing the whole life cycle of web applications, which require the same communication paradigm and interface quality [37]. Implementation oriented approaches like WebML [38] seem the most adequate choice, although development and maintenance with their supporting tools still require a substantial effort.

Moreover, the activities of the configuration management phase are usually neglected. Software configuration management techniques have been around for many years and excellent tools have been around. The WWW needs these benefits as well, especially as the industry matures and continues on its exponential growth path. Applying software configuration management concepts, principles and approaches to web development has been identified as an open research topic for the configuration management area and web engineering [39,40] as well as the e-learning technology field [41]. Our approach addresses this gap and combines ideas stemming from the software engineering (evolutionary development), the hypermedia modelling (the object-oriented paradigm) and the learning technology fields.

6 Concluding remarks and feature plans

There is a need for a disciplined approach to managing the evolution of WbEA because they are complex systems, have a long lifetime and their representation and evolution history is very complex. This is a problem analogous to the software engineering community for software systems and configuration management has the potential to provide a solution in this area.

In this paper we described a model for the development and evolution of WbEA configurations. Our model couples the development and evolution process into one framework, which seems promising for the implementation of a configuration

management process. This approach has not been a formal method as yet. It needs to mature in order to have computerised support.

So, among our future plans are the following:

- the refinement of the model at the micro view in order to formalise the configuration management process
- the definition of a version model for the WbEA that will support both revisions and variations.

In this work we mainly focused on the revision aspect of WbEA. However, we could have variations of the WbEA in order to support multilinguality and adaptability. We have already developed the first version of an adaptive WbEA and its underline architectural model [42] and we will now work on applying the proposed model for the evolution of the system's configurations.

Finally, we are in the process of designing a configuration management tool for having a computer assistant in this process. We do not try from scratch but we try to make additions to existing tools that are being used in software configuration management. We also tend to persuade ourselves that both course management tools (like WebCT, Blackboard, etc.) and the learning objects management tools (which follow the LTSC LOM or IMS or EML standard) should have features that support versions of the specific infrastructure, the learning material and the WbEA in general.

References

1 Sun Microsystems (1999) 'Understanding distance learning architectures', *A White Paper*.

2 IEEE Learning Technology Standards Committee (LTSC) (2001) Draft Standard for Learning Technology – Public and Private Information (PAPI) for Learners (PAPI Learner) – Core Features, Draft 8, November, Available from: http://ltsc.ieee.org.

3 Gagné, R., Briggs, L. and Wager, L. (1994) *Principles of Instructional Design*, Fort Worth, TX: HBJ College Publishers.

4 McCormack, C. and Jones, J.D. (1997) *Building a Web-based Education System*, Wiley Computer Publishing.

5 Lowe, D. and Hall, W. (1999) *Hypermedia and the Web: An Engineering Approach*, John Wiley Ltd.

6 Moore, M.G. and Kearsley, G. (1996) *Distance Education: A Systems View*, Wadsworth Publishing Company.

7 Carlson, P.A. (1998) 'Advanced educational technologies – promise and puzzlement', *Journal of Universal Computer Science*, Vol. 4, No. 3.

8 Lindner, R. (2001) Expertise and Role Identification for Learning Environments (ERILE) Proposed Standard Draft for German DIN NI-36, Available from: http://www.igd.fhg.de/~lindner/PROMETEUS/SIG-DESIGN_Meeting-Point.html

9 Ford, P., Goodyear, P., Heseltine, R., Lewis, R., Darby, J., Graves, J., Sartorius, P., Harwood, D. and King. T. (1996) *Managing Change in Higher Education: A Learning Environment Architecture*, Open University Press, London.

10 Halaris, J., Geropoulos, S. and Pintelas, P. (2002) 'E-Learning using multimedia tele-teaching labs', *Themes in Education*, Vol. 3, No. 2, pp.141–164.

11 IEEE/ANSI (1987) *IEEE Guide to Software Configuration Management*, ANSI/IEEE Std 1042–1987, IEEE Press, New York, NY, USA.

12 IEEE/ANSI (1990) *IEEE Standard for Software Configuration Management Plans*, IEEE Std 828-1990, IEEE Press, New York, NY, USA.

13 IEEE/ANSI (1990) *IEEE Standard Glossary of Software Engineering Terminology*, IEEE Std 610.12-1990, IEEE Press, New York, NY, USA.

14 Estublier J. (1988) 'Configuration management: the notion and the tools', *Proceedings International WorkShop on Software Version and Configuration Control*, Stuttgart, FRG.

15 Estublier J. (1995) 'Software configuration management', selected papers from *International WorkShops on Software Configuration Management*, SCM-4 and SCM-5, Vol. 1005 of Lecture Notes in Computer Science, Springer-Verlang, Seattle Washington.

16 Estublier J. (1999) 'System configuration management', *Proceedings of the 9th International Symposium on Software Configuration Management*, SCM-9, Toulouse, France, Vol. 1675 of Lecture Notes in Computer Science, Springer-Verlang.

17 Conradi, R. (1997) 'Software configuration management', *Proceedings of the 7th International Workshop on Software Configuration Management*, ICSE'97, Vol. 1235 of Lecture Notes in Computer Science, Boston, MA, USA, 1997, Springer-Verlang.

18 Magnusson, B. (1998) 'System configuration management', *ECOOP'98 SCM-8 Symposium*, Vol. 1439 of Lecture Notes in Computer Science, Brussels, Belgium, Springer-Verlang.

19 Van Der Hoek, A. (2001) 'Software configuration management: new practices, new challenges and new boundaries, *Proceedings of the 10th International workshop on Software Configuration Management*, 23rd ICSE, Toronto, Canada.

20 Retalis, S. (1998) 'CADMOS: an instructional systems development methodology with emphasis on the construction of web-based learning resources', *PhD thesis*, National Technical University of Athens (in Greek).

21 Schash, S.R. (1990) 'Chapter 3: software life cycle models', *Software Engineering*, Homewood, IL, Aksen Associates, pp.20–40.

22 Rowntree, D. (1994) *Preparing Materials for Open, Distance and Flexible Learning – An Action Guide for Teachers and Trainers*, ISBN: 0749411597, Kogan Page, UK.

23 Tennyson, D. and Breuer, K. (1997) 'Instructional theory: psychological perspectives', in Tennyson, R.D., Schott, F., Seel, N. and Dijkstra, S. (Eds.): *Instructional Design: International Perspectives, Vol. I: Theory and Research*, Hillsdale, NJ, Lawrence Erlbaum Associates, Publishers.

24 Spector, M.J. and De La Teja, I. (2001) 'Competencies for online teaching', *NY: ERIC Clearinghouse on Information and Technology*, EDO-IT-2001-09, December, Available from: http://www.ericit.org/digests/EDO-IR-2001-09.shtml.

25 International Board of Standards for Training (2002) *Performance and Instruction (IBSTPI)* Available from: http://www.ibstpi.org/

26 Richey, R.C., Fields, D.C., Foxon, M., Roberts, R.C., Spannaus, T. and Spector, J.M. (2001) *Instructional Design Competencies: The Standards*, 3rd ed., Syracuse, NY, ERIC Clearinghouse on Information and Technology.

27 Goodyear, P., Salmon, G., Spector, M., Steeples, C. and Tickner, S. (2001) 'Competencies for online teaching', *Educational Technology Research and Development*, Vol. 49, No. 1, pp.65–72.

28 Scwhabe, D. and Rossi, G. (1995) 'The object-oriented hypermedia design model OOHDM', *Communications of the ACM*, Vol. 38, No. 8, pp.45–46.

29 Scwhabe, D. and Rossi, G. (1998) 'An object oriented approach to web-based application design', *Theory and Practice of Object Systems*, Wiley and Sons, New York, ISSN 1074–3224, Vol. 4, No. 4.

30 Booch, G., Rumbaugh, J. and Jacobson, I. (1999) *The UML User Guide*, Addison-Wesley.

31 Global Learning Consortium (2001) *IMS Content Packaging, – Best Practice and Implementation Guide*, version 1.1.2 – Final Release, Available from: http://www.imsproject.org/

32 Advanced Distributed Learning Initiative (2001) *Sharable Content Object Reference Model*, Available from: http://www.adlnet.org/

33 WebCT (2001) Available from: http://www.webct.com/

34 Garzotto, F., Mainetti, L. and Paolini, P. (1995) 'Hypermedia design, analysis and evaluation issues', *ACM Communication*, Vol. 38, No. 8, pp.74–86.

35 Isakowitz, T., Stohr, E.A. and Balasubramanian, P. (1995) 'RMM: a methodology for structured hypermedia design', *ACM Communication*, Vol. 38, No. 8, pp.34–44.

36 Nanard, J. and Nanard, M. (1995) 'Hypertext design environments and the hypertext design process', *ACM Communication*, Vol. 38, No. 8, pp.49–56.

37 Fraternali, P. (1999) 'Tools and approaches for developing data-intensive web applications: a survey', *ACM Computing Surveys*, Vol. 31, No. 3, pp.227–263.

38 http://webml.org

39 Dart, S. (1999) 'Content change management: problems for web systems', in Estublier, J. (Ed.): *Proceedings of 9th International Symposium*, SCM-9, Toulouse, France, Vol. 1675 of Lecture Notes in Computer Science, Springer-Verlang, pp.1–16.

40 Murugesan, S., Deshpande, Y., Hansen, S. and Ginice, A. (2001) 'Web engineering: a new discipline for development of web-based systems', in Murugesan, S. and Deshpande, Y. (Eds.): *Proceedings of the International Conference on WebEngineering 2000,* Web Engineering 2000, Vol. 2016 of Lecture Notes in Computer Science, Springer-Verlang, pp.3–13.

41 IEEE Learning Technology Standards Committee, (LTSC) (2001) *Draft Standard for Learning Object Metadata (LOM) Draft 6.4*, Available from: http://ltsc.ieee.org.

42 Papasalouros, A. and Retalis, S. (2002) 'Ob-AHEM: a UML-enabled model for adaptive educational hypermedia applications', *Interactive Educational Multimedia*, Interactive educational Multimedia, ISSN 1576-4990, special issue on the theme 'adaptive educational multimedia', No. 4, April 2002.