ICESEE: A Tool for Developing Engineering Courseware

Sergio Ochoa (+), José A. Pino (+), Nelson Baloian (+), and David Fuller (*)

(+) {sochoa, jpino, nbaloian}@dcc.uchile.cl
Computer Science Department
Universidad de Chile

(*){dfuller}@ing.puc.cl
Computer Science Department
Pontificia Universidad Católica de Chile

Abstract

In this article, a computer tool is introduced to support courseware building and evolution in computer science and engineering education. The tool is intended to fulfill requirements, which are typical of this area of education. The tool has been implemented and initially tested. Instructors and students have been satisfied with the tool, therefore indicating encouraging results.

Keywords: Computer Science and Engineering Education, Courseware Evolution, Development Tool.

1. Introduction

Many engineering educational institutions use courseware to support the teaching process nowadays. This courseware provides an educational environment comprised of didactic material and communication/collaboration tools. Usually, this didactic material is multimedia and interactive, and involves planned activities to carry forward the teaching-learning process –discussions, self-examination, laboratories, and so forth. Moreover, the communication/collaboration tools facilitate the interaction among instructors, assistants, and students.

Due to the fast and continuous technological development in fields such as computer science and engineering, courseware for teaching should meet additional requirements that are not present in fields

with a slower path of development. These requirements have revealed the restrictions posed by most current courseware development tools [16]. These restrictions have to do, among others, with the complexity involved in the courseware building and evolution process, and the lack of support needed to perform their reengineering and to incorporate the typical applications for the application domain. For this reason, a tool called ICESEE (Integrated Collaborative Environment to Support Engineering Education) has been introduced, and has been designed to support courseware building and evolution in the aforementioned fields. In general, the tool takes into consideration the typical requirements of courseware and particularly those present in computer science and engineering courseware. Although it might be extended to other engineering fields, ICESEE has been tested in the specified domain only.

The following section contains a description of the typical requirements for computer science and engineering courseware. The available tools for courseware building are introduced in Section 3 as related works, along with the strengths and weaknesses of tools to handle the described requirements are discussed in Section 2. Section 4 presents the proposed tool, as well as its features and basic components. In Section 5 the courseware building/evolution process using ICESEE is described as well as the solutions incorporated in this tool in order to deal with the items specified in Section 2. Section 6 shows preliminary results obtained from computer science and engineering courses using this tool, and finally a discussion and the conclusions of this work are presented in Section 7.

2. Computer Science and Engineering Courseware Requirements

A description of the requirements that a courseware for teaching should fulfill is listed below, with a focus on fields of fast evolution. For these requirements, there is a wide range of software already developed, which can be used to support learning. These requirements are in addition to those that should be commonly present in any courseware building and administration system.

- 1- *Courseware not only has to be easy to build, but it also must evolve easily.* In order to build courseware, the participation of a multidisciplinary team is generally recommended, including the course instructor, psychologists, and experts in instructional design, computing and graphic design [27]. Due to the fast evolution of courseware in the engineering field [13], interdisciplinary work is impractical because it requires continuous commitment of all those involved. Despite the fact that an interdisciplinary team could create certain courseware, its evolution is generally in the hands of instructors and their assistants. These tasks are usually performed with limited time and scarce resources. Furthermore, they demand constant redefinition based on the obtained feedback [8, 27]. Thus, courseware in these fields should be easy to build and evolve. The achievement of this objective mainly depends on the capabilities of the development tools entailed.

- 2- *Courseware should easily integrate specialized educational components.* Learning in these fields normally implies the execution of experiments, calculations, problem analysis through variables, program designs, etc. [3, 23]. Therefore, courseware should be able to include, as part of its didactic material, components such as simulations, worksheets, computer programs, etc., plus typical capabilities such as handling audio/video, images, and text. This requirement imposes two conditions on the courseware development tools. In the first place, these tools should have standard components, which are typical of the application domain [23]. Thus, there is no need for the instructor to develop the components, in case he or she wants to make use of them. Secondly, these tools should somehow anticipate the inclusion of external components –possibly non-standard– within the didactic material [23]. Many times an instructor wishes to incorporate special components (e.g. specific simulation software), so these tools should provide a way to carry this out. Hence, the evolution of didactic material does not depend on the components included in the utilized development tool.

- 3- *Courseware should be able to embed instructional models.* According to our experience, many instructors in these fields are not aware of the pedagogical aspects implied in knowledge transmission. Consequently, the courseware didactic material and activities could be wrongly structured from an instructional point of view. On the other hand, there are instructional models that have produced good results for the organization and transmission of knowledge in engineering, such as *Problem-Solving* [4, 7], *Meaningful Learning* [21], or *inquiry-based learning* (e.g. [14, 24]). Courseware should have the possibility to embed some of these models. Thus, the instructor would have a reference point to adequately organize knowledge and its transmission. This means that the development tools involved should be able to embed instructional models in the courseware, which have been proved in the application domain [2]. In addition, they should provide enough flexibility to allow the instructor to adapt models following his or her own criterion.
- 4- *Courseware should support its own reengineering.* The pace of knowledge evolution in computer science and engineering is fast, which means courseware in this field has a high update rate [12]. The two aspects involved in courseware reengineering are the evolution of knowledge delivered through it, and the improvement in achieving planned objectives. The instructor can manage the former, for he or she is an expert in the matter. The latter, however, involves aspects that are beyond the instructor's field of expertise, and requires information –generally statistical– which is hard to obtain. That is why it is necessary to assist the instructor with objective information, intended to help him or her to perform this task. This information should include number of accesses, percentage of reviewed material, total connection time, number of –successful and unsuccessful– self-examination resolutions. In addition, this information should be grouped or disaggregated depending on the instructor's information needs. Courseware should capture this information. Therefore, courseware development tools for the defined domain should be able to embed the necessary mechanisms to accomplish the delivery of this type of information in a courseware [1].

3. Related Work

Many tools concerning courseware building have emerged lately. They are known as *courseware tools* [16]. Some of them are *Lotus Learning Space*, from IBM Lotus Corp [17]; *WebCT*, developed at University of British Columbia [31]; *TopClass*, from WBT Systems –previously known as WEST– [26]; *Virtual-U*, developed at Simon Fraser University, [28]; *WCB*, from Virginia Commonwealth University [30]; *CourseInfo*, from Blackboard Inc [6]; *ToolBook II*, a set of commercially available products from Asymetrix Learning System Inc. [25]; and *Luvit*, developed at Lund University [18]. All these tools have been designed to support standard courseware building, which does not belong to any particular field. Consequently, they do not consider most requirements mentioned in the previous section. For example, most existing tools do not provide additional assistance for the reengineering stage (requirement 4), so the instructor is required to have the necessary knowledge to do it. There are tools, such as WebCT, which provide much information regarding the use of courseware –number of accessed pages, number of accesses per type of browser, and per person, among others. Nevertheless, these tools supply little or no information on statistics related to the results of the teaching-learning process supported by the courseware. Examples of these statistics are the number of approved/failed exams and the number of questions/answers generated at the discussion forum following various classification criteria. This information could help the teacher when performing the reengineering tasks.

Conversely, traditional courseware tools do not allow for embedding any specific instructional model (requirement 3), since they are intended to support education in general. This forces instructors from different fields to completely design the instructional model that will be used in each courseware. Also, these tools do not include specialized components for different application domains (requirement 2), and so the didactic material is impoverished, while the associated development effort increases.

Normally, these tools include general components that, to a certain degree, simplify the courseware building and evolution process (requirement 1), but they have not reached yet the ease of use required by instructors or assistants. Due to the lack of compliance with the rest of requirements specified in Section 2, it is possible to declare these tools are inadequate for courseware development in computer science and engineering.

On the other hand, there are tools such as *SimulNet* [Ani00], *Virtual Apparatus* [11], and *OXIGEN* [12], which also allow for the creation of courseware. These, unlike the traditional courseware tools, follow a totally different building strategy. Their construction essentially involves the building of courseware through the assembly of prefabricated modules, called *components*. These are much like *software components*, and may be visualized as building blocks [Pre00]. These tools are better prepared for the development of courseware in computer science and engineering. Their development strategies facilitate the building and evolution processes of computing applications (requirement 1) [23, Pre00]. In addition, they make it possible to incorporate any external component following the same specification (requirement 2, partially) [Pre00]. Some of the most common specifications are LOM [9], IMS [10], and the traditional specifications related to software components, such as JavaBeans, ActiveX, COM, etc.

The lack of support to manage the remaining requirements specified in Section 2 is among the most important constraints of these tools. That is why, despite being better prepared than the traditional courseware tools, they are not suited for courseware development in this field.

In order to provide a solution for this problem, the first version of ICESEE was built in 1997. This tool introduces a strategy that handles all four requirements mentioned in Section 2, in addition to the standard requirements, and intends to be adequate for the achievement of this task. This tool is described in the following section.

4. ICESEE System Description

ICESEE was conceived as an educational environment for the implementation of computer science and engineering courseware in a traditional university. The primary goal was to develop a tool not only to support planning and delivery of this courseware, but also to allow its incremental reengineering (evolution). Because ICESEE is a collaborative tool, it makes it possible for instructors, assistants, and students to work at the same time, without interference. Furthermore, because ICESEE was developed with Lotus Notes [19], it inherits some important benefits from this platform. These are robustness, high performance, ability to deal with different protocols, and event synchronization, among others.

ICESEE work strategy is simple. It considers the reutilization of the entire infrastructure of standard courseware, which is automatically generated when creating the courseware (requirement 1, partially). Therefore, courseware building and evolution is reduced to the building and evolution of the didactic material. This means ICESEE automates a great deal of the courseware generation, limiting the consideration of the four defined requirements only to the didactic material building and evolution. In order to execute these tasks, the tools make use of a building strategy based on components. Consequently, it facilitates the product creation and evolution process [23, Pre00] (requirement 1).

Many of the components used to build the didactic material have special capabilities, mainly because they have to support activities that belong to the application field. In addition, there are other external components –specialized or non-specialized–, which can be utilized by ICESEE for the building or evolution of didactic material, as long as they follow the same specification (requirement 2). In order to comply with requirement 3, ICESEE provides templates that reflect instructional models proven in the engineering field. These templates help the instructor to organize the didactic material in order to facilitate the student's approach to it. Finally, many of the components utilized for the building of

courseware, as well as subsystems, have the capability of automatically storing information about their usage. ICESEE, on the other hand, has the ability to retrieve this data, and to automatically generate a set of predefined information –mainly statistics regarding the courseware usage. Hence, the fourth requirement is fulfilled.

Courseware built with ICESEE is composed of three different macro-components or subsystems, *Contents*, *Collaboration* and *Administration*, and one courseware's *Navigation System* (Figure 1). The contents subsystem stores the courseware didactic material. The collaborative subsystem provides support to perform communication/collaboration activities among instructors, assistants and students. The administration subsystem is in charge of user administration, collection of statistics about courseware usage, and administrative information publication.

Figure 1. ICESEE Working Scenario.

Ochoa S., Pino J.A., Baloian N., Fuller D. (2003) ICESEE: A tool for developing engineering courseware. Computer Applications in Engineering Education, 11, 2, 53-66.

When the instructor creates a courseware, a new instance of it is automatically generated. This instance contains all 3 subsystems aforementioned and the courseware navigation system. From then on, a courseware building/evolution task is reduced to the building/evolution of the didactic material. Section 5 describes in detail the building process of didactic material.

Once the didactic material has been built, courseware is published so it can be accessed through the Internet. Then, it is used to support course. At the end of the course, the administrative subsystem contains information regarding the results of the finished process, which helps in the evaluation of the course objectives and to improve the courseware itself. This evolution is achieved through the redefinition of didactic material. The following three sections briefly introduce each of the subsystems that constitute the courseware infrastructure.

4.1. Contents Subsystem

This subsystem is a repository of multimedia and interactive didactic material, designed to carry forward the teaching-learning process. The main objective of this subsystem is to store, by means of different structures, the contents and activities to be developed in a course. This repository may be accessed through the Internet and through ICESEE.

Figure 2. Example of Course Content Subsystem

Figure 3. ICESEE in Authoring Mode

Students can only interact with the courseware through a WWW browser (Figure 2). Instructors and assistants, on the other hand, can also access it through ICESEE. As an authoring tool, ICESEE's main task is the implementation and evolution of didactic material, bearing in mind the requirements described in Section 2. Figure 3 displays the ICESEE interface used by authors to perform the building and evolution of didactic material.

4.2. Administration Subsystem

This subsystem implements three functions: (a) presentation of administrative information, (b) management of users, and (c) handling of information –mainly statistics– about courseware usage. Function (a) publishes the relevant information about the implemented course (Figure 4); that is, goals, contents, instructors and assistants, class schedule, news, grades, etc. It also implements a small library with reference material such as books, papers and URLs, which may be of interest for the students.

Figure 4. Administration Subsystem Interface

Administration subsystem function (b) consists of tools for the creation of user accounts and definition of user roles. The roles of a user in the system may be as an *instructor*, *assistant*, or *student*. Each role indicates the relationship between user and courseware. It consists of a set of grants assigned by default to a user, which can be modified by the instructor.

Statistics about the system usage (c) are valuable input for the instructor in order to perform the courseware evolution process. Among other relevant data, the system provides the access frequency to the courseware and to each chapter, the number of times students tried to solve an evaluation or self-examination –successfully or unsuccessfully–, the number of messages sent/received among users or, in a discussion forum, the percentage of material reviewed by each student, total connection time to the courseware, among others. These values may be requested for a certain period, for a student, or for a group of students (Figure 4).

4.3. Collaborative Subsystem

This subsystem has three tools, which implement collaborative activities, namely *chat*, *e-mail*, and *discussion forum*. The first one is for synchronous work while the other two are for asynchronous interaction. Naturally, these tools are, in a strict sense, just for implementing communication and not collaboration, but in this scenario we expect the exchanged messages to contain some information or knowledge that has a direct impact on the receiver's knowledge and generates some collaboration among participants.

Chat. The chat tool is for implementing an informal dialogue (conversation) among students, instructor, and assistants. Within the chat the information is not persistent –it vanishes after the session is over. Different chat sessions are organized as different rooms to which users may access. Rooms may be

created and named according to the topic being discussed. A user can only take part in one conversation at a time and will only receive the messages generated in that room.

E-mail. The email tool has the ability to send/receive e-mail messages and to communicate with external mail servers (POP3 and MAPI). However, the first experiences with this system show it would be better to implement it as an internal mail system in order to avoid interfering with the normal –perhaps less formal– e-mail system.

Figure 5. E-mail Tool Interface

Figure 6. Asynchronous Discussion Tool Interface

This tool keeps received and unsent messages in separate folders. It can also handle e-mail messages for a single person or a group, and introduces an alarm system to remind the user of important messages or deadlines (Figure 5). It also includes a calendar, a trashcan, and an agenda to record tasks and meetings. A planned feature for the internal e-mail system is the grouping of messages according to course topics.

Discussion Forum. The discussion forum is a tool that was included to keep an organized record of the discussions among participants. This not only helps the students to exchange opinions and ask for help or feedback, but also helps the instructor to monitor the quality of collaboration in terms of learning. The discussion forum has been structured following the IBIS model [15]. According to this model, a discussion consists of *issues*, *positions*, and *arguments*. Issues are questions that participants can answer

with positions, which in turn can be supported by arguments. Issues about related topics are grouped as shown in Figure 6.

The discussion forum tool allows the display of the discussion in different formats in order to follow the process of formation of different points of view. The discussions are permanently stored in order to keep this information available for the current and future students of the course. The following section describes the courseware building/evolution process using ICESEE.

5. Building/Evolution of Didactic Material

As mentioned above, when ICESEE is used, the courseware building/evolution process is limited to the building/evolution of didactic material. It greatly facilitates the work of the instructor, but it does not mean that this task will become trivial.

The instructor begins the building process with a definition of the didactic material structure. The instructional model to be used might be suggested, or it can be created by the instructor according to his or her expectations and needs. In the first case, the instructor only indicates the model to be used. Then, ICESEE creates the structure of the courseware didactic material using the template associated to this model. In the second case, the instructor may define a template according to his or her own criteria, or indicate the courseware will have a free format –without restrictions. The structure proposed by the templates conditions the organization of the didactic material, as well as the dynamics of the teaching-learning process (requirement 3). This is reflected in the didactic material consistency.

Once the template has been selected or defined, the instructor indicates the number of chapters the courseware will have, the subject matter of each and the way they are composed. The same thing has to be done with the remaining courseware components. For example, when the instructor creates a new

chapter, the template's sole purpose is to automatically create its internal structure –sections, self-examination, summary, etc.– as specified in the template. Although these components will be empty, they show the instructor the structure to be observed. The result of this task is an empty framework of the courseware didactic material. All that remains in the process is to specify the components of the framework. There are two possible ways to do this. The first is to create the contents of each component, using smaller components –text, image, audio/video, tables, etc. The second, and frequently used, is reusing existing components, adapting them as needed. Some of these components are standard, general parts and others are specialized in performing tasks that belong to the application domain (requirement 2).

Users utilizing the role of instructor can edit these components, which allows content adaptation as needed. Instructors from other courseware can also copy and paste them in courseware where they have the instructor role. Hence, the didactic material stored in the courseware accessible to the instructor may be reused, in an organized manner. This greatly facilitates the task of courseware building and evolution (requirement 1). Once this task has been completed, the result is a courseware that will be published so it can be accessed through a browser.

The courseware is then used to support a course. Throughout this process some of its components will store information about its usage, which will be useful afterwards to identify items to be improved. This, in turn, serves as raw material to perform the reengineering or evolution of didactic material (requirement 4). The reengineering process starts with a completely implemented courseware. Therefore, redefinition can take place over the framework structure, the contents, or both. Structure reengineering may include a template change, and in order to accomplish this, all the previous structure is translated into an equivalent representation over the new template. It may also involve the internal restructuring of components; for example, add, remove, or integrate chapters or sections. While performing this task, component reutilization could facilitate the instructor's work.

Conversely, contents reengineering is related to the modification of knowledge stored in the didactic material. The explanation for this might be that the contents were not adequate to reach the defined objectives or simply due to the natural evolution of knowledge. Whichever the cause, this task involves the building or update of contents. This is accomplished in a similar way to the building process of didactic material. The following sections introduce ICESEE templates and components.

5.1. Templates

Templates offer a way to organize the didactic material of a course, according to a particular instructional strategy. Generally, templates are defined by experts in instructional design and reused by instructors. Their objective is to automate the organization of knowledge in a courseware, and to structure it in order to facilitate the way it is approached. Thus, the instructor is freed from the task of designing an instructional strategy for each courseware. Until now, only two templates have been defined in ICESEE; the *meaningful learning template* and the *problem-solving template*. Figure 7 displays the template designed following the guidelines of an instructional model known as *meaningful learning* [21]. The proposed structure for the template may be modified by the instructor according to his or her needs and expectations.

Figure 7. Meaningful Learning Template

In the case of the meaningful learning template, the proposed structure organizes the didactic material in chapters, sections, and concepts. Chapters consist of a set of sections, a self-examination, and an evaluation. A section consists of a multimedia document with four parts: *motivation*, *set of concepts*, *relationship among concepts*, and *summary*. At the beginning of each section, there is a *motivation* message from the instructor. Then, the concepts included in the section are introduced. Once the concepts have been presented, it is necessary to show the relationships among them. Examples, conceptual maps or problem solving, among others, may be presented in order to achieve this.

Usually, at the end of each chapter there is a self-examination and an evaluation for the student. The self-examination consists of a window with check boxes and/or pop-up menus for selecting the answers to a set of questions. The self-examination goal is that students receive feedback about their own knowledge, before new concepts are presented to their mind scheme. Depending on the degree of inaccuracy of an answer, system-delivered feedback could recommend the student to review some concepts, sections, or the whole chapter. To access the next chapter, it is necessary to correctly answer the self-examination.

The final evaluation of the chapter could be the resolution of a problem, a research paper, or a task that the instructor considers adequate to measure the knowledge acquired by the students. These evaluations may be done individually or in groups.

The problem-solving template has a similar structure to the above –meaningful learning template– although it follows a different instructional strategy. Find below an introduction to the didactic components defined in ICESEE, which are useful when building a courseware didactic material.

5.2. Didactic Components

In ICESEE, there are three types of didactic components defined: *basic components*, *content components*, and *activity components*. The basic components defined are *text*, *image*, *audio*, *video*, *multimedia area*, *table*, *vignette*, *spreadsheet*, *applet*, and *external program*. This last component involves any external computer program, whose execution can be displayed in an html page by means of a plug-in. These components constitute the raw material for the building of the didactic material.

On the other hand, the purpose of the content components is to facilitate the creation and publication of didactic material with a higher granularity than the basic components. Current content components are *chapter*, *section*, *subsection*, *concept*, *example*, *motivation*, and *summary*. Each of these components essentially represents a piece of the courseware structure. Finally, included activity components are *exercise*, *evaluation*, *discussion*, and *brainstorming*. Their objective is to facilitate these activities with the courseware. Thus, these components are also part of the contents. Find below the content components and the activity components. Basic components will not be defined because they represent simple and very well-known concepts.

Chapter. The chapter macro-component is intended to orderly store and publish the courseware didactic material. It consists of four components: *motivation*, *set of sections*, *summary*, *self-examination*, and *evaluation*. Each of them may or may not be displayed, depending on the selected template. The same occurs with the remaining components. The motivation included in the chapter helps students to understand the scenario in which the topic covered by the chapter is inserted. The sections that comprise a chapter represent the set of contents and activities designed to achieve the goals of the chapter. The summary emphasizes the most important aspects of the chapter and the self-examination lets students evaluate the knowledge acquired with the chapter. The final evaluation is used by the instructor to evaluate the students' knowledge.

Section. The section component represents a set of related themes and consists of *motivation*, *set of subsections and/or concepts*, and *summary*. The objectives of motivation and summary are the same as those described above for the chapter component. Evaluations have not been included in these low granularity components of a chapter in order to avoid overloading the students.

Subsection. The subsection component represents a set of related concepts, and it consists of *motivation*, *set of concepts*, *examples*, and *exercises*; however, it does not contain a summary. The subsections are seldom used, essentially because they make it difficult to navigate and access information. In general, students prefer to have longer items and to eliminate the subsections. In any event, there are times when this is not feasible and subsections must be used.

Concept. The concept component represents a specific concept presented by the instructor. It consists of *motivation*, *content*, and *conclusion*. The motivation is the same as in previous components. The content is a collection of basic components, examples, exercises, discussions, and brainstorming.

Figure 8. Example Component Included in ICESEE

Figure 9. Example of a Self-Examination

Example. The example component consists only of content, which, as in the previous case, is a collection of basic components organized according to the criteria established by the instructor (Figure 8). The examples serve to apply the theoretical concepts acquired by the students and to visualize the relationships among them.

Motivation and Summary. Some aspects of *motivation* and *summary* have been presented during the definition of the previous components. The motivation should be short and may be implemented using text and audio. The summary can be longer than motivation, and may include text and images.

Evaluation. The evaluation component represents an activity in which the students should demonstrate their knowledge on a topic proposed by the instructor (Figure 9). The students can respond in groups or individually, depending on the rules established by the instructor. The instructor can design the characteristics of the evaluation using predefined templates or a combination of them. The templates

used for evaluations are tests, self-examination, questions, problem solving and research. Typically, this activity is carried out in an asynchronous manner.

Exercise. The exercise component can present problems yet to be solved, questions yet to be answered or a research to be performed. It consists of *presentation* and *set of questions*. The presentation is a content that, just like the evaluation, contains an organized collection of basic components. The point of the presentation is to describe the problem and its setting. The questions express requests that must be fulfilled by the student in relation to the presentation. These components have the ability to store information about usage.

Discussion. The discussion activity allows the instructor, assistants and students to discuss a topic that is generally proposed by the instructor. This activity uses the IBIS model [15] and can be performed in an asynchronous way. This component stores information about its use.

Brainstorming. The brainstorming activity is used to explore new ideas in order to complete a task or solve a problem and to explore other's viewpoints. This activity follows a sequence of several steps to reach the final goal [22] and may be performed using different styles such as free, chauffeured, and so forth. For example, the instructor can include a brainstorming session as part of a group problem solving process. This component stores information about its use.

6. Results Obtained

This section presents the results obtained with ICESEE usage in some university level courses of computer science and engineering. Two kinds of evaluations were made. The first evaluation tested the adequacy of the proposed tool in supporting courseware development and evolution in the specified

domain. The second evaluation tested the quality of courseware developed with ICESEE. The next sections present and discuss these evaluations.

6.1. ICESEE Evaluation

In order to evaluate the adequacy of the tool, the eight items proposed by Nielsen [20] were employed to measure usability (1-8), plus four additional points intended for measuring work capabilities in the fields of computer science and engineering (9-12). The elements considered are presented in Table 1.

5- Use of simple, natural language.	1- Provision of shortcuts for experienced users.
6- Minimization of required memory load from the user.	2- Informative error messages.
7- Consistency in terminology and symbols used throughout the interface and compliance with the domain norms and standards.	3- Prevention of errors.
8- Degree and quality of system feedback.	4- Provision of needed didactic components to develop or evolve a courseware.
9- Provision of clearly marked exits and undo.	5- Provision of sufficient information to evolve a courseware.
	6- Didactic material easy to build and evolve.
	7- Useful templates.

Table 1. Evaluation Items

The participants in the evaluation were instructors from three courseware implemented using ICESEE, and three experts in graphic interfaces. Instructors rated every item, while experts only assessed the first eight. They were provided with an evaluation sheet in which they could assign marks and weight to each item. The mark is a value that qualifies an item and the weight indicates the importance of it according to the instructor or expert. The range of assigned values was from 0 (worst) to 10 (best), including decimals. The results obtained are shown in Table 2.

		1	2	3	4	5	6	7	8	9	10	11	12
Expert	Average	9.0	7.5	9.5	7.0	8.5	6.0	7.5	7.0	N/A	N/A	N/A	N/A

Marks													
Expert Weights	Average	9.5	8.5	8.1	7.8	8.1	6.2	6.0	5.0	N/A	N/A	N/A	N/A
Instructor Marks	Average	9.2	7.8	9.4	6.5	7.8	6.1	6.8	6.5	9.0	9.0	9.2	9.5
Instructor Weights	Average	9.0	8.7	8.0	7.0	7.5	5.0	5.4	4.5	9.1	9.3	9.5	9.3

Table 2. Results of ICESEE Evaluation (experts and instructors)

The results are still not enough to make definite conclusions. However, they indicate ICESEE has provided satisfactory support for the building and evolution of courseware in the fields involved. All items with a weight higher than 8.5, also have a rating above this value, except for item 2. This means that the tool covers the most relevant aspects according to courseware developers. The most outstanding items were 1, 9, 10, 11, and 12. These were directly related to the capabilities required to develop courseware in computer science and engineering, which were identified in Section 2.

6.2. Courseware Evaluation

Experiments were conducted with ICESEE in a university environment starting the second term of 1998. At that time, a courseware for a course called *Software Engineering Laboratory (S.E.L.)* was built. In that course, students must develop a software project according to a standard by working in teams. Three reengineering processes have been applied to this courseware. Another two courseware have been built; namely, *Computing Introduction (C.I.)* and *Groupware Laboratory (G.L.)*. Although the first reengineering has taken place for both, this task has not yet been evaluated.

Course – Term / Item	S.E.L.				C.I.	G.L.
	2 nd Term – 1998	1 st Term - 1999	1 st Term - 2000	1 st Term - 2001	2 nd Term – 2000	2 nd Term - 2000
Number of Students	7	12	27	50	55	26
Number of Projects	1	2	3	5	N/A	3
Grade Average	7.5	8.1	8.5	8.7	7.9	8.0
Template	Meaningful Learning	Meaningful Learning	Meaningful Learning	Meaningful Learning	Problem-Solving	Problem-Solving
Mark / Weight (1)	8.5 / 9.1	8.6 / 9.2	8.5 / 9.0	8.7 / 9.1	8.6 / 9.2	8.8 / 9.1

Mark / Weight (2)	7.0 / 8.5	7.8 / 8.4	8.0 / 8.3	8.5 / 8.2	7.5 / 8.4	7.0 / 8.3
Mark / Weight (3)	7.8 / 7.9	8.0 / 7.8	8.0 / 7.9	8.1 / 7.7	7.5 / 8.0	7.7 / 7.9
Mark / Weight (4)	5.0 / 8.0	6.0 / 8.1	6.5 / 7.9	6.8 / 8.0	6.2 / 7.9	6.4 / 8.1
Mark / Weight (5)	8.5 / 5.0	7.8 / 5.2	8.4 / 5.0	8.6 / 5.3	7.5 / 5.0	7.8 / 5.4
Mark / Weight (6)	6.0 / 5.5	5.8 / 5.6	6.1 / 5.3	6.2 / 5.3	6.0 / 5.1	6.0 / 5.0
Mark / Weight (7)	6.5 / 6.0	6.3 / 5.5	6.5 / 5.3	6.2 / 5.1	6.4 / 5.2	6.1 / 5.0
Mark / Weight (8)	6.1 / 5.0	6.1 / 4.7	6.2 / 4.9	6.1 / 5.0	6.0 / 5.1	6.0 / 4.9
Mark / Weight (9)	8.5 / 9.5	8.9 / 9.5	9.0 / 9.3	9.2 / 9.4	8.8 / 9.3	9.0 / 9.5
Mark / Weight (10)	8.7 / 9.2	8.9 / 9.1	9.1 / 9.0	9.3 / 9.2	8.9 / 9.0	8.8 / 9.3
Mark / Weight (11)	8.7 / 9.0	8.8 / 9.1	8.8 / 9.2	8.7 / 9.1	8.5 / 8.9	8.4 / 9.0
Mark / Weight (12)	9.0 / 9.3	9.1 / 9.3	9.2 / 9.4	9.2 / 9.0	9.1 / 9.2	9.3 / 9.0
Satisfaction (13)	7.8	8.1	8.3	8.6	7.7	7.9

Table 3. Experimental Results of Courseware Evaluation (students)

Table 3 is divided in two parts. The first part shows informative data about the courses involved in the evaluation. The second part shows the average marks and weights assigned by students in relation to the 13 items linked to the quality of courseware developed with ICESEE. The first eight items are those defined by Nielsen, and the remaining five were defined by the development team, based on their experience. These items are the following:

- 9- Organization of the didactic material.
- 10- Quality of the didactic material.
- 11- Method of examination.
- 12- Navigation model.
- 1- Level of satisfaction.

Table 4. Additional Evaluation Items

In relation to data obtained through courseware evaluations (Table 3), a few comments can be made. After each reengineering applied to *Software Engineering Laboratory (S.E.L.)* courseware, a positive evolution can be observed in the indexes. Another highlight is that the two templates implemented have shown similar results for the first courseware version. Besides, according to the students, these templates have organized the material in an appropriate manner.

Almost all items with a weight higher than 8.5, also have a rating above this value. This means the most relevant aspects of a courseware from the students' viewpoint have been covered. The most important items according to the students are 1, 9, 10, 11, and 12; therefore, work has to focus mainly on these items, in order to improve courseware. The initial values for the three analyzed courseware have been acceptable, showing certain similarity among them. This could mean the basic quality of courseware developed with ICESEE may be closer to the values that appearing in Table 3. Finally, it can be said that no improvement could be observed in items 5 through 8. This is because said improvement is related to the improvement of ICESEE's didactic components and, until now, this subject has not been overtly covered.

7. Discussion and Conclusions

This paper introduced a courseware development tool called ICESEE. This tool has been designed to support the building and evolution of courseware in the fields of computer science and engineering. Courseware for these subjects has the following requirements: (1) it should be easily built and evolved, (2) it needs specialized educational components, (3) it should be able to embed instructional models, and (4) it should support its own reengineering. Traditional courseware tools do not comply with these requirements; generally, they only handle the first one. Conversely, building tools based on components cover requirements 1 and, in part, 2. This means that none of these tools is adequate to fully support the development of courseware in computer science and engineering.

The proposed tool incorporates these four requirements. Specifically, the integration of requirements 3 and 4 make it possible for ICESEE to differentiate from the remaining available tools. In order to facilitate courseware building and evolution (requirement 1), ICESEE uses a set of prefabricated didactic components, component-based building, and automatic generation of a great part of the courseware. In addition, it reduces the courseware building and evolution process to the building and evolution of the

didactic material. ICESEE provides a set of specific components for computer science and engineering, so courseware in these fields could have specialized components (requirement 2). Despite the fact that several components represent general concepts, the capabilities they include have been specifically designed to support teaching in this field. In addition, it allows for the inclusion of a wide range of external components, which could be developed by third parties.

ICESEE provides templates that help the instructor to organize the didactic material in order to facilitate the students' approach to it, so the courseware can embed instructional models (requirement 3). Finally, ICESEE retrieves the information regarding courseware usage and preprocessed delivery, so it can support its own reengineering (requirement 4). The analysis of this information allows the identification of the weaknesses, which are the raw material for the execution of courseware reengineering based on its own results.

In order to test its features, ICESEE has been used in three courses of computer science and engineering. Several conclusions arise from the results obtained. The main conclusion is that the proposed tool is adequate for supporting courseware building and evolution in the specified domain. Moreover, the developed courseware showed to be appropriate to support the subjects taught in the aforementioned fields (according to the students' appraisal). Although these results are still premature, the plan is to continue experimenting with ICESEE in courses with large number of students in various fields of knowledge.

The main constraint of ICESEE is that it automates a great deal of a courseware building and evolution process. This reduces flexibility of courseware implementations. For this reason, at this moment we are testing a tool called CDS (Courseware Development System), which transforms the didactic components in reusable software [23]. In CDS, each figure, section, or courseware is a software component; therefore,

the range of solutions to be implemented may be as flexible as one dictates. Consequently, it will reduce time and costs associated with the building of a courseware, and increase the reutilization of didactic material.

Acknowledgements

This work has been partially funded by Fondo Nacional de Ciencia y Tecnología de Chile (FONDECYT) grants No. 103-0959, 198-0960 and 100-0870, and the Scholarship for Doctoral Thesis Completion.

References

- [A00] Anido, L., Llamas, M., Fernández, J., Caeiro, M., Santos, J, Rodríguez, J. *A Component Model for Standardized Web-based Education*. Proc. of the 10th WWW Conf. Hong Kong (May 2001). 86-95.
- [1] Baloian, N., Fuller, D., Ochoa, S. *A Model for Component-Based Courseware Development (CBCD)*. Proc. of the Int. Conf. on Soft. Eng. and Knowl. Eng. (SEKE'01). BS AS, Argentina. (Jun. 2001).
- [2] Benyon, D., Stone, D. *Experience with Developing Multimedia Courseware for the World Wide Web: the need for better tools and clear pedagogy*. IJHCS. 47, (1997) 197–218. Academic Press.
- [3] Bringslid, O. *Multimedia Books in the Mathematical Education of Engineering*. European Journal of Engineering Education. 24, 12 (Jun. 1999). 189-199. Carfax Publishing. Taylor & Francis Ltd.
- [4] Clap, T., Slocum, M. *Theory of Inventive Problem Solving Pedagogy in Engineering Education, Part II*. The TRIZ Journal, (Dec. 2000). The TRIZ Institute, CA, USA.
- [6] Blackboard, Inc. (Jul. 2001). *CourseInfo*. URL: product.blackboard.net/courseinfo/
- [7] Frye, E. *Engineering Problem Solving for Mathematics, Science, and Technology Education*. Dartmouth Project for Teaching Engineering Problem Solving, Dartmouth College, Hanover. (1997).
- [8] Hunter, A., Ellis, A. *The Development Process for Courseware Material: A Computing Methodology Approach*. Proc. of 17th Conf. ASCILITE'2000. Coffs Harbour, Australia. (Dec. 2000).
- [9] IEEE LTSC. *Learning Object Metadata*. (May 2001) URL: ltsc.ieee.org/wg12/doc.html
- [10] IMS Global Learning Consortium, Inc. *IMS Content Packaging Specification*. (May 2001), URL: www.imsproject.org/content/packaging/index.html
- [11] Ip, A. and Fritze, P. *Supporting component-based courseware development using the Virtual Apparatus Framework*. Proc. of Ed-Media 98 Conference. Freiburg, Germany. (Jun.1998).
- [12] Ip, A. *A Model for Learner Support Using Computer Networks*. PhD Thesis. MEU. University of Melbourne, Australia. (1999).
- [13] Jones, M. *The Changing Educational Paradigm and Continuing Engineering Education*. Proceedings of the SEFI Annual Conference Copenhagen. Denmark. (Sep. 2001).
- [14] Kulthau, C. *Learning in Digital Libraries: An Information Search Approach*. Library Trends, 45, 3, (1997). 708-724.
- [15] Kunz, W., H. Rittel. *Issues as Elements of Information Systems*. Working Paper No. 131, Institute of Urban and Regional Development, University of California at Berkeley, Berkeley, CA, (1970).
- [16] Landon, B. *OnLine Educational Delivery Applications: a Web Tool for Comparative Analysis*. Standing Committee on Educational Technology. (Jun. 2001). URL: www.ctt.bc.ca/landonline/
- [17] IBM Lotus Corp. *Learning Space*. (Jul. 2001). URL: www.lotus.com/home.nsf/tabs/learnspace
- [18] Luvit Corp. *Luvit*. (Jul. 2001) URL: www.luvit.com
- [19] Lynd, D., Kern, S. *Lotus Notes and Domino 5 Development Unleashed*. Sams. (1999).
- [20] Nielsen, J. *Usability Engineering*. Academic Press, London, 1993.

Ochoa S., Pino J.A., Baloian N., Fuller D. (2003) ICESEE: A tool for developing engineering courseware. *Computer Applications in Engineering Education*, 11, 2, 53-66.

- [21] Novak, J. (1998). *Learning, Creating, and Using Knowledge: Concept Maps As Facilitative Tools in Schools and Corporations*. Lawrence Erlbaum Assoc. (1998).
- [Pre00] Pressman, R. *Software Engineering: A Practitioner's Approach*. 5th Ed. McGraw Hill (2001).
- [22] Rees, J. and Koehler, G. *Brainstorming, Negotiating and Learning in GDSS: An Evolutionary Approach*. Proc.of the 32th Hawaii Int. Conf. on System Sciences (HICSS). Maui, HI, (Jan. 1999).
- [23] J. Roschelle, C. DiGiano, M. Koutlis, A. Repenning, N. Jackiw, D. Suthers. *Developing Educational Software Components*. IEEE Computer, 32, 9, (Sep. 1999) 50-58.
- [24] Theng, Y., Mohd-Nasir., N., Buchanan, G., Fields, R., and Thimbleby, H. Dynamic Digital Libraries for Children, *Proc.ACM/IEEE Joint Conference on Digital Libraries* (Roanoke, VA, June 2001).
- [25] Asymetrix. Learning System Inc (2001). *ToolBook II*. URL: www.asymetrix.com/
- [26] WBT Systems (Jul. 2001). *TopClass*. URL: www.wbt systems.com/
- [27] Uden, L. and Willis, N. *Courseware Engineering Methodology*. Proc. of On-line Learning Euro 2000. Cheltenham & Gloucester College of Higher Education, UK. (Sep. 2000). 6-9.
- [28] Simon Fraser University, Canada (Jul. 2001). *Virtual-U*. URL: virtual-u.cs.sfu.ca/vuweb/
- [30] Virginia Commonwealth University . *WCB*. (Jul.2001). URL: www.madduck.com/wcbinfo/wcb.html
- [31] University of British Columbia, Canada. *WebCT*. (Jul. 2001). URL: homebrew1.cs.ubc.ca/webct/