Experience in Early and Late Software Engineering Project Courses

Birgit Demuth, Mike Fischer and Heinrich Hussmann
Department of Computer Science
Dresden University of Technology
D-01062 Dresden, Germany
{Birgit.Demuth, Mike.Fischer, Heinrich.Hussmann}@inf.tu-dresden.de

Abstract

We report about our experience which we gained in different software engineering project courses at our department. A student who is specialized in software engineering has to finish besides lectures on software engineering and related fields two project courses: an "early" basic project course and a "late" complex project course. For both of them we developed leveled teaching approaches considering the different knowledge and skills of beginners respectively "advanced" students. The paper describes, discusses and evaluates these approaches.

1 Introduction

Software engineering education at the Department of Computer Science of the Dresden University of Technology is part of several study programs:

- Graduate program in Computer Science (bachelor and diploma degree)
- Graduate program in Media and Computer Science (bachelor and diploma degree)
- Graduate program in Technology of Information Systems (bachelor and diploma degree)
- Post-diploma program in Software Technology (second diploma degree)

Courses in software engineering are embedded into the "early" basic study program as well as into the "late" main study program. It is not surprising that software engineering lectures should be accompanied by practical software development and be leveled to the already acquired knowledge and skills of the students. Our main attention in the training of future software engineers is focused on object-oriented (OO) techniques and doing teamwork. We, accordingly, developed a teaching approach for an early and a late software engineering project course and refined them during the last years based on our experience.

The early project course (*basic project course*) follows the idea that beginners have to climb several rungs of a very demanding ladder of technical knowledge. Because of teaching the "OO thinking" we lead the students to a "good" OO application design by using the domain framework *SalesPoint* [1].

Based on first object-oriented software development skills of the students, the late project course (*complex project course*) focuses attention on higher requirements in software analysis and design as well as in project management by an increased number of team members. Whereas, for example, the application architecture in the basic project course is given by the used framework, the students of the complex project course have to discuss different architecture designs and have to choose appropriate implementation technologies.

In the following, we outline the structure of the software engineering education at our department and present more details about the project courses. We discuss our experience and summarize our “Lessons Learned”.

2 Software engineering education at Dresden University of Technology

2.1 Overview

Graduate programs at the Department of Computer Science are structured into a basic program (four semesters) and a main program (six semesters for bachelor degree, nine semesters for diploma degree).

Table 1. Software engineering and related courses

Type of course	basic program				main program				
	1	2	3	4	5	6	7	8	9
Basic programming	Algorithms and data-structures	Programming							
software engineering			Software engineering		Software engineering II	Software management			
					Software development tools	User interfaces and dialog techniques			
						Software components	Diploma semesters		
						Formal specification of software systems			
practical courses		Programming		Basic project course		Complex project course			

Table 1 shows the curricula of the graduate programs Computer Science as well as Media and Computer Science in the point of view of software engineering education. Programming courses during the first and second semester introduce the students to the “Programming-in-the-Small” and prepare them for the subsequent software engineering courses. The students are taught (among other things) structured programming and verification techniques, sorting algorithms and data structures including hash tables and AVL trees. Programming tasks in the practical course are defined precisely, and typically could be solved by one person in a few days.

Software engineering represents the introductory course to the “Programming-in-the-Large”. The course emphasizes techniques of object-oriented software development including object-oriented analysis (OOA) and design (OOD) with the Unified Modeling Language (UML), Java-based implementation and design patterns.

All courses in the basic program are mandatory for all graduate programs (including the post-diploma program). The following software engineering courses in the main program are optional. They deal with different fields and questions in software engineering like reverse engineering, development tools, user interfaces, software components and formal specification of software systems.

2.2 Basic project course

For most of the junior students, the basic project course is the first hands-on experience in software engineering. The basic project course as in the following described was performed several times both at our department and at the University of the Federal Armed Forces Munich [9]. It requires extensive practical work and provides valuable experience for programming assignments and advanced courses in the main program. According to the learned concepts of the introductory course the used technology in all project has to be OOA/OOD with UML and Java-based implementation. The central technical idea of the basic project course is the use of a domain framework which gives the students a guidance in object-oriented design by its architecture and implemented classes [1].

Besides the training of the technical skills the students have to develop a number of new ("soft") skills [2] in order to succeed with their projects:

- They are required to work in teams of five or six members. They assume different roles (chief programmer, developer, etc), have to communicate in a professional way within the team and with customers, and work systematically during three months following a plan made by themselves.
- They are required to specify what their task is. In contrast to the programming tasks they were given in the programming course the project definition is rather sketchy. It is the students' job to find out in discussion with their "customers" what to do in detail. However, in point of the view of the teaching staff the tasks are rather well-defined. All tasks represent various point-of-sale applications [3]. The same task is given to three project teams every year so that there is much experience about various solutions.
- They are required to present their work. Both oral and WWW presentations are required.
- They are required to acquire tool experience. In addition to some Java IDE they need an UML-CASE tool, a version management system like CVS and some HTML knowledge to set up their own WWW project site.

The project teams are guided by senior students who in turn are supervised by a project course leader. This organization is necessary because we have to teach hundreds of students in the basic project course every year. The senior students work as tutors. They are both consultants for the younger students and customers for the software project. This way the senior students acquire own advanced soft skills in project management.

A rather rigid time table is prescribed for project work. Once per week, the project progress and the results (documents, programs) have to be reviewed together with the tutors. Furthermore, the software process has to be documented at the WWW project site. After the strict project deadline a formal oral presentation per team is required where the main results including the developed application have to be shown and questions to be answered.

2.3 Complex project course

The complex project course is attended by students of the main program, typically in the 6th or 8th semester. At this point the students have completed most of the theoretical lectures, like both Software engineering courses. Additionally, they have acquired practical experience in software development by several practical courses, including the basic project course. Most of the students have also acquired practical skills in short industrial projects. Overall our observation is that the students have high skills in carrying small projects supervised by professionals.

Considering these skills, the complex project course provides some students the opportunity to gain experience in more realistic projects, e.g. with higher complexity and with more realistic tasks, but with technical support of the teaching and research staff.

There is typically only one group of the complex project course with approx. 9 students (in future more groups have to be established). For each course, a new task is specified. The different tasks span a large field from a web-based game up to a task with requirements derived from real customers. For example, the recent project *ProjectWeb* [4] was to develop a web-based working environment for worldwide distributed researcher groups. The requirements came from a European project in which the Dresden University of Technology is involved. The real background of the task allows students to take part in real contractor-customer-relationships: the students are the contractors and the project members are the customers.

Students assume full responsibility for the project work. However, if obvious risks are observed, the project mentor gives some hints to the students. The students have also the possibility to request support in each of the project phases. Additionally, each phase ends with a presentation of the results. With the initial task description we also provide the students also a milestone plan proposal for timing. However, it is the responsibility of the project group to refine the plan and to generate its own milestone plan. Following, the main steps of the complex project course are be described:

- Initially the students establish a project team with roles (e.g. team leader, secretary, technician, ...) and rules (style guide and management of project documents – more general project conventions, regular dates for project meetings, ...) which are appropriate for a higher number of project members.
- The task of the complex project course is very sketchy at the beginning. In contrast to the typical task of the basic project course it is also sketchy for the mentor because there is no experience with the given real problem. Therefore the students have to collect the requirements in the first step. Typical questions are: “What does the customer really want?”, “What is the meaning of terms which the customer uses?” and so on. This first step creates a product description which is signed by the customer. This document is used for evaluation of the results at the end of the course.

- The second step of the project is the classical object-oriented analysis. Our students have very high theoretical skills in this field. However, the new challenge is to analyze a complex system and to coordinate the analysis work of several project members.
- The third step is focused on the architecture design. This includes the choice of appropriate technologies (e.g. Java Server Pages, Enterprise Java Beans) and software (e.g. Tomcat, JBOSS) used for software development in the project course.
- And finally the software has to be implemented and tested.

The complex project course ends with a public oral presentation of the results. The students also hand over documents like project code, project documentation and an application manual.

2.4 Comparison of the early and the late project course

After a short description of the basic project course and the complex project course we present the differences between the two approaches.

At first table 2 summarizes the early respectively late properties of the basic respectively complex project course.

Table 2. Early and late properties

	basic project course	**complex project course**
Semester	4th semester	6th semester
theoretical skills	basic	advanced
Practical skills	only little	medium

According to the students' project experience and the total number of students in one project course we established the following different organizations (see table 3).

Table 3. Organization of the project courses

	basic project course	**complex project course**
number of project teams	25 and more	1 (to be enlarged in future)
Members per project team	4-6	8-10
project time	13 weeks	almost half a year
mentored by	senior students	teaching and research staff
Consultation	once per week	on demand of the students
project deadline	Hard	soft

Table 4 summarizes the differences of project tasks in both project courses. The rather low motivation of the students in the basic project course come from the fact that their task – a point-of-sale application – is only a simulation and is not used by any user after the project course. In contrast to these simulation tasks which will be only solved "for the desk of teaching staff", tasks with real customers and needs create a better work atmosphere with a high motivation of the team members.

Table 4. Type of the project task

	basic project course	complex project course
Requirements	well-defined	sketchy
Complexity	Rather low	rather high
effort in hours per student and week	about 8	about 16
Reality	simulation	high
used technologies	predefined	open
Software architecture	predefined	open
motivation	rather low	rather high

The priorities of teaching objectives in the project courses are given in table 5. This emphasizes our point of view what are the knowledge and skill we want to introduce to students early and late in their programs. Notice that analysis and programming have high priority in the early study program. Design issues are partially skipped by mandatory reuse of classes of the given domain framework. In contrast, the complex project course especially requires knowledge and skills in requirements engineering, architecture and OO design as well as in project management.

Table 5. Priorities of teaching objectives of the project courses

	basic project course	complex project course
requirements engineering	no	high
OO analysis	high	secondary
OO design	secondary	high
architecture design	no	high
reuse	high	no
OO programming	high	secondary
Project management	secondary	high
learning of previous unknown technologies	no	secondary

3 Experience and lessons learned

The described teaching approaches for software engineering project courses were developed and improved over a period of several years. After C++-based basic project courses without our current strict organization (no framework, soft deadlines) we started the *SalesPoint* project courses in 1997 and performed them once or twice per year [5]. Besides a lot of informal feedback from students and tutors we also requested detailed questionnaires to evaluate the project course approach. For example, the results of questionnaires in summer 2001 are given in [7]. With the complex project course we have four years experience [6]. In retrospect, we have learned very valuable lessons in both courses.

There are never too many project courses!

The experience shows that there are too few project courses in our program! Even highly qualified students who worked, for example, as tutors in basic project courses had trouble in OOA/OOD as well as in project management during the very demanding complex project course. Therefore, it makes sense to require that the complex project course is mandatory. However, project courses need much human and infrastructure resources which can not be provided as often as we would like during a program with such a large number of students.

The framework-based approach in project courses is considered suitable for beginners in software development!

Beginners tend to do everything from scratch and “reinvent the wheel” many times. Only by using the framework, will they be guided to a real object-oriented design and produce solutions that can be maintained easily [1].

Real tasks motivate students to complete their work!

For example, in no case a single student or the whole group cancelled the complex project course. In contrast, we had some motivation problems in the basic project course because of the simulation character of the tasks mentioned above. Therefore, we experimented last year with real industry-driven tasks. After the project course, we asked the students how the project task was interesting (on a scale of very interesting (1) to not interesting (5)). Students in our “internal” project course valued this subject lower (3.2) than students working in “real” projects (1.7). However, it is almost impossible to transfer our described early project course approach to an industrial environment and above all for hundreds of students.

The students do not use documents created during early software development phases as tools for their project work!

The students do not understand requirements, analysis and design documents as tools for their project work. Instead it seems that they create such documents in order to please the mentors and customers. Therefore, a better training is required to show how different models interact together in different phases. A further aspect of documentation is that support and guidance by tutors respectively mentors is impossible unless progress is continuously and extensively documented.

Mostly, students underestimate the project effort!

The students have no experience in estimating the project effort, especially the time involved. The resulting specification during the requirements phase is often extremely oversized. So an important question is: What is a realistic scope of the task to be solved in the available time? This observation we made both in basic and in complex project course.

Object-oriented analysis is hard work!

Junior students do not often see the necessity of the analysis of the given task. After much trouble in their first software project they recognize that they were wrong. Senior students in the complex project course have problems to deal with complex problems. It is important to exercise how to decompose the whole task into small manageable tasks. A further problem is to do analysis independent of technology.

Architecture and object-oriented design needs much experience!

Architecture design which is required in the complex project course is very hard for inexperienced students. It takes a lot of human resources for different problems: evaluate existing technologies, guide other project members with regard to the chosen technology, evaluate software, design a system architecture and go in iterative steps to a detailed design.

Therefore, we often just skip these problems in the basic project course by the use of an application framework.

Coding and test

The last steps of the project courses are coding and test. Mostly, these phases are trouble-free above all in the complex project course since the students have already high skills in programming. We are also surprised every year about the high productivity which is achieved by both the junior and senior students in their projects measured in lines of code per student and hour.

Teamwork is the most important experience for junior students!
Students' feedback in all basic project courses confirmed that the teamwork was a new and exciting experience. The students learned a lot and do not want to miss this experience in spite of every effort [7].

A strict deadline is a powerful tool in student project courses!
To avoid disaster especially in basic project courses with hundreds of students, some strict discipline and a lot of communication is required. Teams have to adhere to a strict project deadline which is synchronized with the end of the semester. The high success rate of about 90% reinforces this observation.

In future, we want to refine both basic and complex project course. An interesting guide in this regard will be the SWEBOK project [8]. Furthermore, we will turn our attention to an effective guidance of the students according our lessons learned. A difficult task will be the creation of industry-driven project environments with very similar requirements to the constraints of our existing basic project course.

Acknowledgments

We thank Lothar Schmitz from the University of the Federal Armed Forces Munich for his valuable contributions to the basic project course approach and his steadily effective cooperation. Furthermore we extend our gratitude to students who, through their efforts and contributions as tutors, made the organization and the cycle of the project course successful – with special recognition going to Steffen Zschaler and Kathleen Krebs.

References

[1] B. Demuth, H. Hussmann, L. Schmitz, St. Zschaler. A Framework-Based Approach to Teaching OOT: Aims, Implementation, and Experience. Proceedings Thirteenth Conference on Software Engineering Education & Training 6-8 March 2000, Austin, Texas, eds. Susan A. Mengel and Peter J. Knoke, IEEE Computer Society

[2] B. Demuth, H. Hussmann, L. Schmitz, St. Zschaler: Teaching OOT Using a Framework and Both Direct and Net-based Tutoring, ED-MEDIA 2001 World Conference on Educational Multimedia, Hypermedia & Telecommunications, Tampere, Finland, June 25-30, 2001, eds. Craig Montgomerie and Jarmo Viteli, AACE

[3] Dresden University of Technology. Softwarepraktikum Sommersemester 2001. http://www-st.inf.tu-dresden.de/sp

[4] Dresden University of Technology. Komplexpraktikum Sommersemester 2001. http://www-st.inf.tu-dresden.de/kp

[5] Dresden University of Technology. Software Engineering Project Course. http://www-st.inf.tu-dresden.de/BasicProjectCourse/

[6] Dresden University of Technology. Complex Project Course. http://www-st.inf.tu-dresden.de/ComplexProjectCourse/

[7] Dresden University of Technology. Evaluation of Questionnaires, summer 2001. http://www-st.inf.tu-dresden.de/SalesPoint/v3.0/papers/evaluation.html

[8] IEEE Computer Society. Guide to the Software Engineering Body of Knowledge (SWEBOK). http://www.swebok.org/

[9] University of Federal Armed Forces Munich. The SalesPoint Framework V2.0 Homepage. http://inf2-www.informatik.unibw-muenchen.de/Lectures/SalesPoint/