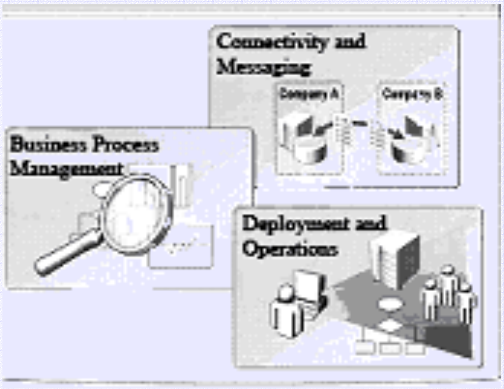
Working Paper Series

Working Paper Series No 1

Report Completion: September 2006

Framework for Component-Based Multimedia Courseware Development

Syed Akhter Hossain
M Lutfar Rahman



EAST WEST UNIVERSITY
CENTER FOR RESEARCH AND TRAINING

Working Paper Series No 01/2007
Report Completion: September 2006

Framework for Component-Based Multimedia Courseware Development

by

Syed Akhter Hossain*, Lutfar Rahman**

Center for Research and Training (CRT)
East West University
Dhaka, Bangladesh
Summer 2007

* Associate Professor, Department of Computer Science and Engineering, East West University, 43 Mohakhali C/A, Dhaka-1212, Bangladesh, ** Professor, Department of Computer Science and Engineering, Dhaka University

Table of Contents

Abstract

Multimedia Courseware supplements significantly in interactive teaching and learning scenario. The visualization and virtual environmental stimulation through interactive multimedia courseware is a story of both successes and failures. The success and failure most of the cases relates to the acquisition of resources for developing the quality courseware, the lack of well defined methodology that could facilitate this process while ensuring a certain level of product quality. The classical problems of software engineering have been also observed in case of multimedia courseware development. Existing multimedia authoring tool does not support rule based courseware development depending on the user context. In this work, we have presented a general solution to the stated problem in a systematic and proven model for courseware development to support development and reengineering process. The proposed framework is intended to facilitate the component based process to reduce the development time and costs, as well as to improve the capability for predicting the courseware's quality level.

Keywords: Multimedia, Courseware, Software Engineering

1.0 Introduction

Today computers as well as computer-based learning material are already a sine-qua-non part of any undergraduate, graduate, technological, and even non-technological study curricula [see Communications of the ACM, January 1998 for a number of examples]. This kind of material, combined with the tools which make it available to the students and that enable collaborative and/or cooperative learning activities among themselves, is known as "courseware" and with effective integration with audio, video and text transform the media to "Multimedia Courseware". Courseware, however, may also be used effectively to complement and enrich the traditional type of learning. Courseware development is not an easy task and often requires well defined methodology suitable for the specific subject to be taught. The result of a courseware development effort often depends on the people doing the work, and the set of tools and methodologies chosen for the task [1,2].

The development of multimedia courseware is technically difficult, conceptually iterative and thus altogether a very costly process. Important characteristics that distinguish the development process of multimedia systems from the development process of other software systems are: the inter disciplinarily of the developers (computer scientist, psychologist, graphic designer, media specialist, domain expert), the high requirements on creativity, synthetic design, the consideration of psychological and ergonomical aspects, as well as engineering aspects during the development process [3].

Currently, there are good known models and methodology for developing courseware like CDM -Courseware Development Methodology- [4] [cf. Appendix A], PROFIL [5] etc. However, these methodologies are not well specified, making it very difficult to apply out of the original context they were conceived for. Most of this basically follows cascade development model, thus presenting a number of well-known weaknesses [3].

On the other hand, there are courseware tools [6] available which offer a computer-based environment to develop multimedia courseware but do not have an associated methodology to guide the development process along with the strict compliance with the pedagogic and didactic approaches. The methodology is particularly necessary to support the courses-curricula development in association with pedagogic and didactic approach [7].

It is clear that, since the courseware development process depends on the people, tools, and methodologies involved, and considering the fact that there is still not a clear methodology or framework to carry out this process, the results will depend mainly on the abilities of the involved people. This situation is prone to cause many drawbacks, as discussed in [8], which are typical for any 'hand-crafted' process.

Consequently, we need a development framework that has a strong connection to the learning context. Highly desirable is a component based model that encapsulates a semantic meaning and is strongly related to the learning subject, but still flexible enough so that it can be adapted to different needs and different contexts.

The motivation and aim of this research work is to create a framework based on Component Based Development (CBD) methodology to facilitate flexible environment for the production of multimedia courseware. The developed framework was tested on an open source platform for specific courseware in the area of software engineering. The key aspect to use effectively the framework is the management of requirements, specifically, the management of courseware learning goals as if they were user requirements. This framework includes specific characteristics of courseware development, and it has helped the authors to carry out this task in a systematic way, by following a set of well-defined steps.

In the rest of the paper, Section 2 presents the courseware development lifecycle and discusses its similarities and differences with a traditional software development lifecycle. In Section 3, the proposed framework is described with the context of multimedia courseware development. The framework is used in a case study and is discussed in Section 4. Conclusions of the work are presented in Section 5.

2.0 Multimedia Courseware and SDLC

Software Development Life Cycle (SDLC) as defined by the classical software engineering comprises Analysis, Design, Implementation, Testing, Deployment, and Maintenance phases. Likewise to software development, in multimedia courseware development the phases: analysis, design, implementation and validation are obligatory though the work processes are different as compared to the traditional software development [1-3].

During the analysis phase both the work scenario and the requirements for the product to be built should be defined. The requirements are the set of educational goals to be reached by the students during the conduct of the course. In this phase, both the process and product risks should also be identified, and the effort level for courseware development should be estimated as well along with the resource identification for the multimedia.

During the design phase the requirements are translated into a representation of the final product that can be evaluated in a static way (similar to a blueprint). In the case of a multimedia courseware, this consists on designing the means to be used for reaching the course goals with the help of media and interactivity. These means are a combination of a didactic activity, media element and a course content, designed for attaining a specific educational goal.

In the implementation phase, the final product (courseware) is built, based on the design obtained in the previous phase. This courseware should contain at least the following elements: the didactic material, the tools to carry out the foreseen activities, and a detailed planning of the teaching/learning process to be developed. All these components should be organized in such a way that the course's dynamics inline with the media interactivity lead to the attainment of the preset goals.

Finally, in the validation phase, both levels for courseware goal attainment and for courseware acceptation are evaluated with the help of the stakeholders. Based on this feedback data, the weaknesses and strengths of the product are analyzed. This information will then be used as input for a reengineering phase as shown in Figure 1.

The most important differences that can be distinguished between both the software and the courseware development processes are the following ones:

1. From the point of view of the requirements, both processes have product and user requirements [4]. However, unlike the software systems, the courseware design depends fundamentally on the user requirements rather than on the business process. Thus, the only way to validate the courseware requirements is through experimentation based on the goal. Even different requirements may evolve from different user perspectives for the same courseware development, which is unlikely to happen in case of software systems.

2. Since we have considered the educational goals as a user requirement, the characteristic of requirement management is an important issue to be considered during the courseware development process. This kind of requirements should be organized hierarchically in order to assure that the upper goal be reached if the lower goals are previously attained as well. Typically, this does not happen with the requirements of software systems.

3. Generally, the evolution speed of a courseware is greater than that of software systems. Consequently, the development process needs a strong support for the reengineering activities.

4. In courseware, automatic mechanisms should be foreseen to verify the achievement of the various goals, because these cannot be verified statically. The only way of verifying the completion of these goals (user requirements) is by using the product. This information will then be used to redesign the product. With software systems, this is unlikely to happen, because this feedback is left to the user's opinion.

5. In the software lifecycle, both software system specialists and applications domain experts develop each phase. With courseware, instead, experts of the applications field are required to carry out each phase.

2.1 Multimedia Courseware

Courseware refers to content specific instructional software, which functions to generate instruction with the support of instructional delivery system. Multimedia courseware evolves within the same definition of the courseware in an environment encompassing the interactions and transformations of the semantics through multimedia. In a multimedia courseware product there are five basic elements namely the content and learning/pedagogical methods as the main components, the learning objectives and the medium as its attributes, the media component and the architecture, which organizes the courseware as shown in the Figure 1.

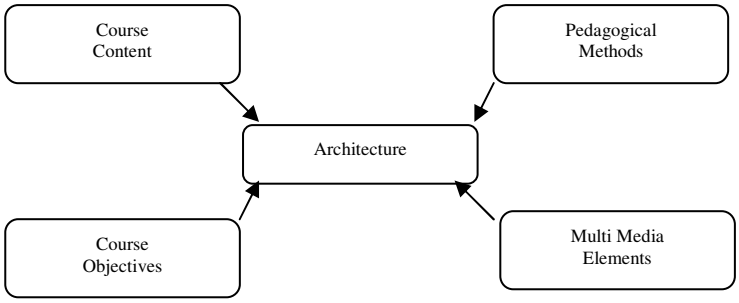


Figure 1. Block structure of Multimedia Courseware

As shown in the above block diagram, the course content or the subject matter with the determination of the multimedia components to meet the learning objectives plays the essential role of the multimedia courseware. This content with the required media element plan is linked to some types of learning/pedagogical methods for the strategy to achieve the expected learning outcomes. The multimedia elements (text, audio, graphics, animation and video) along with the content, pedagogical methods and objectives are organized into a specific courseware product through the control component called architecture.

2.2 Limitations of Existing Framework

There are authoring tools and technologies for the development of multimedia presentation with proprietary format that requires specific learning on using the tool. In the area of dynamically generating multimedia presentation and courseware, there exist only few research approaches. Cuypres system [9] employs constraints for the description of the intended multimedia programming and logic programming for the generation of multimedia document. Also the different approaches uses complex and application-specific customization which imposes requirements for additional programming. The existing approaches usually based on fixed data model for describing user interfaces, structural presentation constraints, technical infrastructure etc. The reusability in most of the existing framework and tools requires enhancement of the existing model. A change of the input data models as well as an adaptation of the presentation with the story boarding and synchronization of the learning path is difficult. The proposed framework focuses on maximum reusability from the component point of view. The framework addresses coupling

and cohesion issues and provides an extendible and flexible environment for the development as well as maintains a multimedia courseware.

3.0 Proposed Framework

This section presents the Framework for Multimedia Courseware Development (FMCD), a generic software framework for the development of courseware based on reusable software components. The general overview of the framework in shown in Figure 2. Multimedia Courseware developed using the FMCD framework requires four sub-systems or macro components as shown in the figure 2. Content sub-system, Navigation sub-system, Collaboration

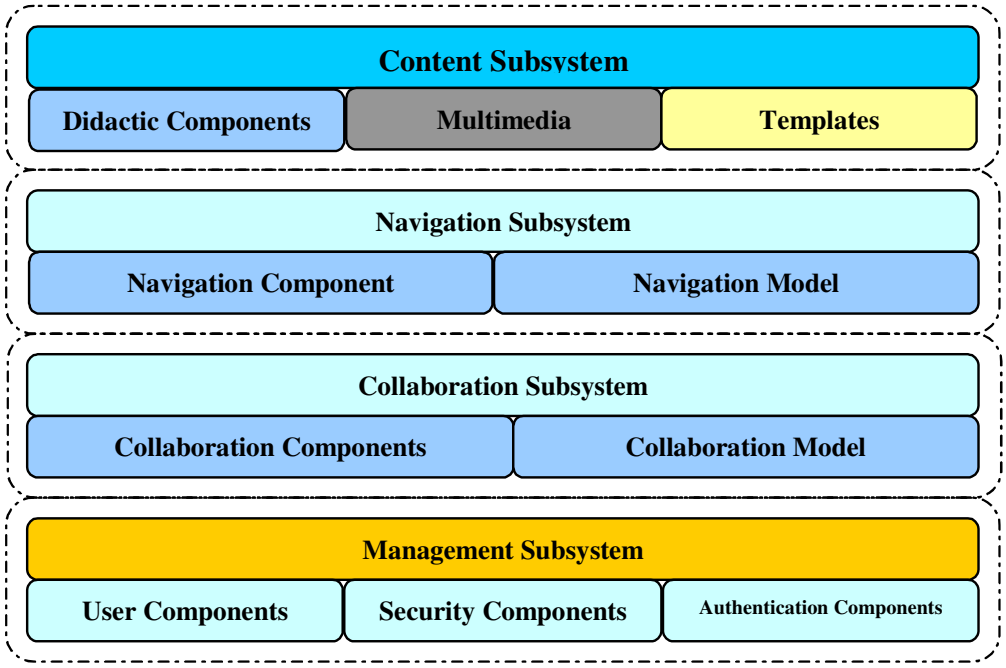


Figure 2. Framework for Multimedia Courseware Development

sub-system and Management sub-system. Each of the sub-system is composed of some basic and some customizable components decentralized for specific responsibility in the architecture of the courseware. The sub-systems are abstracted from the common multimedia courseware development requirements. The content sub-system stores the courseware didactic material. The navigation and collaborative sub-system provides support to perform necessary collaboration activities among the instructors, students and other

assigned roles. The management sub-system is in charge of user administration and necessary authentication and security management of the contents.

The architecture of the content sub-system is illustrated in figure 3. The framework is designed to be extensible such by embedding additional modules through well-defined interfaces.

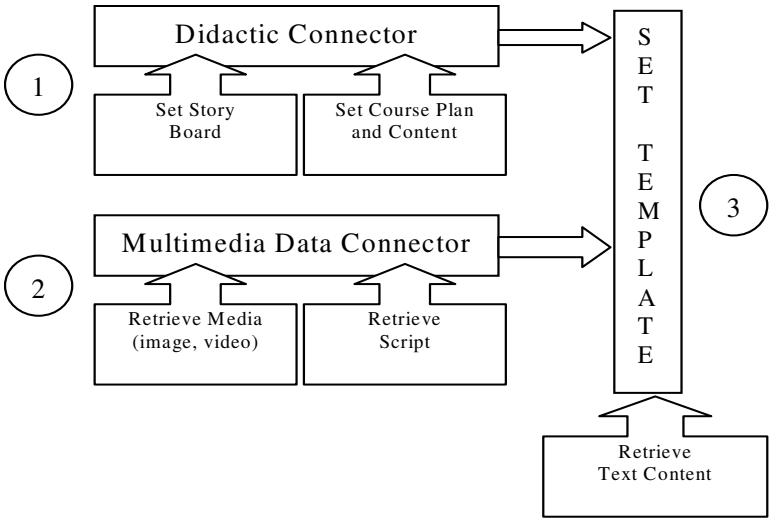


Figure 3. Architecture of Content Sub-system

The architecture of the Content sub-system is shown in the Figure 3. It is divided into several layers, which provide modular support for the different task for the framework shown in the Figure 2. The didactic connector component brings story board and content plan data into the framework. The component is also responsible for necessary interactions in line with the learning goals. The task of the connector is to integrate story board and course plan with the framework. The multimedia data connector component brings media data and scripts related to the required transformation of the media data into the framework.

The multimedia data connector abstracts from the access to media elements in different media storage and retrieval solutions. The template components collection provide layout and organization for the courseware to organize the didactic material of a course according to a particular instructional strategy. The template component structure organizes didactic material in chapters, sections, and concepts. A set of sections in relation to self-test and assessment formulate a chapter. A section contains multimedia document with motivation

and objectives, concepts, relationship among concepts and section summary. The template set interacts with the retrieve text content to generate presentation of content dynamically.

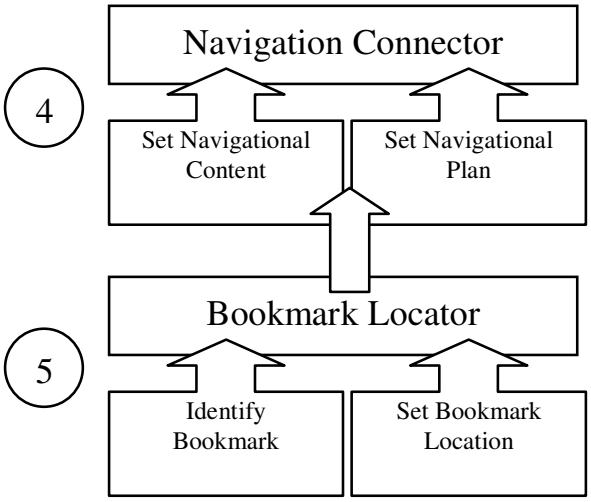


Figure 4. Architecture of Navigation Sub-system

The architecture of the Navigation sub-system is shown in the Figure 4. It is composed of Navigation connector and Bookmark locator to provide modular support for the navigation and bookmark task for the framework shown in the Figure 2. The navigation connector component brings navigation content and navigational plan data into the framework. The task of the connector is to integrate navigation content and plan with the framework. The bookmark locator connector component brings bookmark location data into the framework. This helps learners in progressively using a multimedia courseware and follow-up with the learning goals.

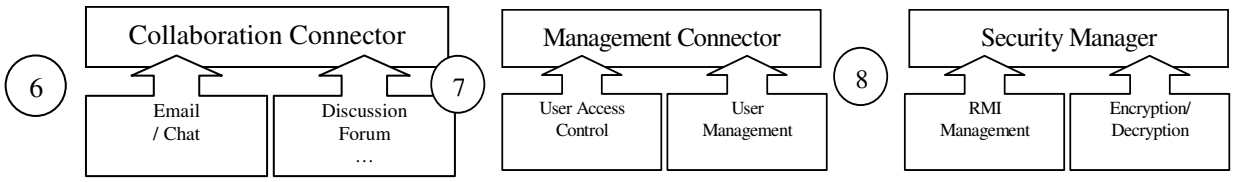


Figure 5. **Architecture of Collaboration and Management Sub-system**

The architecture of the Collaboration and Management sub-system is shown in the Figure 5. It is composed of Collaboration connector, Management connector and Security manager to

provide modular support for the required collaboration component e.g. Email, Discussion forum etc, User Access control, and security task for the framework shown in the Figure 2. The collaboration connector component brings collaboration components and tools into the framework. The management connector component brings user management with the access control into the framework. The security manager brings required security components into the framework.

4.0 Design Case Study

The department of Computer Science and Engineering offers Software engineering course primarily targeted to tenth semester students. The course components are: lectures, discussions, projects, case study and instructional materials. There is a three-hour lecture session and two-hour laboratory session allocated for the course every week. With this framework outlined in section 3 and shown in the figure 2 subsequently, we could easily prototype a multimedia courseware using open source technology. The courseware must be designed for students for maximum learning outcomes. For the architectural part of the course, it is required to capture online materials, case studies, assignments, team projects, past exam papers, online library repository and reference links from the central library of the university. The curriculum of the course has to be broken down into components and all the required components are required to be designed and implemented based on the proposed framework. In the process of developing and implementing the framework on the target platform, the adherence to development methodologies based on the software-engineering model comprising Analysis or Specification, Design, Development or Production, Implementation and Evaluation and finally accommodating feedback in the maintenance phase is essential.

In the specification phase, it is required to define the target audience and identify aims and objectives. The subject matter along with the pedagogical approaches and assessment methods are to be defined in this phase. Followed by the specification phase, identification of the contents for the courseware part and learning activities with the courseware must be accomplished.

The courseware components are structured for the access, layout, navigation as per the framework requirements. In the design part, all the text, graphics, sound, animation and

video are captured and necessary scripts are generated for the development phase. In the development phase, all the captured items are processed and organized according to the storyboard defined for the courseware. The courseware components are integrated and pilot test and evaluation is conducted with the learners and tutors.

5.0 Conclusions

With the framework presented here, we have aimed at defining in a simple way a methodology to systematize the development process of a course within the context of a typical university or tertiary level educational institution. As mentioned before, the proposed model is a formalization of a design and redesign process used by the authors on a software-engineering course. Part of the course design can be validated before implementing it —an action that can reduce both the product's development time and costs. In addition to the courseware, very important products that can be obtained by applying the framework is the course's activity plan or course schedule and synchronize with the learning goals.

Besides presenting the proposed model, the current paper has intended to demonstrate that the software and multimedia courseware developments are more interrelated than what most researchers think. This is perhaps a timely occasion to reuse the solutions found in other areas.

Acknowledgment

This work has been funded by the East West University Center for Research and Training (EWUCRT).

References

[1] Humphrey, W. (1995). *A Discipline for Software Engineering*. Addison-Wesley.

[2] Jacobson, I., Booch, G., Rumbaugh, J. (1999). *The Unified Software Development Process*. Addison Wesley.

[3] Boehm, B. (1988). *A Spiral Model for Software Development and Enhacement*. IEEE Computer, Vol.21, Nº5, pp. 61-72.

[4] Retalis, S., Makrakis, V., Skordalakis, E. (1997) EONT Courseware Development Methodology. Proceeding of EdMedia '97. Calgary. Alberta. Canada. June 14-19. pp. 1073-1079.

[5] E. Lippe. CAMERA, Support for Distributed Cooperative Work. 1992. SERC, Postbus 424, 3500 AK Utrecht, Netherlands.

[6] Landon, B. (2001). OnLine Educational Delivery Applications: A Web Tool for Comparative Analysis. Standing Commitee on Educational Technology; URL: www.ctt.bc.ca/landonline/

[7] McCalla, G. (1992). The Search for Adaptability, Flexibility, and Individualization: Approaches to Curriculum in Intelligent Tutoring Systems. M. Jones, P. Winnie (eds.). Adptative Learning Environments. Springer-Verlag, NATO ASI Series. pp.123-143.

[8] Paulk, M., Curtis, B., Chrissis, M., Weber. C. (1993) *Capability Maturity Model for Software.* CMU/SEI-93-TR-024. SEI. Carnegie Mellon University.

[9] J Vvan Ossenbruggen, F. Cornelissen, J. Geurts, L. Rutledge, and H. Hardman (2000), *Cuypers: a semi-automatic hypermedia presentation system,* Techn. Rep. INS-R0025, CWI, The Netherlands.

Appendix-A: CDM -Courseware Development Methodology

Multimedia and computer networks can be used in various ways for the implementation of learning environments in ODL. Within EONT project, such a learning environment, the EONT-ODL environment, was developed in the effort to evaluate the effectiveness of the use of these new technologies in ODL as a supplementary instructional delivery mode to the traditional one. Pre-and post-course questionnaires completed by students were the most important data sources for this evaluation study. It has been found that two predicting variables, the "design of instructional material" and the "preferred mode of study" explained 32.3% of the EONT-ODL environment effectiveness variability. While previous experience in computers and time spent working with EONT-ODL environment had not any significant impact on its effectiveness. The preliminary results presented in this paper, show that evaluation plays an important and decisive role in designing, developing and implementing teaching innovations.

Introduction: the Context of the Study

Providers of university education today are faced with the challenge of building an education system which could meet the current and future needs of society [Ford et al., 1996]. In this effort, open and distance learning (ODL), in particular based on multimedia and computer networks, has been witnessed an increased development, acceptance and recognition as an innovative and productive delivery mode of instruction and learning [Kaye, 1991; McConnell, 1991;1994; Riel & Harasim, 1994; Hiltz, 1995]. Indeed with the advent of the Internet, the World Wide Web (WWW), and the accompanying WWW browsers, the provision of ODL instructional material has taken on a whole new dimension [Maddux, 1994; Makrakis, 1996; Marshall & Hurley, 1996]. These technologies can be used in various ways for the implementation of learning environments in ODL.

One such learning environment is depicted in Fig. 1a [Koutoumanos et al., 1996]. This learning environment is being used in the EONT project. In this environment the instructional material is stored in a server computer and accessed by the learners through multimedia client computers connected to the server via a computer network. The heart of the learning environment is the hypermedia system HyperWave [Maurer, 1996].

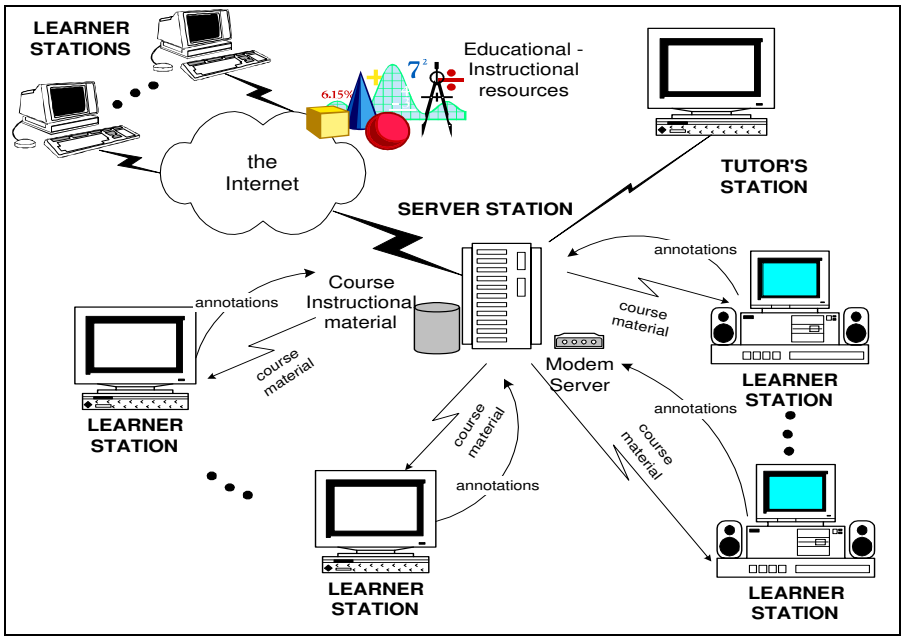


Figure 1a: Schematic view of the EONT-ODL learning environment to be used in the experiment.

EONT is a partnership project between seven universities from seven European Union countries within the Socrates Framework Program. Two of these universities are Distance Learning, whereas the rest are conventional. The partnership was formed on the basis of the partners' common interest in experimenting with ODL using new information and communication technologies. The project started on December 1st, 1995, and was completed in three years. For the purpose of the experiment, each partner developed multimedia instructional material within the domain of Informatics by adopting a common software engineering methodology [Retalis et al., 1997]. The language of each course is both the native language of the associated partner (native version) and English for the international version. The national version was offered once during the second year (1996/97) of the project and also offered once during the third year. The international version was offered online during the third year, as a means of providing learners in one partner's country with access opportunities to the course instructional material of the other partners. To assess whether the goals set were achieved, both formative and summative evaluation activities were conducted. These undertakings were considered as integral parts

of the whole development and implementation process, since they provide valuable insights and feedback to the development team for necessary changes and additions.

Appendix-B: Open Source and Open Standard

Open source software is based on open distribution of the source code that forms the software's foundations and specifically GNU general public license. This presumes that any technically competent programmer is able to examine the inner works of the source code, and potentially make changes to the operation to the software that best fits as per the requirements. Open source software is typically provided free of charge or with a nominal distribution cost. Some open source licenses require that any changes to the source code must be redistributed on the same open source license terms as the original source code.

On the other hand, Open standards are transparent descriptions of data and behaviour that form the basis of interoperability. Interoperability is the ability of different software systems to exchange information in such a way that they can both act in equivalent ways on the information, leading to equivalent user outcomes. In practice, interoperability means that users are not locked to any one software system – they can substitute a standards-compliant system for another standards-compliant system. Open standards can be implemented by commercial systems and open source systems alike, and provided that all systems adhere to the same standards, there is no impediment to environments which combine commercial and open source software systems.

So to take the first question, it may be natural to think that open source would be preferable to open standards if you were forced to choose between them. This is because in open source software development, all of the source code is freely available, and if it does not correspond to open standards, it can be modified to solve this problem. In the case of commercial systems, which support open standards, they rarely provide access to their source code, so external developers are not able to change the software as desired. Hence when forced to choose, open source appears the more flexible option.

In practice, it may not be so simple. The complexity of the open source code may be so great that a high level of technical expertise is needed to modify the code, and the cost of doing so (in time and/or money) may be great if the modifications are substantial. In addition, the changes may not be compatible with the intention of the original software, which can cause a "fork" in development that splits the original open source developer community into separate groups, potentially weakening both efforts.

Most open source e-learning projects have not arisen spontaneously from the goodwill of freelance software developers. They are typically the result of government or foundation funding where developers are paid for their contributions to the project (either as contractors or as salaried employees of organizations such as universities). In the wider open source movement, a voluntary community of developers supports projects such as Apache and Linux, and hence their ongoing development is independent of the vagaries of project funding. This is not the case in e-Learning, making any given open source developer community highly susceptible to collapse when project funding ends. This is a major problem, which is not well understood by governments or foundations, which provide current funding. The problem arises from the difference between "traditional" open source developer communities and funded-project developer communities.

Lack of interoperability in open source e-learning development can be illustrated by reference to the new IMS Digital Repositories Interoperability (DRI) specification and two recent e-learning initiatives. First, the OKI project is based on a range of open source e-learning service APIs, including a digital repositories service. OKI has collaborated closely with IMS over the past year – the same period in which the DRI was developed. However, at the February 2003 Vancouver IMS meeting, when asked about interoperability between the OKI digital repositories service and the new IMS DRI specification, it became clear that the two were not compatible at that time.

About this series…

Papers in this series are not formal publications of EWUCRT. They present prelude results subject to modifications and updating in future. These papers are circulated to encourage discussion and comment. Quotation and use of such papers should take into account of their provisional character. For copies of these papers, contact: EWU Center For Research And Training (EWUCRT), East West University, 45-46 Mohakhali C/A Dhaka-1212, E-mail:ewucrt@ewubd.edu

CENTER FOR RESEARCH AND TRAINING
43 Mohakhali C/A, Dhaka-1212, Bangladesh
Phone: 9882308, 9887989, 8814786, 8811381
E-mail:ewucrt@ewubd.edu
www.ewubd.edu