DELIVERING EFFECTIVE TRAINING IN SOFTWARE ENGINEERING

Janusz GÓRSKI

Abstract. The paper presents an approach to education in software engineering, implemented as a postgraduate course for industry. A distinguishing feature of the approach is its stress on team projects and a flexible way of controlling the proportion between active and passive participation of students. This provides for high visibility of life cycle activities during the project, despite the actual performance of a particular group.

Keywords. Software engineering; education; team project; industrial training.

1 Introduction

Effective teaching of software engineering depends on the students’ experience with real projects. If the students have little (if any) experience, it is difficult for them to understand the importance of the topics related to non-technical aspects of software development and they often consider those subjects as being obvious and/or expressed in a too wordy and sometimes fuzzy way. On the other hand, to large extend, this is the non-technical aspects like management, organization, leadership, planning, communication, cooperation, which are dominant factors of success in software engineering projects. Therefore a real challenge to software engineering education is to find effective and efficient ways of delivering training which covers the broad scope of topics such as problem identification, project organization and management, quality assurance and technical development.

In this paper we discuss the contents of a postgraduate study program in software engineering. The program is addressed to software industry. Students are assumed to already possess some experience with real software projects as project members or technical managers. The goals of the program are as follows:

- developing understanding of software engineering within the context of a software product life-cycle,
- developing the skills of teamwork,
- providing training in identification and definition of problems,
- providing training in developing a problem solution within the object-oriented paradigm.

*Technical University of Gdańsk, Gdańsk, Poland, e-mail: jango@pg.gda.pl
The core of our approach is the *dragging-through* concept [1] which provides for controlling the proportion between *development-oriented* and *case study-oriented* effort during realization of a student project. Those two kinds of activities are very different in respect to the predictability of the required effort. It is much easier to estimate and control the effort necessary to study an already existing solution of a problem than to estimate and control the effort needed to solve this problem. The latter is in fact the main source of difficulties with introducing real-life scale projects to the training curricula: there is no guarantee that the project will be completed within the specified time frame. The other extreme, i.e. teaching real-life scale projects through pure case studies of ready solutions is also unacceptable as the active component of the learning process is in this case almost completely reduced. The *dragging-through* approach is a solution to this dilemma: it assumes that the student project is the mixture of *active development* and *passive case study*. The proportion between the two is individually controlled for each team of students working on the project.

2 Problems with introducing projects to SE training

Fig.1 presents main activities related to a software project. They include problem identification and definition, designing a solution to the problem and its subsequent implementation. Those activities are supported by project management and quality assurance. At the bottom we show a time frame within which those activities are realized. A common problem is that this time frame is very difficult to predict in advance as it depends on many factors like problem complexity, the suite of methods used to solve the problem, the underlying technology, professional maturity of the team solving the problem, stability of requirements and many others. In particular, one can expect that this time frame will be different for different teams independently working on the same project, even if they use the same methods and technology, as the quality of people involved in those teams will not be the same.

![Diagram of software project activities](image)

Fig.1. Fitting a project into its time frame.

The above observation exposes the main problem with software engineering projects during education. For such projects the time frame is fixed and the same for all teams. This
time frame results from the overall schedule for the training and is not flexible (no company would accept the training which takes between 3 weeks and 6 weeks, depending on „how well we will do with the project”).

A solution to the above problem could be to force the project team to meet the deadline. However, if this is done just in the form of a directive („you have to meet the deadline”) the team will concentrate on the final product rather than on the process and will start to implement too early, skipping over the definition and design related activities and reducing management and quality assurance. In this way the training would totally miss its target: instead of teaching good software engineering practices it rather reinforces the bad habits and practices very common in the industry.

Another solution could be to give the priority to the process and to force the team to work according to a well defined methodology with proper focus on the early phases of the project, explicit design, sound management, quality assurance and good documentation. In this case however, unless the problem to be solved is small, it is very likely that, when the deadline comes, the team ends up somewhere in the middle of the project, with unfinished design and implementation. Reducing the size of the project does not help here because a very small project would not expose the students to the real problems of software engineering as many of them become dominant and visible only for larger projects.

Both solutions are not satisfactory as both fail to expose the students to all project activities maintaining proper balance between them and controlling the use of good methods and practices.

3 The dragging-through approach

The dragging-through approach attempts to provide uniform distribution of the effort through the life-cycle and the equal visibility of all life-cycle phases to the student groups involved in the training. The underlying assumption is that the project work is a combination of active development and passive case study. The proportion between the two is specific to each team. To meet the deadline a team with poorer performance will do less of active development and more of case study whereas for a more productive team this proportion can be reversed. Fig.2 illustrates this idea by showing the totality of the work to be done in the project and its distribution for two teams with different performance.

To provide for a more fine control over this active-to-passive ratio the project is split into phases, each phase involving a well defined task and the resulting product. The end of each phase is a checkpoint during which each team is presented with the golden solution for this phase. Then, by studying the golden solution a team compares it with its own work and focuses on understanding the differences and completing the omissions. This way the team is dragged-through the rest of this phase and is ready to start the next one. Even if a given team was far from completing the previous phase at the beginning of the next one, all teams start from the same level.

Golden solution is an initial investment which has to be done before the training is offered to the market. However, our experiences so far show that the golden solution does not „wears up” too fast and can be re-used several times.
4 The project course

Phase 1 During this phase the team defines its structure, decides about the roles, estimates the effort and the resources. This results in the Preliminary Project Plan. The planning activity is then continued throughout the whole project with continuous monitoring and updating the plan as the project progresses.

Phase 2 The next activity is the development of requirements, which is the process of continuous requirements elicitation and analysis. The only available source of application-domain knowledge is the customer. He is not a software expert - it is the students' task to drag out of him all the necessary information. The main product is the System Requirement Specification (SRS) document. The Acceptance Test Plan which is a part of SRS, sets up the criteria for the system acceptance by the customer.

Phase 3 The knowledge about the application domain is represented by the Application Model, conforming to the OMT methodology. The process of modeling is supported by a CASE environment. The documents created at this stage of development provide information on what the system should be. The documents created at this stage of development are presented to and agreed with the customer.

Phase 4 Then, the system model created at the previous stage is analyzed and the strategic decisions about the system architecture are undertaken. In particular, they define the system structure in terms of subsystems, the target implementation environment for each subsystem and the communication means for subsystem interfacing. These results are described in the System Design document.

Phase 5 Obeying the strategic decisions from the System Design, the models are refined until the level enabling implementation is reached. The Object Design documentation have a mixed form: graphical (refined model), textual and pseudo-code. The Test Plan document covers the issues of module, integration and system level testing.

Phase 6 The Implementation of the system is performed in the Object-Oriented Database (O2) environment. The O2 workbench is used, with methods implemented in O2C
language (close to C++). The final testing of the system is performed with the presence of the 
client, and in accordance to the Acceptance Test Plan from the SRS document.

Phase 7 The final presentation of the product gives the opportunity to show the 
results, discuss the problems encountered and make the final evaluation. The Project History 
Report is to be submitted, which contains the detailed report on all tasks performed, with the 
real effort put into them, for each individual team member. The Individual Final Report, 
written by each student includes personal remarks on the individual contribution to the project, 
list problems encountered, etc. To be able to write it at the end of the course, it is 
recommended that each student keeps a personal diary throughout the project.

5 The curriculum

The structure of the curriculum is given in Fig.3. The figure shows the Team Project 
which is a sort of the backbone of the whole program. A number of teaching modules feed the 
Project with the theory necessary to proceed.

![Fig.3. Structure of the curriculum](image)

The following set of modules is presently offered together with the Project (in the 
brackets we give the number of teaching hours of a given module):

**Project Management (14)**

*Contents:* Understanding projects; Contractual matters; Creating the project 
infrastructure; Project management in the larger business context; Strategic planning; 
Feasibility studies; Requirements capture and specification; Planning and control; 
Verification and validation; Quality assurance and the importance of quality in project 
management; The end-of-project review; Managing incremental delivery projects.

**Requirements Engineering (14)**

*Contents:* The context of requirements engineering; Requirements capture and 
analysis; Viewpoints; Interviewing; Integration; Analysis of requirements; 
Presentation; Validation; Requirements specification: Guidelines and standards; 
Attributes of a specification; Assumptions; The problem of gaps; Change control.

**Metrics and Quality Assurance (14)**
6 Conclusion

In the paper we have presented the ideas related to the new training program presently offered to the software industry in Poland. The first edition of this training is scheduled for October 1998. Similar ideas have been previously successfully used to train international groups of students in the Franco-Polish School of New Information and Communication Technologies, in years 1993-1996. More details of the dragging through approach and the
experiences in its practical application can be found in [1,2]. A full description of the present program (in Polish) is given in http://www.eti.pg.gda.pl/cku/sp/io/.

7 References