Strategies for Industrial Relevance in Software Engineering Education

Claes Wohlin and Björn Regnell
Dept. of Communication Systems
Lund Institute of Technology, Lund University
P.O. Box 118, SE-221 00 Lund, Sweden
Phone: +46-46-222 3329, Fax: +46-46-145823
E-mail: (Claes.Wohlin, Bjorn.Regnell)@tts.lth.se

Abstract

This paper presents a collection of experiences related to success factors in graduate and postgraduate education. The experiences are mainly concerned with how to make the education relevant from an industrial viewpoint. This is emphasized as a key issue in software engineering education and research, as the main objective is to give the students a good basis for large-scale software development in an industrial environment. The presentation is divided into experiences at the graduate and postgraduate levels respectively. For each level a number of strategies to achieve industrial relevance are presented. On the graduate level a course in large-scale software development is described to exemplify how industrial relevance can be achieved on the graduate level. The strategies on the postgraduate level have been successful, but it is concluded that more can be done regarding industrial collaboration in the planning and conduction of experiments and case studies. Another interesting strategy for the future is a special postgraduate program for people employed in industry.

Keywords: education, software engineering, industrial relevance, large-scale software development, technology transfer

1. Introduction

A key issue in university software engineering education is to achieve industrial relevance. With industrial relevance we mean that the education prepares students so that they are ready to cope with large-scale software development. It is also important that students are aware of the challenges and proven techniques related to industrial development of software. This is crucial as software engineering implies large-scale software development, which means that the focus is primarily on engineering aspects rather than on a specific design method. Specific methods are needed, but they should be viewed from a larger perspective when they are applied. Thus, we must have a comprehensive view of software development, instead of looking for silver bullets (Brooks 1987). We would like to embrace the definition of software engineering according to IEEE (IEEE 1990), that is "software engineering means application of a systematic, disciplined, quantifiable approach to development, operation and maintenance of software". In particular, this means that software engineering education is not only about software design methods and programming languages. We must be able to teach aspects such as requirements engineering, process improvement, software testing and software quality just to name a few important areas. This is a challenge as the areas are rather difficult to learn solely through reading software engineering literature. Thus, ways are needed that allow

us to emphasize the industrial relevance, and in particular the application of sound engineering principles to the development of real software.

The industrial relevance is important both at the undergraduate, graduate and postgraduate education. In the Swedish system, undergraduate refers to a Bachelor degree, the graduate education results in a Master's degree, and the postgraduate education includes a Licentiate degree and a Ph.D. degree. The differences are further discussed in Section 2, where a background to the Swedish system is provided to explain the context of the experiences presented.

This paper highlights some opportunities and experiences of software engineering education. The objective is to present and inspire educators to consider different alternative ways of performing education in software engineering. Most of the ideas are applicable to other areas as well, but the experiences are from software engineering.

The focus is primarily on approaches that we have applied to make both the graduate and postgraduate education relevant from an industrial perspective. In the graduate education, the education is primarily provided in large classes with 30-150 students taking the courses. This gives some specific problems, as it is not feasible to provide a personal education for each individual. This is, however, possible at the postgraduate level, where the students are employed at the Department and participate in research projects. Thus, different ways to achieve industrial relevance and stress the need for engineering of software have to be found. The objective here is to present our experiences of some different approaches we have identified. Experiences from both graduate and postgraduate education are presented.

The paper is organized as follows. In Section 2, the background of the students entering the graduate and postgraduate education is given to provide a context of the experiences presented. Important aspects and some tested strategies to obtain industrial relevance in the education for both postgraduate and graduate education are presented in Section 3. The presentation starts with the postgraduate level as it provides some essential information for the graduate level. In Section 4, some of the experiences are highlighted and some future possible directions to further improve the education is discussed.

2. Background

2.1 Department and the Master program

In Sweden, a Master's degree in engineering corresponds to 4.5 years of full-time studies, which is equivalent to 180 Swedish credit points (i.e. one year equals 40 credit points). At Lund University, there are 9 Master's Programs in engineering, and software engineering is involved in two of them, i.e. Electrical Engineering and Computer Science & Engineering. It is normal to enter Master's Programs after nine years of compulsory education and 3 years of upper secondary education at an age of 19-20 years. Undergraduate studies resulting in a Bachelor's degree are carried out at other parts of the university. At an Institute of Technology, most of the students go directly for a Master's degree. Thus, our Department is primarily involved in graduate and postgraduate education, although we have lately been more and more involved in a new undergraduate program in Software Engineering.

The Department of Communication Systems is one of 10 departments providing courses for students following the Electrical Engineering or Computer Science & Engineering Programs. The number of employees in the Department is 30, including academic staff, postgraduate students, and administrative and technical personnel.

The Department gives nine courses within the Master's Programs. The Master's courses range from telecommunications courses to theoretical courses in queuing theory, and three of the courses are software engineering courses. The Department has traditionally focused on systems analysis, and in particular, performance analysis of telecommunication and computer networks. In the middle of the 1980´s, it was, however, realized that it was not possible to teach a system view of large systems without taking the software into account. Thus, it was decided to provide a number of software engineering courses, with a particular focus on large-scale software development, where the software development process was judged to be a critical success factor.

2.2 Graduate students in the Master Program

The software engineering courses are optional in both the Master's Program in Computer Science & Engineering, and for the students in Electrical Engineering who have chosen to specialize in telecommunications. The courses are normally taken in the fourth or even fifth year, with a few students taking them in their third year. This means that the students are very familiar with computers, programming and mathematics. They have completed a number of courses in mathematics, statistics, physics and electronics, which, of course, are combined with a number of courses on computers and computer science. The computer science courses, which are of particular interest, include programming with different paradigms, compiler technology and object-oriented design. The students have worked in software development projects, but normally only with design and programming issues. A number of the students taking the software engineering courses normally pursue their Master's thesis work at the Department.

2.3 Postgraduate students in the Licentiate and Ph.D. Program

The students entering the postgraduate education are normally recruited from the students having a Master in either Electrical Engineering or Computer Science & Engineering, and they have typically focused on the courses provided by the Department and also performed their Master's thesis at the Department. The Department normally employs the postgraduate students, unless they are industrial postgraduate students. This is further discussed in Section 3. The postgraduate students are thus viewed as part of the staff at the Department, and they participate normally in a research project, which usually provides some funding. Some postgraduate positions are, however, provided by the university and no other funding is required, but for most postgraduate positions external funding is required. The external funding implies that the actual research is determined by the projects for which we manage to get external funding. Government agencies or companies fund the external projects.

The postgraduate program includes two degrees, Licentiate and Ph.D. The Licentiate degree consists of one year full time courses and one year of research work, which are presented in a thesis. A Ph.D. degree is received after another half year of courses and an additional one and a half year of research work. The Licentiate work is included in the Ph.D. degree (The Licentiate degree can be viewed as "mid-term exam" corresponding to "half" Ph.D., all the doctoral students graduated at the Department also have a licentiate degree.). Thus, a Ph.D. degree means that you have taken one and a half year of courses after your Master's degree, together with two and a half years of research work which are presented in a dissertation, and defended at a seminar. The actual division between courses and research studies may differ slightly between different departments. But, in general, it means that a Licentiate degree equals two years work and a Ph.D. degree is four years of full time studies. The average time to finish a degree is, however, normally longer as the

postgraduate students are expected to contribute in the work at the Department, for example, as teachers in courses at the graduate level.

A professor or associate professor supervises the postgraduate students during their research studies. The professor is normally responsible for formulating the research questions and applying for funding of the research projects.

3. Important aspects and tested strategies

In our work to achieve industrial relevance in our software engineering education, we have identified a number of important aspects to address. We have also tested a number of strategies in our teaching work, which we have found successful. The important aspects and the tested strategies are summarized in Figure 1 below. The aspects and strategies are divided between the graduate (Master) program and the postgraduate (Licentiate and Ph.D.) programs.

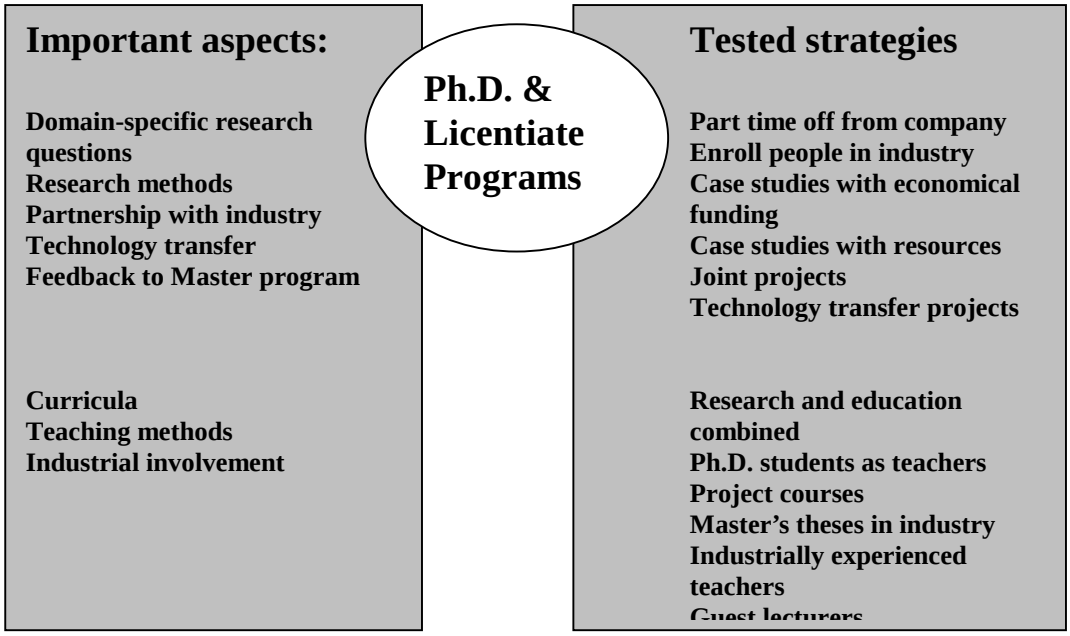

FIGURE 1. Some important aspects and tested strategies for industrial relevance in software engineering education

3.1 Licentiate and Ph.D. Programs

This section outlines a number of aspects in software engineering research, which we have found important in order to achieve our objectives of providing industrially relevant research and postgraduate education. [illegible]ber of aspects are presented which we have found important based on [illegible]ome of the aspects are highlighted through some examples of stra[illegible]d together with our postgraduate students in their Ph.D. studies at

Based on our experiences the following aspects are important for successful postgraduate studies in software engineering, where successful means (1) industrially relevant research, (2) the results are easy publishable, and (3) the students accomplish their degrees.

- *Domain-specific research questions at an application-oriented department*: Most software engineering research is probably conducted at a computer science department. We have not conducted any formal study, but based on experiences from looking at the affiliation of people at conferences and of authors in journals we think this is a fair statement. We have, however, found that there are benefits with conducting research in software engineering at an application-oriented department. This does not mean that the research should not be carried out at a computer science department. We would rather like to stress that there are other opportunities and that they can be successful. Our Department is focused on telecommunications as its main application area, and some of the research is directly aimed at solving research problems faced by the telecommunication industry. Thus, the research being conducted is in itself applied. Software engineering research is inherently applied, hence from this perspective it fits well into the Department. Moreover, the focus on telecommunications provides a natural application area for the research results. The objective is that the research results should be generally applicable, but we have come to the conclusion that telecommunications provide a good starting point for evaluating our research results. Another benefit with a specific application area, in our case telecommunications, is that the Department has good industrial contacts in this area based on its special focus.
- *Research method*: A key aspect is the research methods used. We do not believe that we can continue to present new methods without being able to have some empirical evidence. Thus, we need to use empirical methods in our software engineering research, which means that experimentation (Basili, Selby & Hutchens 1986) and case studies (Kitchenham, Pickard & Pfleeger 1995) are essential to succeed. Experimentation is important as it provides a better understanding of the methods investigated, and it also works as an important step before it is possible to transfer any research results for broader use. Case studies are an important tool to identify research questions, and as a way of monitoring industrial software development. In particular, they can be used as a means for introducing new technologies and evaluating the outcome before and after a specific change in, for example, the software process.
- *Partnership with industry*: A close relation with one or several industrial partners is crucial to enable the postgraduate students to perform case studies in industry, and to have a natural environment for evaluating new research results. Furthermore, it is important to have a close relation with industry when identifying important research questions to ensure that we, as researchers, address relevant research questions. The research group in software engineering has an active collaboration with ten companies involved in software development. Moreover, the Department takes an active part in regional, national and international collaborative networks. The networks include both industry and other universities. The objective is to improve the cooperation both between universities and between academia and industry. The industrial partners support the initiative economically (mainly through personnel resources) and by providing their environments as potential study objects.
- *Technology transfer*: An aspect, which is closely related to the close relation with industry, is methods for technology transfer. These are important when we would like to transfer research results to practical use in industry. It is not enough to present new results, we must be able to show empirical evidence to succeed in technology transfer from academia to industry. Thus, we would like to stress that the research methods discussed above are one important means for technology transfer. Other important aspects are to recruit people to postgraduate studies, who have a background in industry. They could either work full time or part time as postgraduate students. Finally, another opportunity is to let postgraduate students spend part of their studies in industry in order to identify new research questions and to transfer the results so far.

- *Feedback to Master Program*: The graduate courses in software engineering have been improved in a natural way, incorporating new research results, since the postgraduate students as part of their employment participate in graduate teaching. This is obviously important for the graduate courses, but it also provides excellent feedback to the postgraduate students. It is not until you are able to teach a subject that you really can claim that you know it in depth. Furthermore, the graduate students are always curious and have a number of questions concerning the subject, which provides important feedback to the postgraduate students.

The aspects listed are all important, and to highlight how we have instantiated some of them, we would like to provide a list of examples of postgraduate studies which are along the lines outlined in the items above.

Currently, we have tested six different strategies of managing the industrial relevance and the close contact with industry. At least one postgraduate student is, or has been, working according to each item presented below.

1. *Part time leave from company*: We have had one postgraduate student who was working part time with his studies and part time as a consultant in software engineering. This allowed him to obtain industrial experience, which was useful in the research. The studies on the other hand gave him good insight into the state-of-art in software engineering, which was useful for him in his work as consultant. Persons of this category are thus formally employed part time by the university.

2. *Enroll people in industry*: A particular challenge in postgraduate studies is to enroll people who are working full time in industry, but is given the opportunity, by their employer, to conduct research (aimed at receiving a particular degree) as part of their normal work. This typically means that some of the work, which should be conducted any way should be of the type that it can be used within their postgraduate studies. The main difficulty in this situation is how to prioritize different types of work, and the problem is normally that the postgraduate studies have low priority from the company perspective. It is after all more important to have satisfied customers. It is, however, an interesting way of conducting postgraduate studies in software engineering as it really combines industrial work with postgraduate studies. This type of postgraduate studies requires strong commitment from the companies involved as the persons doing this type of studies are supposed to perform their research as part of their normal work.

3. *Case studies with economical funding*: Company funding is a third opportunity. In this case, a company is interested in investigating a particular issue, and prepared to fund a research project at the university. In this situation, case studies are often natural. The research question is often coming from a particular problem that the company has observed. The research is normally conducted by observing through case studies, and then based on the observations changes are suggested and the outcome after the change is observed. Hopefully, the change was an improvement.

4. *Case studies with resources*: Another possibility is that funding for the postgraduate student is not obtained from a company, but the company is prepared to put in resources in terms of people and perhaps also fund any additional travel costs due to the study. This can be a rewarding way of conducting research when the research question is formulated at the university, but it is found interesting by an industrial partner. This is not directly a joint project. The people from the university perform the research, although the company finds the work so interesting that they support the work through providing resources that help with information gathering and also fund, for example, some travelling.

5. *Joint project*: Joint projects between academia and industry are often a good way of working. This provides a natural channel for discussions and transfer of knowledge in

both directions. A particular opportunity for this is within in the European Community where a number of projects are conducted across several countries and between academia and industry. The only major problem is that large projects tend to generate a great deal of overhead in terms of administration. In this particular case, we see that people from industry and academia work together in a project addressing a particular problem.

6. *Technology transfer project*: Finally, we would like to highlight the opportunity to conduct a specific technology transfer project. This is in particular recommended when the postgraduate students have been pursuing their studies for a couple of years. In our system, it is quite natural to do this in close relation with finalizing the Licentiate thesis. This means that the students can transfer and try to introduce some of their research results presented in their Licentiate thesis, and at the same time they identify drawbacks and unsolved research issues. These form a good basis for continuing the research studies and finalize their Ph.D. degree.

These approaches have through experience been found to be important success factors in our postgraduate study program in software engineering. It is, however, important to realize that the major impact on industry is through the graduate program, where a large number of Master students are taking the courses and hence are exposed to software engineering. The broadest spreading of good ideas is probably achieved through integrating research results into the graduate study program.

3.2 Master Program

There are a number of important aspects that we consider when trying to achieve industrial relevance in software engineering at the Master level. These aspects include:

- *Curricula*: It is of course important to carefully select the topics that we teach, so that they reflect what industry needs. Later in this section, a list of our other courses in software engineering is provided, and in Section 4 we briefly describe our project course devoted to large-scale software development. It is described to exemplify how industrial relevance can be integrated in the graduate education.
- *Teaching methods*: We think it is vital to address the ways we teach, especially in software engineering. A problem-based teaching method with realistic scenarios of software development that mimic industrial best practice is an important element to achieve deeper understanding. We do not believe that a traditional "lectures from a book"-method alone will achieve the educational goals. This is illustrated through the course description in Section 4.
- *Industrial involvement*: The credibility of education is increased as the students see that people in industry struggles with the problems and solutions that we teach. If it is possible to have some degree of industrial involvement in the courses, e.g. through guest lecturers from industry (see strategy 6 below), we believe that this is very positive. Other forms of industrial collaboration in education are reported in (Kornecki, Hirmanpour, Towhidnajad, Boyd, Ghiorzi and Margolis 1997) and (Harrison 1997).

We have tested a number of strategies in our software engineering courses. These strategies are first described and then the software engineering courses at the Department are briefly described to provide the context in which the strategies are implemented.

1. *Research and education combined*: It has been found essential to involve the same people in both research and education. This includes both the academic staff and the postgraduate students. We are of the opinion that this way of working provides important feedback to the research in the same time as it ensures that the latest findings in software engineering are included in the courses. If some people just work with

teaching, there is a risk that they will become isolated from new developments and from contacts with industry.

2. *Ph.D. students as teachers*: The involvement of postgraduate students in the graduate courses is important as it helps their research. It is also beneficial in terms of making it easier for the graduate students to relate to the lecturers, as they have quite recently taken the courses themselves. The postgraduate students are often highly motivated as teachers since they would like to make others interested in their area of research studies (if nobody is interested then what is the value of the research studies?). Furthermore, the presence of postgraduate students in the graduate courses facilitates recruiting of new Ph.D. students.

3. *Project courses*: Software engineering is about working in large projects developing software systems, hence projects in the education are an important ingredient. This includes both software development projects and projects more relating to investigating different issues in the area of software engineering. In particular, it is important to teach aspects of software development, which normally arise when scaling up and to relate science to engineering (Shaw 1990). The importance of project courses for industrial relevance in the graduate education is one of the reasons why our large-scale software development course is viewed as crucial for the software engineering education at the Department. This course is described briefly in Section 4.

4. *Master's theses in industry*: The Master's thesis work is often made in cooperation with industry. Industry provides the thesis questions and the students are supervised both by an industrial representative and one person from the Department. The actual topic of the thesis and questions to be investigated are formulated in cooperation between industry and the university. This type of Master's work ensures the industrial relevance, and provides an excellent basis for graduation and entering the industry after examination. It is up to the supervisor at the university to ensure that the educational objectives of the project are maintained when the work is performed in industry. The academic supervisor is also the person responsible for actually approving the Master's thesis, the industrial representatives have no voice in this decision.

5. *Industrially experienced teachers*: The objective of the software engineering courses is to teach an engineering approach to software development, imitating and teaching aspects that are crucial for large-scale industrial software development. Thus, it is beneficial if some of the lecturers, including the postgraduate students, have industrial experience from software development.

6. *Guest lecturers from industry*: To further emphasize the industrial aspect, we usually try to invite guest lecturers from industry, who can pinpoint and highlight some of the important aspects of software engineering based on industrial experience. These lecturers are mostly viewed as informative and stimulating by the students, as it highlights that the issues taught in the courses actually are needed and used in industry.

The above items are implemented into three courses at the Department. For example, the researchers and postgraduate students are teaching in the courses below. We try to bring in industrial guest lecturers and people with industrial experience to all three courses. The master theses are formulated within areas, which are taught in the courses below in cooperation with industry as indicated above. The three courses provided in software engineering at the Department are:

- *Large-scale Software Development* (5 credit points)*:* This course is the flagship among our software engineering courses. The course is by the students rated as one of the best courses within the Master program, and it is described in some more detail in Section 4.

- *Software Engineerin*g (Sommerville 1996) (4 credit points): This course provides an overview of software engineering, and also includes a project in which the students are faced with a problem in software engineering.
- *The Personal Software Process (PSP)* (Humphrey 1995) (6 credit points): The PSP was taught for the first time in the autumn of 1996. The objective is that the students should learn how to develop software systematically, and base their work on processes, process improvement and measurement.

3.3 Risks with close industrial contacts

Above, several positive aspects have been reported with close industrial contacts in software engineering education. It is our opinion that it is positive to involve industry in education, but it should be pointed out that there are also risks with having a close collaboration with industry. This includes both graduate and postgraduate education. Some of the risks, which should be evaluated for every specific case, are:

- *Short-term education*: There is always a risk that the education becomes too short-term, in other words that the courses contain mostly material which the industry needs today or that the research questions are aimed at the problems of today. It is important that courses in software engineering include both sustainable knowledge and some of the methods and techniques, which will become valuable. The same goes for the research within the postgraduate education. Industrial collaboration must not mean that the current problems are solved. The university research must aim for investigating new solutions in identified problem areas, and the perspective should be more long-term than for the research departments within industrial companies.
- *Integrity and dependence*: The lectures and researchers must be independent from the collaborative companies. This is important to be trustworthy in front of the students. Furthermore, it is important to be able to address relevant research issues to enable the researchers to come up with new ideas and not be governed by the current directions at one specific collaborative partner.

It is important to be aware of these risks and address them properly.

4. Large-scale software development project course

4.1 Introduction

The large-scale software development course emerged during the late 1980´s, with the objective of providing a course which, based on a standardized process model and design language, imitated a real software project as closely as possible, given the limitations of a university environment.

The Department views a realistic software development course as one of the cornerstones of software engineering education. The students need hands-on experience, where they have to work together in a large project using industrial tools and can actually execute their software on a real piece of hardware, in this case a telephone exchange. At the end of the course, the students are able to use their own software by phoning each other over the exchange.

It has not been possible to find any good generally available literature as the course is based on combining a target machine, a development tool, language and a process model from different sources. Therefore, the material provided for the students has been developed at the Department for this particular course (MD110 1998). This material is

normally complemented with some general software engineering articles, for example (Rook 1986), and slides from the guest lecturers.

125-150 students take the course normally each year.

4.2 Development environment

The development environment has briefly been mentioned above, it is, however, one of the unique features of this course and we would like to point out some of the building blocks in some more detail. The environment can be divided into seven building blocks.

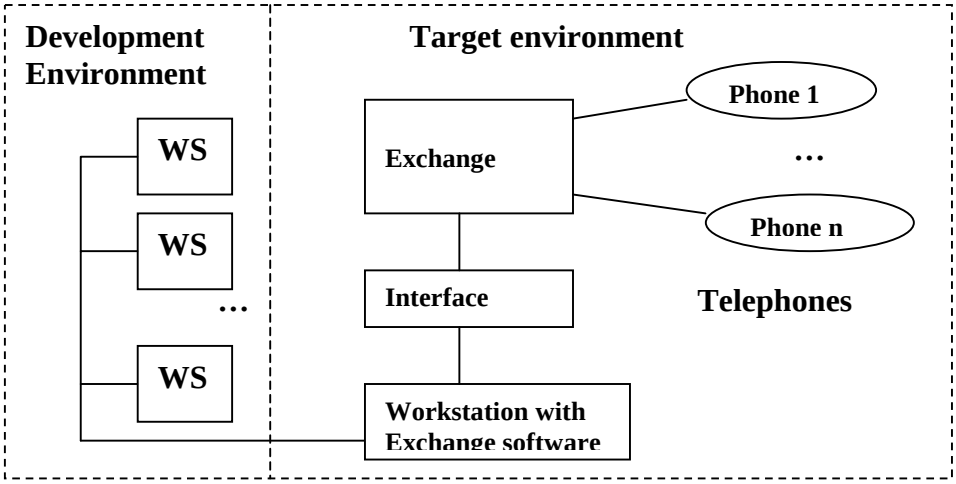

FIGURE 2. The basic system

- *Exchange*: The exchange is a digital switch with a capacity of 100-10000 subscriber lines. The exchange is a modular SPC system (Stored Program Control) supporting integrated voice and data communication. This means that the students are exposed to and develop software for a real system during their graduate studies.
- *Workstation*: Development is carried out in a Unix environment, i.e. a number of workstations connected in a network. At the end of the project, the software is copied to the workstation connected to the telephone exchange. This workstation replaces the microprocessors in the exchange. This means that software executed on the workstation replaces the switching and controlling functions normally made by the microprocessors of the exchange, see Figure 2. This enables the execution of software in any language, although SDL (Specification and Description Language) (ITU 1988) and (Belina 1991) and C are used in the course.
- *Process model*: A somewhat simplified version of the US Department of Defense process model 2167A (DoD 1988) is used. This is a waterfall model and the objective is to provide a standardized approach, rather than experimenting with a more sophisticated process model. The model, as used in the project, is outlined in Figure 3.

In particular, the baselines in the model are emphasized to provide checkpoints in the process (see also the next subsection).

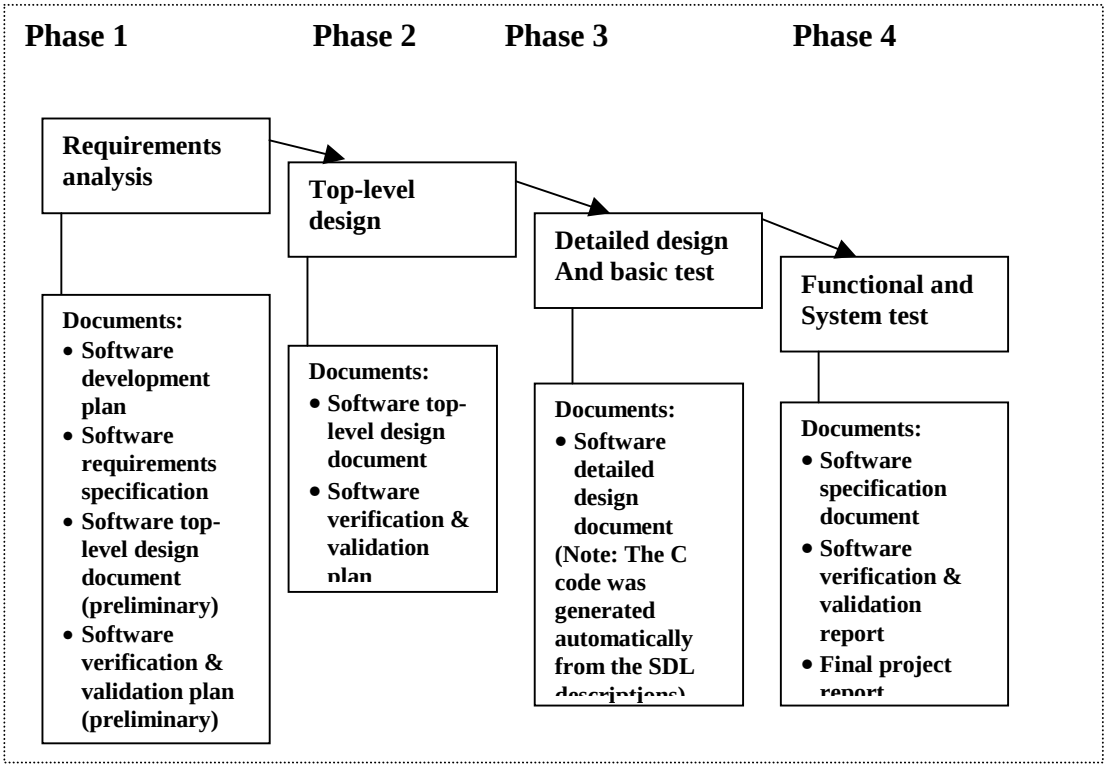

FIGURE 3. The waterfall development model

- *Project*: The software for normal telephone calls is already available on the workstation, and the students are therefore asked to provide some additional subscriber services. This, to some extent, turns the project into a maintenance project, as their software must work together with the existing software.
- *Services*: The students are asked to implement the following four services: Call Forwarding Unconditional, Take Call, Debiting and Maintenance (groups with 14 students do not develop the "Take Call" service). “Call Forwarding Unconditional” is the service where upon ordering calls is moved to another subscriber number

unconditionally. The “Take Call” service implements the opportunity to take calls to one number from another telephone by dialing a code and then picking up the incoming call.

“Debiting” implements both normal debiting for calls and additional debiting resulting from other services, for example, double debiting for call forwarding. The “Maintenance” service includes adding and deleting subscribers, and changing telephone numbers. The services implemented are not described further in this paper. The students are given quite straightforward customer wishes concerning these services, and no information is provided concerning service interaction, special cases and erroneous cases. These should be specified by the students and communicated to the customer, a role played by a Department representative.

- *SDL*: To avoid the course becoming a programming course, it is essential to work with a well-defined design language. In this particular case SDL was chosen due to the availability of tool support and the fact that it has been standardized for telecommunication systems. Other high-level languages would have done as well; SDL is not a prerequisite. SDL is based on extended finite-state machines with states and signal sending. The students are taught SDL very briefly, which means that they learn to use the basic concepts without becoming SDL experts.
- *SDT*: SDT (SDL Design Tool) allows the students to define their services in the graphical syntax of SDL. The service descriptions can then be analyzed syntactically and semantically, before the C code is generated automatically. Thus, the code is generated from the detailed design in SDL. This means that the emphasis can be on project and process issues, rather than on programming problems. SDT runs on workstations and after being thoroughly tested the C code generated is copied to the workstation connected to the telephone exchange (see Figure 2). Thus, development can be carried out on any of the available workstations, while the final execution takes place in the workstation connected to the switch. Moreover, it means that the students use a commercial software development tool within the course, hence making the development realistic from an industrial perspective.

4.3 Project

The background to the project and some basic knowledge of the process model, SDL and SDT are provided through a series of lectures. The students perform most of the, work in their project groups. The groups consist of 14-19 students, which are divided into 7-8 subgroups. One subgroup acts as project leaders, one subgroup is a system group responsible for keeping the system together, three or four subgroups are development groups, where each group develops one service, and finally two subgroups are test groups testing two services each, and with joint responsibility for the system test.

The roles of the subgroups are combined with roles played by Department personnel. Staff from the Department plays the roles of customer, external quality assurance personnel and, of course, technical experts, who can help the students when they run into difficulties. The customer reviews the material produced twice during the project and also performs an acceptance test at the end of the project. It should, however, be noted that executing the process correctly is viewed as equally important as the final product.

4.4 Conducting a project

The first two weeks are quite hectic, with a number of lectures providing the students with an introduction to the development environment and the project. The students are allowed to form groups on their own, and the project leader subgroup is appointed. This

subgroup is then responsible for the division into the other subgroups. This should be settled after the first week, and they should familiarize themselves with their work in the forthcoming weeks.

The requirement analysis documents should be written during the second week, and should be ready for customer review during the third week of the project. The requirement specification should form a baseline after the review. In the fourth and fifth weeks of the project, the students mainly work on the top-level design document. This document is reviewed in the fifth week, which means that the customer can check that the project is heading in the right direction.

Work during the latter part of week fifth and into week eight is focused on the detailed design document and testing, and the product should be delivered at the middle of the ninth week. A project report should be delivered together with the product. This report should contain information about the work, and in particular metrics should be reported concerning time expenditure and faults found during inspection and testing. The delivered product is acceptance tested in the ninth and tenth weeks, and the students are given feedback on their performance.

4.5 Summary of experiences

One conclusion is that the application approach, with actually having an exchange and the opportunity to run the students' software on the exchange, is very important. This makes the projects more realistic for the students, and the lecturer can also exemplify how the software is actually developed and installed in a real switch. Some of the people teaching the course have an industrial background from the telecommunications field, and the guest lectures often emphasize the issues stressed in the course. Therefore, it is concluded that the closeness to an application is an important asset, when trying to teach software engineering realistically.

An important issue in relation to research is to let the course evolve with the research being conducted at the Department. This means two things. First, the introduction of an experience factory concept in the course provides a basis for research related to learning organizations. Secondly, the changes in the course related to the experience factory form a basis for experimental research. Experimentation in software engineering is, by many for example (Basili et. al. 1986) viewed as one important research method to better understand software development. Thus, experiments are used as a means in research, and the course provides a good opportunity for trying out new ideas and carrying out experiments. It is, however, important to stress that the experimentation should always be carried out so that the students gain from it.

In the future the course will likely be complemented with roles related to software quality assurance, which could be played by the students. The quality assurance personnel will be responsible for evaluating process conformance and follow up on project data as part of the course.

5. Future directions

The issues discussed concerning graduate and postgraduate education are, from our point of view, important factors for successful software engineering education. They provide a good basis for educating software engineers, which meet the requirements of industry. This does, however, not mean that the education cannot be further improved. In particular, we have a number of ideas of how to improve the postgraduate education.

The ideas concerning improvement of the postgraduate education include the following strategies that we plan to test in the future:

- *Case studies and experimentation*: We view empirical software engineering research as essential, and the objective is to increase our research into experimentation and case studies.
- *Software engineering shop* (primarily experimentation shop): To further emphasize the experimental approach to software engineering, and at the same time increase the interaction with industry, a software engineering shop is discussed. The objective of the shop is to allow companies to come with their questions, which can form the basis for stating hypotheses, which can be evaluated using experiments.
- *Industrial Licentiate programs*: The Licentiate exam is by industry often viewed as a good degree, since it in 2 years provides deepened insight into the area without spending very long time on academic specialization. The Licentiate students learn research methods, which then can be successfully employed in an industrial context. Thus, it is often possible to encourage industry to enroll some interested individual in a Licentiate program.

The graduate education has been improved recently; two of the courses described briefly above have been introduced into the graduate programs during the latest three years. Thus, there is primarily a need to stabilize and make minor adjustments and improvements of the courses.

In our long-term plan for improving and extending the curricula at the graduate level, we have identified the following idea to be further pursued:

- *Virtual software enterprise*: In (Mayr 1997), experiences from teaching through real projects in a “virtual software enterprise” are described. We are currently working on a plan for introducing an extension of this idea in a 6 semester course, corresponding to 12 credit points, that will “employ” selected students for 3 years in a “company” that will deal with real software product development and maintenance. The first year is devoted to development work, the second year will focus on project management issues, and the final year will focus on strategic product planning and software enterprise management issues on an executive level.

Finally, it should be noted that the results from the Department are not only research results, but also a number of well-educated people in software engineering. Sometimes the emphasis is too much on research results, but the greatest impact on industry is probably through a good educational program. This is one of the main reasons why we feel it is essential to have a close relationship between graduate education, postgraduate education and research.

Acknowledgements

The author would like to thank the companies, and the people at those companies, who over the years have contributed to the software engineering courses at the Department, in particular: Ellemtel AB, Ericsson Telecom AB, Q-Labs AB, Telelogic AB, Telia AB, Telia Engineering AB and Telub AB. Moreover, we would like to thank all the students during the years that have participated in our large-scale software development course, and also contributed to the improvement of the course through valuable comments on the content and conduction of the course.

We would also like to thank the personnel at the Department of Communication Systems, Lund University and the Department of Computer and Information Science, Linköping University for providing challenging environments for education and research.

This work was partly funded by The Swedish National Board for Industrial and Technical Development (NUTEK) grants 1K1P-97-09673 and 97-09690.

References

Basili, V., Selby, R. W. & Hutchens, D. H. (1990), 'Experimentation in Software Engineering', IEEE Transactions on Software Engineering 12(7), 733-743.

Belina, F., Hogrefe, D. and Sarma, A. (1991), 'SDL with Applications from Protocol Specifications', Prentice-Hall, London, UK.

Brooks, F., (1987), 'No Silver Bullet: Essence and Accidents of Software Engineering', IEEE Computer 20(4), 10-19.

DoD (1988), 'DoD-STD-2167A, Defense System Software Development'.

Harrison, J. V. (1997), 'Enhancing Software Development Project Courses Via Industry Participation', Proceedings Conference on Software Engineering Education & Training, IEEE Computer Press, 192-203.

Humphrey, W. (1995), 'A Discipline of Software Engineering', Addison-Wesley.

IEEE (1990), 'IEEE Standard Glossary of Software Engineering Terminology', ANSI/IEEE Standard 610.12, New York, USA.

ITU (1988), 'ITU Recommendation Z.100: Specification and Description Language (SDL)', Blue book, Volume X.1.

Kitchenham, B., Pickard, L. M. and Pfleeger, S. L. (1995), 'Case Studies for Method and Tool Evaluation', IEEE Software 12(4), 52-62.

Kornecki, A. J., Hirmanpour, I., Towhidnajad, M., Boyd, R., Ghiorzi, T. and Margolis, L. (1997), 'Strengthening Software Engineering Education through Academic Industry Collaboration', Proceedings Conference on Software Engineering Education & Training, IEEE Computer Press, 204-211.

Mayr, H. (1997), 'Teaching Software Engineering by Means of a Virtual Enterprise', Proceedings Conference on Software Engineering Education & Training, IEEE Computer Press, 176-184.

MD110 (1998), 'MD110-project', Dept. of Communication Systems, Lund University, Lund, Sweden, (in Swedish).

Rook, P. (1986), 'Controlling Software Projects', Software Engineering Journal 1(1), 7-16.

Shaw, M. (1990), 'Prospects for an Engineering Discipline of Software', IEEE Software 7(6), 15-24.

Sommerville, I. (1996), 'Software Engineering', Addison-Wesley.