How to achieve good skills in Software Engineering – a practical way to train software developers

Dr. Tick, József

Budapest Polytechnic, Hungary, tick@bmf.hu

Abstract: Although the theoretical background of software engineering is important in software development, practical skills are significant in everyday software analysis, design and implementation. To achieve good skills in software engineering, a practice oriented training seems to be the best way. Software engineers have been trained under the cover of technical informatics major for 12 years in the Janos Neumann Informatics Faculty of the Budapest Polytechnic, earlier in the ancestor institution, the Informatics Faculty of Kalman Kando Technical Polytechnic. In this paper I will summarise the corner points of the practical side of software engineering education and of the permanent course development, which continuously monitors and follows the latest trends in software engineering. I will focus on presenting how important practice is next to the theoretical education for fresh software engineers.

Keywords:Software Engineering, Education

1 Market requirements facing software engineers

Due to a high demand in software products nowadays, software development has been industrialised recently, meaning that not only the number of software systems but also their size and complexity have grown lately. Quality and reliability are among the most important criteria set up for software systems, which means not only the quality and reliability of the software itself but those of the development process as well.

In order for a software system to be marketable it should be user friendly, customised and it also should fit to standard operating systems. Time and cost of development, software maintenance, and good professional solutions are no to be neglected at all.

Fresh graduates in software engineering should be aware of all the above requirements. They should be able to become an active member of a software development team and work effectively almost immediately after graduation. They should learn new technologies, be always up to date and they should have good routine to react to the short- and long term changes of the market.

2 Objectives and goals of software engineering education considering market requirements

Objectives and goals of software engineering studies follow below:

- introduction into the science of software engineering, teaching the necessary theoretical background
- training to be able to solve complex, large, software intensive projects
- teaching of software project management
- teaching the basics of software development methodologies
- training to use Computer Aided Software Engineering (CASE) tools
- teaching of practice oriented, well applicable, marketable knowledge
- importance of team-work, emphasis on team roles and training of team roles

The development of the field of informatics, market requirements and a demand in human force necessitated the training of highly qualified specialists in software engineering. This means the training of such software engineers who are skilled to develop larger-sized software systems in teams adapting the latest, up-to-date methodologies that are accepted in this profession. Our goal is to train such experts who are familiar with the theory but have practice oriented knowledge, who are well skilled, have experience in teamwork as well as in developing of small-, and/or medium-sized software systems, who are able to apply methodologies and use the CASE tools based on these methodologies.

3. The necessery theoretical and practical knowledge and skills to achieve goals

The necessary knowledge to achieve the goals could be divided into two parts:

3.1 The theoretical knowledge

The theory of software engineering developed enormously during the last 40 years. It is not easy to select a subset of these knowledge, which:

- contains the necessary basic theories,
- is up-to-date
- is easily understandable by students
- helps students in the “life long learning”
- is coherent and has a strong cohesion
- helps practical training

Taking the above viewpoints into account and considering the goals in 2. the following fields were selected to teach:

- software life cycle models
- software project management
- system analysis
- software requirement analysis
- software planning
- structured software development methodologies
- object-oriented software development methodologies
- user interface design
- software reuse
- software quality assurance

3.2 The practical knowledge and skills

On the basis of the experiences in the software industry the chance for a fresh graduate in software engineering to find a job depends to a great extent on the fact how quickly the employer can put him/her into action in a working software developing team. The integration for fresh graduates in software engineering in a team is made easier considerably if the students learn adequate pratical knowledge and skills before graduation. That is why the practical training has the following features:

- “real life” complex projects solved by students in teamwork during the semester (a three month project for a group of 4 students)
- strong relation with software developers in the software industry (to involve them in the practical course as project leader)
- case studies are solved with the application of structured and object-oriented methodologies and modern CASE tools by 4 member teams, that are either self organising or created randomly.
- development with milestones
- presentation skills like documentation and presentation

4. How to implement the training

Taking the above goals and features into account and considering that only part of the informatics students can specialise in software engineering but emphasising that each informatics student needs to be introduced into the basics of software engineering, the material to be taught – avoiding overlapping at the same time - has to be defined in two separate units:

- Basic course (i.e. Introduction into Software Engineering)
- Specialisation for software engineers

The given frame for the courses are

- 2 semesters (60 hours) for the basic course
- 3 semesters (150 hours) for the specialisation

Subjects introduced follow below:

- For all informatics students:
    - Introduction to software engineering (30 hour theory and 30 hour practice)

- For students specialised in software engineering:
    - Software engineering (60 hour theory and 30 hour practice)
    - Software development programme environment (CASE tool) (30 hour theory and 30 hour practice)

However this paper focuses on the practical aspects of software engineering training, it has to be stated that practice could be derived only from theory. Nowadays the main question is, whether

- only Object-oriented methodology and software technology should be taught, or
- combine Object-oriented methodology and the structured methodology.

Furthermore, what should be taught in the object-oriented field

- only UML, or
- other methodologies like OMT, OOSE etc. as well.

At the drawing up of the entire software engineering course material and also at the evaluation and selection of methodologies, it had to be considered that informatics students who choose software engineering specialisation have already taken their software final exam before starting their software engineering studies in the specialisation. It means that they are already familiar with the theoretical background and practice that is sometimes well beyond skill level in the fields of not only structured but in object-oriented programming as well.

In order to select object-oriented methodologies out of the elaborated and wide spread ones and to integrate them in the course material a criteria system had to be worked out.

The most significant items in the criteria system follow below:

- methodology should be wide spread and well applicable in practice
- standards should be de facto and de jure
- methodology should be well proved and well settled
- a clear and easy to survey notation system should be used
- methodology should be supported by an accessible CASE tool
- it should give possibilities to modelling with simple tools.

4.1 Which OO methodologies to integrate into the training

By paying attention to the above viewpoints, the course material of Software engineering has been continuously developed and changed so far. Considering the

„marketability” of the subject and also the educational requirements that somehow seem to oppose to a certain extent: the topics and the longer time constant, a gradual, „slower” change of methodologies was necessary in the course material.

In order to evaluate experiences only closed educational periods are advisable to be examined. As a result of the selection and earlier evaluation carried out on the basis of the above mentioned viewpoints, four object-oriented methodologies have been integrated in the course material so far.

- Coad/Yourdon Object-Oriented Analysis and Object-Oriented Design methodologies were firstly integrated into the course material in order to insure continuity, due to relationship between structured and object-oriented methodologies, in order to demonstrate methodology developments and changing and due to the methodology’s simple notation system.
- The easy-to-survey and well elaborated Responsibility driven methodology, which was worked out by Wirfs-Brock and their co-authors, was secondly introduced in the course material due to the famous „Class Responsibility Cards” as well as its responsibility attitude, its application of the hierarchy and collaboration graphical notations, its good modelling technique, its simple and „cheap” technique and its easy utilisation in practice.
- Object Modelling Technique (OMT), which is finely detailed, illustrated with examples and labelled with the name of Rumbaugh, was thirdly fitted in the course. This methodology was chosen due to the methodology’s development, its excellent modelling possibility, including not only the static but the dynamic model as well, its simple, easily applicable and well known notation system, the methodology’s spreading and a good support by a CASE tool.
- Unified Modelling Language (UML), which is the most widespread at present and is considered and used as standard, was integrated last in the course material. An excellent CASE support and an opportunity to gain a skill-level knowledge that is promptly marketable also stand by the choice of UML.

The UML only opinion, which can be seen mainly on the courses of American universities and colleges, should be highlighted. According to this opinion, the methodological side of software engineering education does not deal with methodologies that preceded UML. As far as I experienced the integration of the above four methodologies in the course well represent the development and the logical improvement leading to UML. The presentation of interrelationships between methodologies provides a much deeper understanding for students. The presentation of methodologies that preceded UML will help students to understand what comes after UML.

By all means, the selection of the above object-oriented methodologies and their gradual integration into the software engineering specialisation can be considered as successful regarding the several year experiences, the positive feedback and a growing demand.

However, the course material is continuously renewed and updated since new methodologies and new CASE tools appear and gain ground in the market, furthermore, a demand from the users' side is rapidly growing as well.

4.2. Which methodologies should practice support and to what extent?

Practising each of the above selected methodologies during the lab exercise would not be rational even if we consider that students have already learned structured methodology. The available lab time is dedicated not only to getting to know the CASE tool which supports the selected methodology but it also should meet the criteria that follow below:

- Students should become familiar with teamwork and should learn team member roles, the duties and the responsibilities in the team.
- In the course of solving the project students should learn the phases of software development from the analysis through design, implementation to testing.
- Students should get to know project documentation, the importance and techniques of milestones and reviews of the software projects.
- Students should have a look into the difficulties and tactics of cooperation and communication with the user.
- Students should learn and practise project presentation during and at the end of the project.

Taking the above criteria system into account, such a two-term practice course has been set up that supports the recently most accepted methodology namely the UML. The selected CASE tool to be taught is Rational Rose, which is wide spread in Hungary and well accepted in the profession.

The second term practice course is built on the first one meaning that students have the opportunity within the framework of team work to:

- develop a larger scale project,
- learn and practise the use of the CASE tool up to skill level,
- have a thorough view into the entire software development process,

- present case studies, standard solutions and professional tricks during the elaboration of their projects, and
- make the framework of their software development similar to real circumstances, to employ time and effect constraints.

5. Experiences collected through education

The education of practice oriented software engineering presented above had a highly positive feedback, and also conforms to the principle of the polytechnic, namely, it is strong in teaching practical skills in line with theoretical background.

In the light of the last three year experiences we can say that thanks to the building and organisation of the practice course fresh graduates can almost immediately be put into action as software developers in small- and medium-sized firms and in larger software development companies as well.

The rate of employment among fresh graduates in software engineering is quite good. Almost everyone out of the students specialised in software engineering has a job even before taking the final state exam.

One and two years after graduation an opinion poll is made among old students. According to the results, both the methodologies selected for the theoretical education and the practice oriented feature of the lab exercise have proved to be a success.

Questionnaires that were sent out to employers came back with a positive feedback as well. Students' skills, their routine in different project work and solving problems, the professional level of the developed software and the students' thorough grounding in the profession were positively valued.

In summary we can state that a practical way to educate software engineers that demands teamwork but gives adequate theoretical background as well is a good one to train a professionally highly skilled software engineer who is employed by the firms with pleasure.