Teaching Software Engineering through the Net

S. Efremidis, S. Retalis, N. Papaspyrou, E. Skordalakis
National Technical University of Athens, Department of Electrical and Computer Engineering, Division of Computer Science, Polytechnioupoli, 15780 Zografou, Athens, Greece
E-mail: {sef,retal,nickie,skordala}@softlab.ntua.gr

Abstract

The introduction of technology in education seems promising; this is especially so for the new technologies of computer networks and hypermedia systems. However, only through experimentation the effectiveness of these technologies can be demonstrated. There is an ongoing project for this purpose, entitled EONT, partially funded by the European Union within the framework of the Socrates program. Seven Universities participate in this project and each one prepares instructional material for an on-line course to be delivered with the use of new technologies. The National Technical University of Athens is preparing an introductory course in Software Engineering; in this paper we report on the work underway for this particular course.

1 Introduction

Software construction is nowadays in great demand and people need be educated in it [6]. Software Engineering (SE) is a branch of engineering which uses computer science and other branches of science and engineering to offer people a methodology for constructing software in a way that assures the quality and completion of the construction task within predicted estimates of cost and schedule. Many Universities world-wide offer courses in SE; The National Technical University of Athens (NTUA) is one of them.

The Electrical and Computer Engineering Department of NTUA offers an introductory course in SE since 1986, targeted primarily to ninth semester students. The course components are: lectures, discussions, projects, and study material. There is a two-hour lecture session and one hour is allocated to discussion on course material every week. Students gain a lot of experience by working collaboratively in small groups on reasonably large projects. The study material consists of a textbook "An Introduction to Software Engineering" [13] in Greek, and review papers from the recent literature.

However, the traditional mode of teaching faces major problems [8]:

- Lecture attendance decreases. More specifically, the percentage of students attending the course in less than 60%.
- It is difficult for students to ask questions and receive answers outside of the instructor's appointed office hours.
- Students often prefer to ask questions outside the classroom environment.

- The curriculum of the course changes so rapidly that textbooks become quickly obsolete.

Having identified these problems, we started looking for remedies. Recent development of computer and information technology in the fields of computer networks and networked hypermedia systems promise attractive solutions to the above problems. It is now possible to store the instructional material in a central computer and allow it to be accessed by many students through personal computers connected to a local or wide-area network. The problem of interaction can be tackled with the use of synchronous or asynchronous communication [3]. As a result, Open and Distance Learning (ODL), as an instruction delivery method, is a way of overcoming the problems of traditional teaching mode.

Within the EONT project, which is partially funded by the European Union under the Socrates program, we started an experiment in ODL using new information and communication technologies in order to explore the effectiveness of such an approach in teaching [11]. We designed and developed an ODL course "An Introduction to Software Engineering".

The courseware of this course is comprised of the following components:

1. on-line course notes,
2. a study guide,
3. a case study,
4. team projects,
5. printed material (papers, etc.)
6. final exams

The communication between instructors/tutors and students, and between students is facilitated by the Nov-NLE environment [8].

In this paper we aim to present our approach to teaching the course "An Introduction to Software Engineering" in the EONT-ODL environment. The paper is structured as follows. The philosophy adopted when designing the contents of the on-line course notes is illustrated in section 2. Section 3 describes the structure of the ODL course. Section 4 describes the on-line book and section 5 presents the case study. Finally, section 6 contains some concluding remarks.

2 The philosophy of the course

The course obeys the principle that software is needed for automating the execution, through computers, of the soft part of a problem solution. The soft part of a problem solution is that part of the solution which demands information processing. In other words, with the use of software we automate the execution of information processes.

In every artificial system there is a component that automates the execution of relevant information processes.

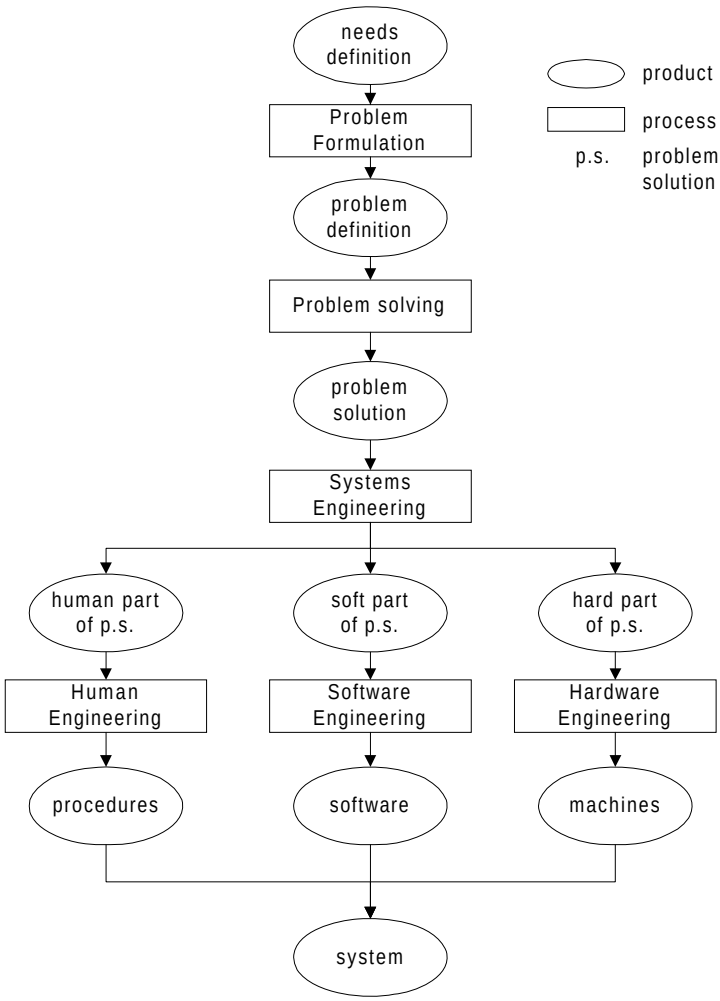

Figure 1: Developing stages of an artificial system

This component is the software system. The stages depicted in Figure 1 are assumed to be followed in developing artificial systems. The main subject of Software Engineering is how to translate a description of the soft part of a problem solution into software. This translation is done in two steps as shown in Figure 2.

3 The ODL course

The course is structured in a way as to serve its philosophy as outlined in the previous section. In this section we describe our approach to developing an ODL course entitled “An Introduction to Software Engineering”. The aim was to provide this course as supplement to lectures and laboratory teaching or as a stand alone, self-study medium.

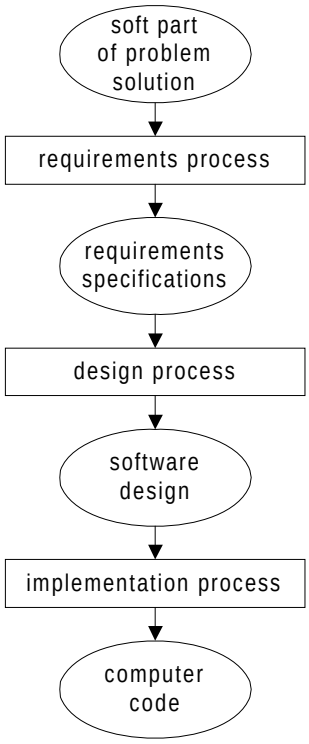

Figure 2: A translation of the description of the soft part of a problem solution into software

This has several advantages over the traditional teaching mode:

- Students can progress *at their own pace* [9] and study the instructional material in an order best suited to their abilities or preferences;
- The information is stored on-line and the course is "open" at any time and from any place for the students registered in the course.

Special emphasis has been put on the structure of the learning environment, called Novel Networked Environment (Nov-NLE). At a physical level, it is a distributed system and uses the client/server model [12]. There is a server, in which all material is centrally stored and many clients (not necessary located in one place), one for each user accessing the instructional material through the Internet.

Nov-NLE consists of the following subsystems:

- The administrative and management subsystem where all data necessary for system's organisation (list of students, of instructors, messages, etc.) are kept.
- The communication-presentation subsystem which is responsible for the presentation of the instructional material to the users as well as for the communication between the various actors (instructors, tutors, students).

The Nov-NLE uses as infrastructure the Internet and the second generation, networked hypermedia system HYPERWAVE [7]. The reason for choosing

Internet is because (1) it offers a number of services, (2) it is a real treasure of information, and (3) it is widely used and available today [2]. The main reason we chose HYPERWAVE is that it possesses characteristics that facilitate the development of Nov-NLE [10, 8].

Apart form focusing on the learning environment, we emphasised on the structure of the instructional material. We decided to adopt the rationale of the UK Open University, which has been used quite widely and successfully in ODL.

The Nov-NLE uses as infrastructure the Internet and the second generation, networked hypermedia system HYPERWAVE [7]. The reason for choosing Internet is because (1) it offers a number of services, (2) it is a real treasure of information, and (3) it is widely used and available today [2]. The main reason we chose HYPERWAVE is that it possesses characteristics that facilitate the development of Nov-NLE [10, 8].

Apart form focusing on the learning environment, we emphasised on the structure of the instructional material. We decided to adopt the rationale of the UK Open University, which has been used quite widely and successfully in ODL.

4 The on-line course notes

The aim of the course is to:

- introduce students to the field of Software Engineering, and
- show how this branch of engineering is practised.

The objectives of the course are to enable students to:

- get an understanding of the Software Development Methodologies,
- get enough practical experience through involvement in a team project, and
- work as junior software engineers in real projects.

The basic component of the courseware is the on-line course notes. The content and learning activities have been organised into self-contained blocks according to the rationale of Open University in the UK [1]. Each block corresponds to specific instructional objectives. It has been divided into units; each unit includes learning activities (questions, exercises, discussion topics) that allow students to check the extent to which they have understood the material and to discuss with fellow students or tutors various course topics.

The content of the on-line course notes is shown in Figure 3. We expect that by breaking the subject matter into small units will not make it “choppy” or disconnected but will increase the chances for students to better understand the material and integrate it into existing cognitive structures.

An Introduction to Software Engineering
OVERVIEW MAP

BLOCK 1. INTRODUCTORY CONCEPTS

- Unit 1 Information Work Processes and Software Engineering
- Unit 2 Software Process Models
- Unit 3 Software Development Methodologies

BLOCK 2. DEVELOPMENT PROCESSES

- Unit 4 Requirement Process
- Unit 5 Design Process
- Unit 6 Implementation Process

BLOCK 3. MANAGEMENT PROCESSES

- Unit 7 Project Initiation Process
- Unit 8 Project Monitoring and Control Process
- Unit 9 Software Quality Assurance Process

BLOCK 4. INTEGRAL PROCESSES

- Unit 10 Verification and Validation Process
- Unit 11 Software Configuration Management Process

BLOCK 5. TOOLS AND STANDARDS

- Unit 12 Tools
- Unit 13 Standards

BLOCK 6. A CASE STUDY

- Unit 14 A front end of the case study
- Unit 15 A back end of the case study: An implementation in MS ACCESS

Figure 3: Content of the on-line course notes.

5 The Case Study

An essential component of the courseware of our ODL course is a case study in which the methodology advocated in the course for carrying out the project, is shown to be used in practice. The case study was considered essential for showing the students how to apply the methodology to the construction of a realistic system.

The case study is concerned with the specification, design, and implementation of a software system for assisting the issuance of the monthly bill for the shared expenses charged to the apartments of an apartment complex, which has independent heating and centrally distributed hot water for domestic use. This software system is meant to be used by a service provider for providing billing services to apartment complexes for their monthly bill issuance.

5.1 Description

We assume that an apartment complex consists of one or more apartments, each one having its own heating devices and hot water supply. The apartment complex has two heaters (a small and a big one) for supplying heat to the heating devices of the apartments. It has also a boiler that supplies hot water to the apartments. The water that enters the boiler is main supply water and is heated by the heaters. Oil is burnt for supplying heat and hot water to the apartments of the complex. Apartments are charged for the amount of heat and hot water they consume. There are counters for counting the hours of operation of various heating devices as well as the volume of hot water consumed by the apartments. Apartments of the complex are also charged with various other expenses like water, electricity, cleaning, lift expenses, maintenance expenses, expenses for repairs, etc. The algorithm for apportioning the cost is based on distribution coefficients that are associated with the apartments.

The case study is structured as two tiers: a front end and a back end. The front end focuses on the problem at hand and its solution. It consists of the needs definition, problem definition, and problem solution. The software engineering part of the solution is put in the general context of systems engineering. The back end focuses on requirements specification, software design, and actual implementation.

Structuring the case study this way exposes the underlying philosophy of the course. The front end is the “static” part, i.e., it remains unaltered since it contains a description of the problem solution, which is not bound to any design or implementation decisions. On the other hand, the back end is the “changeable” part, i.e. it is a function of the design decisions made, implementation, and run time environment. It incorporates decisions that are bound to the particular implementation and runtime environment. Several back ends are apparently possible.

It is interesting to point out that the requirements specification is part of the back end. In the requirements specification the user interface is specified, among other things, which is apparently affected by the implementation environment.

5.2 The front end

As mentioned above, the front end is the “static” part of the case study. It consists of the following:

- *Needs definition*: The need for a software system like the one under consideration is defined. We assume that a software company has identified, through a market research, that in every apartment complex like the one described above there is a need, at the end of each month, for apportioning various costs to the apartments of the complex. The motivation for developing such a system is that the issuance of the monthly bill for the shared expenses charged to the apartments of an apartment

complex is a tiresome and error-prone task if it is to be carried out manually by a human.

- *Problem definition*: The problem is defined clearly and succinctly. The essence of the problem is how to calculate the total expenses (heating, hot water, and various other expenses) each month and how to apportion them to the apartments of the complex.
- *Problem solution*: A solution to the problem at hand is presented. We assume that the solution has been derived by a domain expert and handed to the software engineer. It contains a description of the input data and the necessary steps for carrying out the task of issuing of the monthly bill. Based on this solution the software engineer will design and implement a software system for automating the solution.

Developing a system for the issuance of the monthly bill of the shared expenses in apartment complexes, one has to allocate the various functions to the three components of the system, namely to (a) humans, through human engineering, (b) machinery, through hardware engineering, and (c) computers, through software engineering.

- *Human Engineering-Procedures*: Constitutes the human part of the problem solution and involves all procedures required by a human, like recording constant data of the apartment complex, recording monthly expenses, indications of various counters, etc.
- *Hardware Engineering-Devices*: The hardware part of the problem solution concerns counting the time the heating is on for each apartment, counting the quantity of hot water consumed for each apartment, etc. The hardware used for this purpose consists of heat and hot water counters.
- *Software Engineering-Software system*: The software part of the problem solution involves calculation of the expenses per category, distribution of the expenses per category, issuing an expenses distribution report, calculation of statistics, etc. These operations can be executed automatically by a software system. Such a system can be implemented in various ways. Some are presented in subsequent units.

5.3 The back end

Three back ends have been implemented for three different implementation/run time environments: Microsoft Access, Turbo C, and Microsoft Visual C++. This was done on purpose for making clear the argument that the back end can be implemented in various ways and it is the responsibility of the software engineer to choose the one which is the best under certain considerations. In this case study, the Microsoft Access is argued to be the best choice.

The back end consists of the following parts:

- *Software Requirements Specification*: In this part the requirements are stated for a software system that takes as input various measurements and expenses concerning an apartment complex, and calculates, apportions, and reports the monthly expenses for each apartment of the complex, as well as various statistics. The various interfaces (user, software, hardware, etc.) are specified. In particular, the interaction is specified. Product functions, implementation constraints, and software attributes are also specified. This part follows the IEEE recommendations for structuring [4].
- *Software Design Description*: In this part, the design of a software system for the issuance of shared expenses in an apartment complex is described. This part follows the IEEE recommendations for structuring [5]. It consists of the following subparts:
    1. Decomposition Description: Module decomposition defines the modules of the software system (input, processing, output). Data decomposition defines the structure of the data (tables where data are stored, their fields and attributes, etc.)
    2. Dependency Description: Module and data interdependencies are described.
    3. Detailed Design: Contains internal details of each design module.
- *Implementation*: This part is the actual operational system. It is the code that makes the computer to do what we want it to do.

The mapping from the problem solution to the software requirements specification, to the software design, to the implementation has been made explicit in the case study. The data and the processing steps of the solution are shown to be mapped to more concrete forms as we progress to design and coding.

6 Concluding Remarks

Contemporary educational systems have been criticised as having many drawbacks. In particular, Conventional Universities have been denounced for the constraints they impose on time and place of instruction delivery. Open and Distance Learning has been proposed as a means for overcoming these constraints and contributing to the improvement of the state of higher education. Higher education can profit a lot from the use of new technologies in computer networks and hypermedia systems.

Only through experimentation is possible to give an answer to this expected benefit. EONT project is directed towards this end. Seven European Universities will experiment on this issue by giving, each one, a course the conventional way and on-line with the new information and communication technologies. The on-line mode of delivery will supplement the conventional way. Software Engineering is the subject that NTUA has chosen for its course. A first version of the courseware has been completed and is currently being

evaluated with real students. No evaluation results are available at the moment. The EONT project has a duration of three years and at the moment we are at the beginning of the second year. We expect that the use of modern technology in higher education, and especially for teaching Software Engineering, will be effective.

Acknowledgements

The project EONT is partially funded by the European Commission's SOCRATES Program. Its reference number is TM-OP-1995-1-GR-88(1/0).

References

[1] Coats M., *Learning how to learn*, Kogan Page, Open University, 1995.

[2] Hesslop B., *The Instant Internet Guide: Hands on Global Networking*. McGraw Hill, New York, 1994.

[3] Hilz R., *The Virtual Classroom Learning Without Limits via Computer Networks*, Ablex Publishing Corporation, USA, 1995.

[4] IEEE Recommended Practice for Software Design Descriptions, *IEEE Std. 1016*, 1987.

[5] IEEE Recommended Practice for Software Requirements Specifications, *ANSI/IEEE Std. 830*, 1992.

[6] Gibbs N. E., *The SEI Education Program: the challenge of teaching future Software Engineers*, Communications of the ACM, 5, 1989.

[7] Kappe F., Maurer H., Sherbakov N., Hyper-G: A Universal Hypermedia System, *Educational Multimedia and Hypermedia*, 2(1), 1993.

[8] Koutoumanos A., Papaspyrou N., Retalis S., Maurer H., Skordalakis E., Towards a Novel Networked Learning Environment, *Proceedings of World Conference of Web Society (WebNet'96)*, San Francisco, USA, November 1996.

[9] Marshall A. D., Developing Hypertext Courseware on the World Wide Web, *Proceedings of World Conference on Educational Multimedia and Hypermedia, ED-MEDIA 95*, Graz, Austria, July, 1995.

[10] Maurer H., *Hyperwave: The Next Generation Web Solution*, Addison Wesley, 1996.

[11] Papaspyrou N., Koutoumanos A., Maurer H., Skordalakis E., An Experiment in ODL using New Technologies, *Proceedings of World Conference of Web Society (WebNet'96)*, San Francisco-USA, November 1996.

[12] Sinha A., Client-Server Computing, *Communications of the ACM*, 35(7), 1994.

[13] Skordalakis E., Software engineering teaching at NTUA, *First International Conference in Software Engineering in Higher Education*, G. King, C. Brebbia, M. Ross, G. Staples (Eds.), Computational Mechanics Publications, 1994.