Software engineering teaching at NTUA
E. Skordalakis

National Technical University of Athens, Division of Computer Science, 15773 Zographou, Athens, Greece

Abstract

The value of Software Engineering has been widely recognized. Despite it, the situation today is that most Universities have allocated a single Introductory Course in teaching this subject. Various approaches can be employed in conducting such a course. A brief description of the approach being used at the National Technical University of Athens is presented in this paper.

1. Introduction

Information work processes (IWPs) are stand alone work processes or parts of other work processes. IWPs are characterised by information processing. It was the dream of many human generations to find a way to automate the execution of IWPs which are in general tedious, unpleasant, tiresome, error-prone and time consuming. This dream came true with the advent of computers but under constraints. One of them is that suitable software must be constructed for each IWP to be automated. This task turned out to be not an easy one and a new branch of engineering, Software Engineering (SE), was proposed as the solution to this problem.

SE uses computer science and other branches of science and engineering for fulfilling its goal which is the offering to people a methodology for constructing software in a way that assures quality and completion of the construction task within predicted estimates of cost and scheduling.
Software Engineering in Higher Education

SE has not matured enough as yet. Nevertheless, a big body of knowledge has been accumulated around it. This body of knowledge, can be used for teaching SE since software construction is in great demand and people must be educated in constructing it Gibbs[1]. Though complete curricula have been developed for SE teaching, most Universities World-wide offer only one Introductory Course in SE because under the present conditions they can not offer more, as they might want. The National Technical University of Athens (NTUA) is among these Universities.

NTUA offers a five year degree program in engineering. It has nine Departments. Among them is the Electrical and Computer Engineering Department. In this Department, an introductory Course in SE is given since 1986 at the 9th semester. We have tried in this course variants of the four models of an Introductory Course in SE discussed by Tomayko[2], namely : SE as artefact, topical approach, small group and large project team.

It is the last model that we have found appropriate and for this reason we use a variant of it now. A brief description of this variant is presented in this paper. The following points are emphasized in this approach:
1) there is no easy way to the construction of software (used as a motto again and again during the course)
2) computer based systems automate the execution of IWPVs and software predominantly contributes to this automation
3) not a single software development methodology is suitable in all cases
4) by SE a description of a way of solving a problem is mapped from the problem domain into the computer domain.

The paper is structured as follows. The way that some basic concepts are presented to the students is described in section 2. The structure of the course is described in section 3. A brief comment on the role of the Computer-based Tutoring Systems in SE teaching is given In section 4. Some concluding remarks are made in section 5.

2. Basic concepts

The concepts of what is SE and what is a Software Development Methodology are of paramount importance. For this reason a brief description of the way we present these concepts in the class is given in this section.
2.1 The subject matter of SE

SE is a new branch of engineering that must offer a methodology for transforming a description of a way of solving (a method of solving) a problem by a human being into another description that a computer can execute as depicted by the mapping

\[ D^p_{yq} \rightarrow D^p_w \]

where

- \( D^p_{yq} \) is a description of a way \( y \), of solving the problem \( x \), in the problem domain language \( q \)
- \( D^p_w \) is a description of a way \( w \), of solving the problem \( x \), in the programming language \( p \).

Thus, the implementation of a mapping from a problem domain description of a way of solving a problem into a computer domain description is the subject matter of SE.

People have needs, they formulate corresponding problems and then they find ways of solving them that satisfy these needs. The execution of the ways of solving such problems give rise to IWP's which must be automated in order to free humans from the burden of executing them manually. There is one more reason for such an automation. There are IWP's which cannot be executed by humans due to human limitations in processing speed, memory capacity etc but can be executed by computers. So, automation is at the heart of SE.

When constructing software there must be clear what are the needs, what is the corresponding problem and what is a way of solving it. In old problems, in each one, the needs, the problem definition and a way of solving it are known. All this exists in written form or in the heads of those that are in charge of executing manually the ways of solving these problems (the corresponding IWP's). It is the duty of the software engineer, assisted by a problem domain expert, to collect all this information and have it in the form of a \( D^p_{yq} \). In new problems, a problem solving process must be employed in finding, given certain needs, what is the problem \( x \) definition and a way \( y \) of solving it expressed in the problem domain language \( q \), as depicted in Figure 1. This is not the job of a software engineer but the job of a domain expert, that plays the role of the problem solver.

The transformation of the description \( D^p_{yq} \) into \( D^p_w \) involves also a problem solving process by which a way is found of doing this transformation out of several alternative ways. This process is carried out by a software engineer and may be split into three steps as shown in Figure 2. \( D^p_{yq} \) and \( D^p_w \) are descriptions of the same IWP but have different operations and structure. In the description \( D^p_{yq} \), the operations are meant to be carried out by a worker who is a human being. In the
description $D_{wp}'$, the operations are meant to be carried out by a worker who is a computer.

![Diagram](image)

Figure 1: The steps required in solving a new problem.

The operations and the structure make the difference in transforming the first description into the second description. Software engineers work with descriptions of IWPs that utilise characters from the alphabets of the respective languages.

![Diagram](image)

Figure 2: Three steps may be used in transforming $D_{wp}'$ into $D_{wp}''$.

### 2.2 Software Development Methodology

SE is expected to provide a Software Development Methodology (SDM) that will give answers to all problems encountered in the Software Development Process (SDP). This is an ideal so far. Not a single SDM is good for all application areas (problem domains). For this reason, there are many SDMs in use. A view of SDMs, that is in accordance with this fact, is depicted in Figure 3. The application of a SDM gives rise to a SDP.

Various technical methods, management methods, life cycle models (SDP models) and standards have been devised and pub-
lished in the literature. Many tools have been constructed and are marketed by their vendors. One can choose from these ingredients or invent his own variants and derive a SDM suitable to a given application area and to a developing organisation. The foundation of each SDM is a life cycle model. This model describes what main tasks comprise the SDP and when these tasks take place. The main tasks are independent of each other and are usually referred as the phases of the model. Further refinement of the phases into subtasks and of the subtasks into activities is advantageous. The technical methods describe how the various tasks are implemented. The management methods describe how to control, steer and follow up a software development project as well as who performs which task. The tools support the individuals in their job to perform the various tasks by automating some aspects of these tasks.

![Diagram](image)

Figure 3. Software Development Methodologies and their ingredients.

A SDM includes functional, behavioural and organisational information useful to all involved in software construction, i.e. developers, customers, management, quality assurance, external auditors and configuration management. A SDM needs be well documented in order to be readily accessible to all interested parties.

3. Course components

Our course is an one semester course and it has four components: lectures, discussions, projects and courseware. The components contribute in a balance between the knowledge conveyed to students and the experience they gain in actually working in a real project. The duration of the course is 13 weeks.

3.1 Lectures
There are two hour lectures per week in which is covered, not in depth, the following syllabus:

Week 1 : Basic Concepts
information work processes, automation, software, software systems, software engineering, software development methodologies and their ingredients

Week 2 : Project Management
staffing, scheduling, cost estimation, project monitoring and control, project plan

Week 3 : System and Software Requirements
Customer needs, problem formulation, problem solution, function allocation (to hardware, software and people), system and software requirements document

Week 4 : Design
design concept, design principles, preliminary design, detailed design, structured and object-oriented design

Week 5 : Design Representation
design views, structure charts, pseudocode, design description document

Week 6 : Quality Assurance
quality concept, characteristics of quality, methods of imposing quality, quality assurance plan

Week 7 : Verification and Validation
Verification concept, Validation concept, methods of V+V, reviews, static analysis, dynamic testing, V+V plan

Week 8 : Software Testing
unit, integration, system and acceptance testing

Week 9 : Configuration Management
configuration items, baselines, change control, configuration plan

Week 10 : Human Computer Interaction
textual, non-graphical, graphical interfaces, window systems

Week 11 : Software Development Environments
tools, programming environments, SEE, IPSE, CASE

Week 12 : Software maintenance
corrective, adaptive, and perfective maintenance

Week 13 : Other issues
Professional, ethical, and legal issues

3.2 Discussions
There is one hour per week dedicated to discussion. Subjects for discussion concern issues either from lectures or from the projects.
3.3 Projects
This is the most precious component of the course because students have the opportunity to get a feeling of the issues involved in software construction while working in a reasonably large project. Students are grouped into teams. Each team undertakes a project that it carries out partially using a methodology of its own choice. Most of the tasks prescribed by the methodology are carried out, though not to completion. A good approximation is done for the problem definition, the description of a way of solving this problem in the problem domain language at some high level of abstraction, and the preliminary design. Few modules from the preliminary design are detailed designed, coded and tested.

Work in the project is documented, mostly according to IEEE standards, and deliverables are due in predefined dates. The deliverables are: project plan, requirements document (problem definition, description of a solution), design document, code, test plan and user's manual. There are also two hours per week of supervised work in the lab.

3.4 Courseware
A textbook in Greek entitled: An introduction to SE, and some review papers from the recent literature are used as teaching material. Additionally, an electronic book, with multimedia capabilities, complements the textbook and the lectures. Some of the commonly available software tools are used in the Lab for the project.

4. Computer-based Tutoring Systems

The availability of Multimedia Technology opened the road to more efficient Computer-based Tutoring Systems for self instruction. Such systems have the capability of presenting information in various forms: text, sound, images, video and graphics. Colour can be also used. These systems are interactive and can automate parts of a teacher's job. Additionally, they can perform dynamic simulations, a capability that gives more power to them.

Computer-based Tutoring Systems can be helpful in teaching SE. A first version of such a system has been implemented in our Lab. This system is going to be used next semester as a complementary tool to the lecture component of our Introductory Course in SE. Evaluation results cannot be reported as they are not available yet. Nonetheless, we have the feeling that it will be very helpful especially for its dynamic simulation capability. For instance this system, besides offering a de-
scription of the Cocomo technique for cost estimation it allows the student to practice in this technique by executing it with data of his choice.

5. Concluding Remarks

Though some Universities have programs leading to B.Sc. and/or M.Sc. and/or Ph.D. degrees in SE, the majority of them offer only an Introductory Course in SE. Since a substantial portion of students from the Departments of Computer Science, Computer Engineering, Electrical and Computer Engineering, and Information Systems in these Universities work as software engineers after their graduation, such an Introductory Course is of great value.

The problem for a teacher is what to include in such a course and how to conduct it. A brief description of the content and of the way of implementation of such a course at NTUA is presented in this paper. No formal evaluation of this course has been done. The feedback we have taken from some of our students, now in industry, is positive. Similar papers have been published elsewhere [2]-[5]. Such contributions help in improving both the content and the implementation of such courses to the benefit of students and teachers. Much has been done but more is needed.

References


