Teaching Software Engineering by Developing Commodity Software

Janina Mincer-Daszkiewicz
Faculty of Mathematics, Computer Science, and Mechanics,
Warsaw University, Poland

Abstract

Software engineering is one of the main disciplines of computer science. It plays a crucial role in computer science education, especially on a graduate level. However as important as software engineering education is, it is the subject of constant discussions, changes, and controversies.

At the Faculty of Mathematics, Computer Science and Mechanics of Warsaw University we launched a software project with the aim to produce an integrated student management information system (called USOS) for the faculty, the university and other Polish public universities. It is a huge database-oriented software application done partly in Oracle technology and partly in open source technology (Internet modules). The system is designed and developed by computer science students supervised by lecturers and Ph.D. students. For the last three years software engineering education at the faculty was focused around USOS development. This process is manifold — it takes part on a variety of courses, many methods and tools of software engineering are tested along the way, possibilities for research experiments arise. Students learn by applying standards and being exposed to good practices. USOS provides a unique opportunity to integrate education in software engineering with production of commodity software. That model of software engineering education is very successfull and will be continued.

Keywords: Higher education; teaching software engineering; software engineering in the large; USOS — University Study-Oriented System.

1 Introduction

Software engineering is one of the main disciplines of computer science. Every profesional project leader or member of a software team should know its principles, methods and techniques. Lack of that knowledge is the cause of delayed, over budget, or failed projects.

No wonder that software engineering plays a crucial role in computer science education, especially on a graduate level. It is being taught on computer science faculties all over the world. However as important as software engineering education is, it is the subject of constant discussions, changes, and controversies.

The problem seems to lay at the very roots of software engineering. It demands virtues and abilities which should be gained by experience and as such are hard to be obtained in "artificial environments" of university classes and labs.

One of the possible ways of practicing software engineering is to make students take part in an industry project supervised or co-supervised by people from academia. However such projects, if ever launched, rarely succeed. Industry people seldom treat students as valuable team members. They always put business goals ahead of didactic goals. In countries with emerging economics, like Poland, software companies compete for clients and are not interested in going into risky business of cooperation with an university partner.

The other threat is that students involved in industry projects get detached from the university environment and academic duties. It is hard to put aside business rules in order to achieve academic purposes. Students who get too much involved in off-university duties lose the possibility to finish Master Thesis on time. In Poland students of computer science often start professional life on their 3rd or 4th year, however most of them do not get diploma on time.

Is there any solution to this dillema? At the Faculty of Mathematics, Computer Science and Mechanics of Warsaw University we launched a software project with the aim to produce a student management information system (SMIS) for the faculty, the university and other Polish public universities. It is a huge database-oriented software application done partly in Oracle technology and partly in open source technology (Internet modules). The project is supervised by faculty teachers and Ph.D. students. The computer science students of the faculty are programmers, sub-project leaders, testers, documentation writers, users etc. They all learn software engineering methods and techniques by practicing software engineering in the large, taking part in the real life software project, producing commodity software at academia and for academia. The developed system, USOS (University Study-Oriented System), is currently being used at pilot departments of a few Polish public universities and is being deployed at the others. USOS is on the one hand the vehicle of software engineering education, and on the other — the way towards Virtual University.

This paper is about a new model of software engineering education being adopted at the largest Polish university, with the best faculty of computer science in Poland, which students for many years took part in finals of ACM collegiate contest in programming and this year won a golden medal. We think that we found a successful method of improving the training of software professionals.

2 Problems of software engineering education

The main goal of software engineering courses offered at universities as part of Computer Science curriculum is to make students practice "programming in the large" (cf. [13]). The emerging problems are that programming in the large means taking part in real life projects, aiming at the production of commodity software. Students rarely have an opportunity to participate in all stages of the software process: specification, verification, design, implementation, testing, and maintenance. They are not motivated to meet the requirements, work according to time limits, bother for quality of the final product, take part in its deployment and maintenance.

On university courses students get an opportunity to practice methods and tools, but are not exposed to REAL problems. What is even worse they may get an impression that there aren't any — all problems seem to be well defined, intuitively modeled by known techniques and methods, soluble in a semester (cf. [8]). Students often consider subjects usually taught on software engineering courses as being obvious and/or expressed in fuzzy ways. However real systems rarely have neat solutions and the requirements for those systems are influenced by many technical and non-technical factors. Aspects like management, organization, leadership, planning, communication, cooperation to large extent are dominant factors of success in software engineering projects.

To solve the problem computer science curriculum should drift towards "project based" as opposed to "content based". Statistics show that around 90% of the course contents is forgotten if not applied immediately. Experience sticks much longer.

Students should get acquainted with methods, techniques, and tools, but they also should learn how to function on a team, manage individual and team tasks and projects, communicate effectively orally and in writing, formulate or defend a proposal, write memo, prepare an agenda for meeting, etc.

The existing discrepancy between university education and industry needs is at least partly responsible for the criticism made of many university courses by employers that the courses do not equip students for real work.

3 USOS as a testbed for practicing software engineering

It all started in November 1999, when the New Educational Tools Tempus JEP project JEP-14461-1999 was launched (cf. [6, 10, 11, 12]). The project was partly sponsored by the European Union and it gathered 17 Polish public universities[1] around the common goal of unifying university administration procedures and producing common student management information system. The greatest challenge of the project was to conduct requirement engineering at the participating universities and come up with the software package which would be accepted by all partners, used at the university level, and possibly approved by the Ministry of Higher Education as standard.

One of the very first conclusions was that none of the software packages currently in use could be easily modified to fulfill the needs of all participants. It also quickly became clear that no Polish software company had been able to provide the application software needed for the national level university-wide database with functionality requested for SMIS, mostly due to the limited budget of the project, but also because of the lack of expertise on study programs and procedures involved in university management.

[1]In Poland there are 17 state higher education institutions called *universities*. All of them participated in the project. Detailed information on the Polish system of higher education is available in [4].

The help came from the authorities of the Faculty of Mathematics, Computer Science and Mechanics at Warsaw University, who decided that the needed software would be designed and developed in-house, by the teachers and students of the faculty. By engaging students as programmers the problem of the low budget could be solved. And of course no software company could compete on the ground of expertise in academic procedures with the team of teachers, students, and authorities settled in university environment.

However the production of the system was not the only aim to be achieved, and even not the most important one. The main and strategic goal of academia is to teach and do research. By launching the project of that size and time scale the testbed has been created for working out new models of teaching computer science and driving research in the area of software engineering. It also seemed to be an on-going project for many years to come (mostly due to changes in requirements so typical for education institutions, low level of automation of administration procedures, and expected changes in work style driven by deployment of a large software application). The software could be designed, developed, deployed, and maintained at the university providing the unique opportunity of involving students of many generations in all stages and activities of the project.

The vice-dean of the faculty was appointed a project leader. Being a high rank administration officer on the one hand and software engineer and academic teacher on the other she could state and prioritize requirements, make decisions, enforce their implementation and keep proper balance between business and didactic goals.

This is not the first time students at university take part in producing commodity software. What makes USOS project different from the others is the scale and multilevel integration of design, development and deployment activities with education.

In the sections to follow we describe various aspects of the project, focusing on those which influence software engineering education.

3.1 USOS development as an integral part of computer science curriculum

USOS development has become an integral part of computer science education in our faculty. It involves many activities carried out on various courses and a plethora of educational models. We could say that students take part not just in particular courses but in a whole process. In this section we are going to describe it.

Computer science students at Warsaw University at the second year of their studies take an obligatory course on *Software engineering*. On lectures they learn the theory of software engineering (usually according to some classic textbook, e.g. [13]). On accompanying labs they follow all steps of the medium scale software project (this year they design the solution for the train ticketing system), using Rational Rose set of tools to draw diagrams, and a simplified version of RUP (Rational Unified Process) as the leading methodology.

The first step towards more team-oriented experience they make at the 3rd grade, taking the course on *Team programming*. For example two years ago one USOS module has been designed and developed. In our faculty students first register for courses and then choose between various groups scheduled for each course (groups may be lead by different assistants and take place at different days and times). The module for group registration allows students to deliver priority lists of groups they want to attend. Then it runs a special algorithm which takes into account some ordering of students registered for the courses (e.g. according to grade averages) and student priority lists and distributes students among groups trying to avoid collisions (e.g. appointing students to groups of various courses which take place at the same time). This module was designed and developed by teams of the 3rd graders. The module requirements were first vividly discussed by students and academic teachers. The whole faculty voted for the best module which then had been incorporated into USOS nad now is used at the faculty. The winning team was obliged to deploy the module. It turned out that so many practical problems arosed during deployment and so many extra programming work was necessary that in the end the module became a Master project for the members of the team.

Evening students are exposed to USOS on the course on *Design of very large data bases*. They usually have some elementary knowledge of databases and SQL programming and little experience with Oracle design and developer tools. They are given various small tasks. Let us look at one example. The USOS database was filled with the data transferred from the old databases. However in the old ones the address had been stored as one string, whereas in USOS we wanted to keep various parts of the address separately. Students were given a task to write an SQL script for changing the structure of the table and rewriting data from the old to the new. Students had to apply some heuristics on how to correctly recognize various parts of the address in the string. A few students did the same task — their scripts were tested on real data and the percentage of correct conversions was determined. Then the best script was run on the data exported from the old databases to the USOS repository. The winning students not only got good grades, but also an extra satisfaction.

Other students of the same course were given other tasks. For example we decided to change the model of data concerning description of program studies leading to various special diplomas etc. Shortly speaking in a few places one-to-one relations among entities were changed to many-to-many. This change touched many tables, forms and reports. Students were supposed to do thorough impact analysis of this change on the database tables and user interface. They had to prepare scripts for changing database structure and new versions of forms and reports. All modifications were thorougly tested and then they were integrated with USOS.

Another course which delivers USOS-oriented tasks is *Web programming*. USOS main database can not be accessed through the web, however there is also a Web module, called USOSweb ([7]). It is a separate application in a sense that it is attached to a separate database kept on a separate server. It is made in open source technology, like PHP and MySQL, and many installations of USOSweb may run simultaneously, e.g. one per each department of the university. Web databases are partial copies of the main Oracle database. In particular highly confidential personal data are not stored on the web, data are available mostly for reading, and modifiable data comprise only a small subset. Such architecture allows to avoid high cost of licensing which would arose from permitting massive access to the central database, It also increases scalability of the solution (e.g. we may install one web database per faculty) and the security (confidential data are only available to limited number of users and are not endangered by web access). On the other hand, however, this architecture is more complex from the technological point of view since databases have to be synchronized by special algorithms and mechanisms.

Students taking part in the course on *Web programming* were working on search engines for USOSweb or database synchronizers for synchronizing main Oracle database with Internet databases. Since we steadily enlarge the amount of data kept in the database and want to make it available to students and academic professors, new tasks emerge of making these data available through web browsers. The examples of such data are detailed information on degree requirements, individual requirements of student profiles, student progress, personalized class schedules. We drift towards the solution were USOSweb plays a role of a Virtual Students Office, offering integrated set of services.

A couple of Master projects concerning USOSweb were carried out. The main examples are: a module delivering main functionality of USOSweb (course catalog, electronic examination sheets, electronic transcripts of records), a module for course registration, a module for course assessment, a communication module which allows students taking particular courses and course lecturers to communicate — it generates group aliases and redirects incoming mail to e.g. cellular phones or e-mail addresses (according to personal configuration).

Most important modules were developed by students taking part in *Master seminar on Databases* and *Master seminar on Software Engineering*. Here are some examples:

1. *Roles* The main subject of this project was the development of the system of roles meant as privileges to select, insert, delete, and/or update rows and columns of database tables. Such roles can be freely defined in USOS and then assigned to USOS users. Every user can have a few roles assigned but only one of them is active at a time. The whole USOS interface is transparent to the system of roles. A user cannot read data it is not authorized to by his role. When a user tries to modify records he can select but not update, the “insufficient privileges” message is being displayed and the operation is aborted. The interface (Oracle forms) have to be designed and implemented in compliance with these requirements (what means that all Oracle programmers have to be aware of

them and write code in a proper way). The solution is very flexible since no Oracle programming is involved in defining a new role, and it can be done any time the need arises.

2. *Filters* The system of roles solves the problem of authorized access to subsets of data, but does not solve the problem of having to deal with excessive amount of data. What is needed is the flexible mechanism of filtering the data. In USOS such functionality is delivered by a system of filters. A filter has its name and can be defined by some SQL statement. Filters can be freely made available to users, and various sets of filters can be accessible on various Oracle forms. On each such form one of the filters is defined as default. No Oracle programming is involved in defining and configuring filters.

3. *Requirements of study programs* That module delivers flexible mechanisms for defining requirements of study programs. Compliance with requirements can be checked for one student or for a group of students (e.g. all 1st year students of mathematics).

Currently some Master projects in progress concern modules for handling students dormitories, financial aid and scholarships, class schedules, diploma supplement (a document established by the European Commission, Council of Europe and UNESCO/CEPES in order to improve the recognition of educational credential). Smaller modules are developed as Bachelor projects.

Recently we started a new experiment. A student writes a specification of a new module — he gathers requirements, prepares use cases, delivers them to the clients asking their opinion, then prepares final version of the requirements specification. Next he designs the implementation and describes it in another document. The documentation is passed to another student with the task to develop the module strictly according to the specification. The student-programmer generally does not communicate with the student-designer. The completeness and precision of the design is tested by the quality of the final product. In this pilot experiment we test UML and Rational Rose set of tools.

The early student projects were generally focused on producing particular functional modules. Recent projects are more diverse, focused on design and methodology, more software engineering oriented. The examples are *Describing USOS in the Unified Modeling Language*, *Calibrating and maintaining large scale databases*, *Designing user-friendly interfaces*, *Standards for database applications* etc. Software reengineering, data mining, database profiling, software modeling — these are the topics we want to cover in future projects.

After three years of using USOS in a variety of faculties and universities we also start to re-design and re-program some of the modules. A Master project may thus consist in getting information on users' satisfaction of the particular module and coming up with the suggestion of new solutions and functionalities.

Master and Bachelor Theses describing USOS modules have determined layout and structure of chapters. Each thesis contains, among others, technical documentation, user manual and administrator manual of the module involved. Software written has to adhere to standards (which were formulated in one of the thesis).

3.2 Software engineering methods and techniques involved in USOS development

In the beginning stage of the project it was developed in accordance with the CDM Fast Track methodology ([3]), designed by Oracle, which is a standard for enterprises conducted in RAD technology. The important techniques described in the paragraphs to follow were used in the project.

For each task in the project its **priority** had been determined and **strict time limits** were specified for conducting this task. This was particularly important due to the needs of the end-users expecting deliverables on time, and due to psychological aspect of keeping active involvement of participants and developers.

Workshops and **seminars** were very effective technique for information gathering and decision making. Their particular importance implied from the scale of the project involving universities spread all over Poland. We organized workshops at least every few months to keep universities informed about the progress made and to involve them in decision making about the possible direction of software development. Students often discuss their modules with tutors and colleagues, presenting them on seminars.

Prototyping was used from a very early stage in the project. The character of the project allowed us to deliver prototypes quickly, soon after having agreed on the first functional requirements. Prototypes were delivered to end-users and helped in getting quick feedback on functionality and usability of the product. Early prototypes offered limited possibilities for automation of various procedures for groups of students (students could only be handled individually), supported only few most important reporting facilities and almost non global statistics. Nevertheless they were used by administration officers who could comment on interface, test correctness of implemented routines, or point out missing options. The close contacts between designers, programmers, and customers were one of the most valuable aspects of developing software in-house. In particular students could discuss requirements with "users" — administration officers in their department — having them often in the same building.

Iterative development was driven by feedback from end-users. They worked with prototypes and came up with ideas for modifications. Requirements were not fixed at the start point of the project, but were developed along the way.

Since various tasks of the project were conducted by various groups of students we separated those tasks on the basis of the size of the group and time deadlines for the expected functionality. **Partitioning** of the overall project into smaller tasks was an important activity, driven by deliverables expected at each milestone.

Last year due to growing scale of the project we decided to start using **UML** ([2]) and **Rational Rose**. Oracle design tools play a main role in design and implementation phases but do not support sufficiently the requirement analysis process. The Unified Modeling Language can be applied to various areas of software development, such as data modeling, enhancing practitioners' ability to communicate their needs and assessments to the rest of the team. UML can also be used to describe the complete development of databases from business requirements through the physical data model. In USOS use cases modeled in UML become standard tools supporting talks between developers and clients for working out system requirements.

Practices applied in USOS in many respects match the rules of **Extreme Programming** (cf. [1]). The most important ones in USOS are: plan in a short run and in a long run, conform to standards, make it simple, work closely with the clients, integrate constantly, often deliver prototypes.

Students taking part in the project are exposed to various methods, techniques and tools, have possibility to compare them and judge their usefulness for various software engineering tasks and learn how important it is to adhere to the rules of rigid software engineering routines. These methods and tools are not just educational toys, they are real "products" used in industry projects.

3.3 Educational benefits of the USOS project

Whenever we describe the scheme of system development applied in our faculty many doubt that it can work in a long run. Negative examples are given of projects carried out in a university environment which failed without delivering any useful outputs. To our own surprise in our case the situation is opposite. The longer project runs the more stable, mature and diverse in character are sub-projects carried out by students. It seems that the large application like USOS brings out new ideas which may be worked on by students on various platforms, from different perspectives, using a plethora of tools and methods.

The most important difference between usual academic projects and USOS projects is enormous: the latter HAVE to deliver a final product and the author HAS to take part in its deployment. Students join teams, practice new software engineering methodologies, validate various software development techniques, play various roles in the project (programmers, sub-project leaders, system administrators etc.) They have to work in a strict time regime, use software version management systems, adopt standards, prepare and issue software distribution packages, use bug-tracking systems, feel responsibility for the final product, use flexible solutions (the produced code will be maintained and reworked by somebody else). Students work in teams and always have a tutor — they may discuss emerging problems either with the tutor or with the team members. They deliver presentations and have to defend their solutions. On the other hand they also learn how to talk with clients, who are not IT experts. Last but not least — they use the system themselves, being members of academic community.

We teach by example and practice. If a student cannot understand what "user-friendly interface" really means we can make him sit for a day at the help-desk for administration officers! Or make him responsible for training end-users how to use Oracle forms! Next time he will really make the best effort to design a simple, self-explained and intuitive interface. If he does not believe in the importance of thorough testing he can be made responsible for gathering bug reports from angry clients.

The code is often refactored when old solutions are replaced with new better ideas. Some of the students fail (even if they produce software, it is not attached to the official version of USOS), majority succeed — all learn "programming in the large", absorb "good practices" of software engineering and working culture. It is the best way to educate future leaders of software teams and improve their management skills.

Software companies producing commodity software rarely can afford experimenting with various techniques, methodologies and tools in the same project. We can — USOS becomes a testbed for such experiments. For example the project manager can waste "human resources" to test how web modules would behave with various open source databases. Software companies in stress of a hitting deadline may compromise quality or standards of the application — students can not, their Master degree may depend on adhering to the standing rules.

Let us summarize major advantages of the project (e.g. [9]):

1. students take part in all stages of a real software development process, including software deployment and maintenance;
2. students learn how to be team members, how to collaborate with colleagues from various groups, study programs, and even faculties;
3. students learn how to use professional tools, meet quality and performance requirements, apply standards, feel responsibility for the product, confidentiality and integrity of the handled data;
4. students get involved in the project even not participating in it personally (every student of the faculty is aware that software is developed in-house by the faculty members and computer science students). They offer advice concerning interface and functionality of the system, take part in testing;
5. last but not least the academic community obtains the high quality software.

There is also another aspect of students' involvement in the development of USOS. They obtain valuable experience which should help them to start their professional careers. Some of them may even by hired at universities deploying USOS.

Last but not least we want to point out the possible problem of the whole enterprise: projects like USOS demand much higher involvement of the academic staff, often exceeding the ordinary academic obligations. On the other hand, however, they may drive research in the area of software engineering giving the possibility of experimenting with various methods and techniques in a controlled environment.

No formal methods have yet been applied in the project.

4 Summary

Tempus project ended in December 2001. In 2002 UCI (University Center for Informatization) consortium of Polish high education institutions, similar to Swedish LADOK [5], was established with the purpose to support further development of USOS and its gradual deployment at the participating institutions,

After creation of UCI the organizational structure of the project changed. UCI hired a Working Team which is responsible for carrying the project, appoints a project leader, sets task and priorities for the team. The USOS Working Team has been located at the Faculty of Mathematics, Computer Science and Mechanics of Warsaw University. The main task of the group is to deliver help-desk, support software deployment, gather bug reports, debug and maintain software, prepare documentation, issue distributions and collect new requirements. The cost of these tasks is covered from the fees of the member institutions based on the project budget. Fees are low and devoted for basic system maintenance.

The mainstream design and programming is still done by student groups. Projects run by students are approved by the team, which is also responsible for integrating the projects' results with the system. Sometimes in one sub-project work together hired-programmers and student-programmers ("hired-programmer" may also mean a student hired to develop a particular piece of software). There is always a clear distinction between those two ways of involvement. For some students that may even be an extra motivation — they may get an opportunity to earn money if they prove their value first.

The profile of the student projects changes, they are more diverse, more software engineering oriented. The number of USOS Master and Bachelor Theses grows steadily — there have been 15 by now, many more are on the way. USOS provides a unique opportunity to integrate education in software engineering with production of commodity software, to involve students in a real project, teaching them software engineering methods and tools along the way. We are determined to continue that model of software development.

In April 2003 USOS is used on daily basis at five universities (at pilot faculties), a few others are at various stages of deployment. At Warsaw University we started academic year 2002/03 with USOS at four faculties. The next year some pilot modules (students' scholarships, registration for foreign languages courses) will be used at all faculties.

References

[1] K. Beck, *Extreme Programming Explained: Embrace Change*, Addison-Wesley, 2000.

[2] M. Fowler, K. Scott, *UML Distilled: A Brief Guide to the Standard Object Modeling Language*, second edition, Addison-Wesley, 2002.

[3] S. Gylseth, *Using CDM Fast Track, Oracle's DSDM Compliant RAD Approach*, Oracle Corporation, 2000.

[4] Home page of Bureau for Academic Recognition and International Exchange, Warsaw, Poland, `http://www.buwiwm.edu.pl`.

[5] Home page of LADOK consortium, Umea, Sweden, `http://www.ladok.umu.se`.

[6] Home page of USOS, Warsaw, Poland, `http://usos.mimuw.edu.pl`.

[7] Home page of USOSweb, Warsaw, Poland, `http://usosweb.mimuw.edu.pl`.

[8] P. Klint, J.R. Nawrocki, editor. *Proc. Software Engineering Education Symposium SEES'98*, Scientific Publishers OWN, Poznan, 1998.

[9] J. Mincer-Daszkiewicz, *Developing Commodity Software in Academic Environment* (in Polish), III Krajowa Konferencja Inzynierii Oprogramowania — KKIO'2001, Otwock, 2001, pp. 225–236.

[10] J. Mincer-Daszkiewicz, *Student Management Information System for Polish Universities*, Eunis'2002, The 8th Int. Conference of European University Information Systems, Porto, Portugal, June 19-22, 2002, pp. 271–281.

[11] J. Mincer-Daszkiewicz, *Developing and Deploying Commodity Software in Academic Environment* (in Polish), KKIO'2002, IV Krajowa Konferencja Inzynierii Oprogramowania, Poznan, Poland, October 15-18, 2002, pp. 299–314.

[12] J. Mincer-Daszkiewicz, *USOS — Student Management Information System For Polish Universities*, SAIAC'2002, Joint Int. Conference on State of the Art in Administrative Computing, Tartu, Estonia, November 18–20, 2002.

[13] I. Sommerville, *Software Engineering*, 6th ed., Addison-Wesley, 2000.