WORKING- VS. EDUCATIONAL PROCESSES IN SOFTWARE ENGINEERING VS. CDIO

Daniel Einarson

Computer Science, Kristianstad University, Sweden

ABSTRACT

The relatively short history of IT can unfortunately point out a number of failing projects concerning missing deadlines, functionalities, low quality, etc. Theories and techniques have been developed, to meet inherent problems and challenges. But also software process models, that is ways of working, where several typical activities within software processes have been emphasized. Still, the use of software also seems to bring even further requirements on new techniques. A conclusion is therefore that, besides from some core fundaments, inherent parts of IT are, by necessity, evolving in themselves.

When it comes to educational systems, an appropriate set of theories, techniques, and principles should be taught to prepare for working in software industry. Still, this is not enough. A software engineer actually needs to be able to handle all the steps of a software process. That is, educational systems have to find ways to support teachings, not only in theories, techniques, and principles, but also in ways of working that hopefully should correspond well to the practices of software industry. Furthermore, students should be gained by getting educational support to meet and handle the ever changing future.

This contribution presents project based approaches where the process of developing the project result should have several benefits. First, it should provide a basis for training core practices of Computer Science, second it should prepare for software processes, i.e., ways of working in software industry, and third, it should aim for students being responsible for self learning. Especially, the third point is significantly important in a discipline of ever changing techniques. Inspiration is taken from well known Software Process Models. Such are models are shown to lie close to the CDIO initiative. Software Process Models are discussed, comparisons with CDIO are provided, as well as a case study on a project based course.

KEYWORDS

Project based learning, Education concepts, Software Engineering, Cooperative work, Evaluations for groups.

INTRODUCTION

Software Engineering is probably one of the most recent engineering disciplines. As such there have been major requirements put on that discipline to undergo appropriate transformations from immaturity to higher levels of maturity. This does not only relate to the need for that discipline in itself to be reliable, but also to the ongoing almost explosive change in amount of software that more and more is integrated into society. The latter, on

the other hand, is driven by pure commercial requirements, as well as requirements on security, health, entertainment, and so on.

According to David Parnas, a well-known expert in Computer Science and Software Engineering, software is the most challenging and complex engineering construction in human history. Large scaled software may include millions and millions of lines of code, where high requirements are put on functionality as well as quality, on levels of code as well as on software as a whole. Preparing students for real life software engineering tasks require exhaustive practice in programming skills, theory, and techniques. However, to meet the challenges of developing large scaled software products this is not enough.

The ability of fulfilling a software project corresponds to the ability of controlling and steering the working process that takes you there. That process may start from a position of low or even no knowledge of a specific subject, to a resulting software product, ready for deployment. That is, besides from technical skills, students are also gained by being prepared for the working process itself. Therefore, educational systems should provide ways of experiencing students, in Computer Science and Software Engineering, in such directions.

Several models of Software Engineering processes have been proposed. Typically, those suggest iterative and incremental ways of working, with frequent more or less formalized meetings for discussions and feedback. This, among other things, implies allowance for sets of requirements to mature throughout the process, avoiding mismatches between, on one side, expectations from stakeholders, and on the other side implementations of developers.

From the point of view of educators, the value of creating valuable and well motivated structures for educational purposes should be clear. The relatively new and innovative approach of CDIO proposes educational forms for engineering studies generally. However, in the perspective of the author, it is of course especially interesting to see CDIO in the context of Software Engineering studies. Interestingly enough, we can see that there are several similarities between suggested working process models of Software Engineering, and educational forms suggested through the CDIO initiative.

The contribution of this paper is mainly two folded, discussions of on working processes, and a case study in that context. An investigation on working processes of Software Engineering, and comparisons with the CDIO initiative will be provided. Moreover, project based learning of the educational system of the author's home department will be outlined. A case study on a project based course will also be outlined. Problems, outcome of the course, and possible improvements will be discussed.

SOFTWARE ENGINEERING PROCESS MODELS

By Computer Science we often mean the core theories, techniques and principles behind programming computers. By Software Engineering we normally mean something more, an engineering discipline covering all aspects of developing a software system, including handling requirements, design, implementation, test, deployment, and maintenance. Terminologies and concepts may however vary. Being a programmer may mean that you write a complete program. A software engineer rather implements only parts of a bigger system, in a team of developers. A software engineer also has to regard the mentioned aspects on software engineering ([7]). Furthermore, by a Software Engineering Process we mean all the activities included to fulfil a software project.

Software Engineering as a field was born in 1968 at a NATO conference covering the, so called, software crises ([8]). Software crises reflect on chronic failures of large software projects to meet requirements, quality, budget, schedule, etc. From that point several

Software Engineering process models have been proposed to meet the software crises. Some are listed below.

The Waterfall Model

The Waterfall model ([11]) is a general and natural model for software development. The model describes how to go through a number of steps to fulfil a software product. The steps cover activities such as, defining the requirements and analysing those, designing solutions, implement the software system, test the system, and finally put the system at an operating state and maintain it. Figure 1 illustrates this. The model also points out a temporal order where one step should be finished before starting the next. If, at operational state, failures are discovered at one of the previous steps, those are corrected and the following steps are then corrected according to that. This seems natural; however, a typical problem lies in the originally stated requirements, where those have an ability to evolve during the whole process. This is a result from increasing knowledge of those from the points of views of both the stakeholder, and the developer team. Furthermore, requirements may change because of different kinds of changing circumstances. Changing a requirement is normally considered very costly since that implies changing dependencies in the following steps.

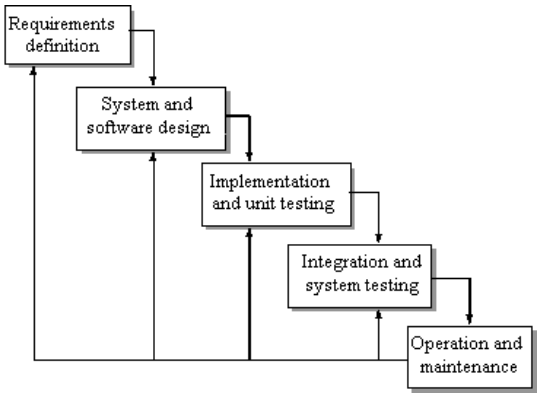

Figure 1. The Waterfall model ([11]).

Iterative and Incremental Models

To meet the problem of changing requirements, Iterative and Incremental process models are proposed. One such model is the Rational Unified Process (RUP, [10]), which instead suggests working in smaller increments of disciplines, such as, requirements definition, design, implementation, and tests. This is briefly further discussed in the sequel.

A selected number of requirements are analysed, designed, implemented, and verified. At the end of an increment there is a meeting with customers to discuss the system so far, how the requirements are met, and to discuss the proceeding work. In this way, and through an exhaustive documentation, the work should be monitored and validated, and serve as a basis for building the next increment. That way of working is iterated until the project is fulfilled.

Briefly, the process model includes four main phases:

- Inception: Set the business case, and define the main requirements
- Elaboration: Develop an understanding of the problem domain and design the system.
- Construction: program and test.
- Transition: Deploy the system in its operating environment

For each of those phases a varying amount of effort is put into the disciplines of requirements, design, etc. This is illustrated in Figure 2, where time scale goes from left to right, and effort on disciplines per phase is outlined. An interesting point of view here is that also project management is included as a discipline of RUP. Project management, here involves planning the process, controlling it, handling risks, etc. For more information, [10] is outlining this process model, for educational purposes.

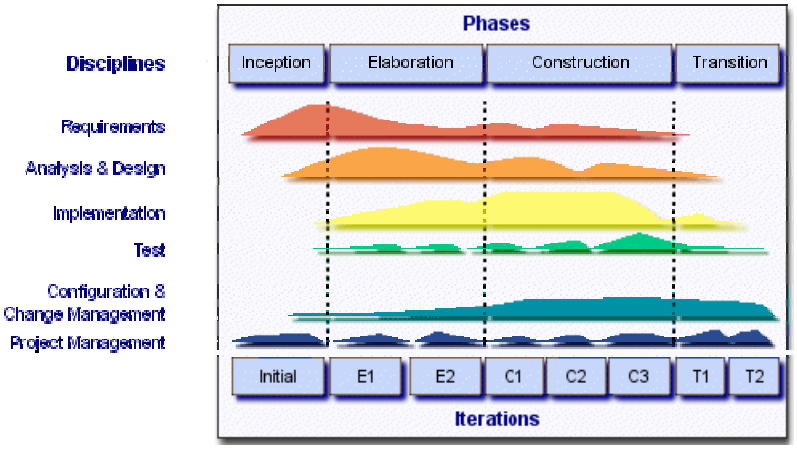

Figure 2. Phases and Disciplines of RUP ([10]).

eXtreme Programming, an Agile Process Model

While e.g., RUP meets changing and unclear requirements, criticisms have been pointed out towards RUP, and other similar process models. Those, among other sides, concern the exhaustive documentation, where that is a part of controlling the process from a management point of view. The main intension is to be even more agile to changing circumstances, and exhaustive documentation is here considered counterproductive. Here is the software rather considered the most important, not documentation. While, e.g., RUP, may be considered as a management choice of process model, agile process models are more considered as the programmers choice. Still, criticisms towards agile process models have also been pointed out, and concern the inappropriateness for large scaled projects. For projects with need for teams larger than about twenty developers, other forms are recommended.

The eXtreme Programming ([14]) approach put several aspects to the extreme. For instance, design should be simple, and tests are done totally integrated with implementation. Furthermore, the customer is one in the team, constantly available for questions on requirements, for feedback on prototypes, and for prioritizing amongst tasks to do next. Those aspects, and others, are captured and illustrated by Figure 3. For more information please see, for instance [14].

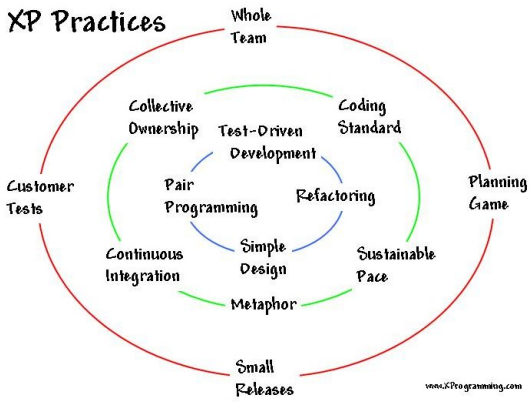

Figure 3. Practices of eXtremeProgramming, ([14]).

Summary

The immaturity of the field of Software Engineering, the lack of experiences, the software crises (see discussion above), in combination with an almost explosive growth of the amount of, and need for software, have forced that field to come up with solutions. Process models regard ways of working, guidelines to reach the goals of fulfilling software projects in time and within budget, and meanwhile meeting the requirements of the customer, in qualitative ways. Still, even if a process model seems to be appropriate that does not guarantee that the project will be successful. It can only increase the probability for that.

From an educational perspective the software engineering process models provide a two folded interest. On one hand, if software industry maintains process models, it is of interest to train students in those for them to get appropriate experience. On the other hand, the ways of working proposed through process models may bring inspiration to educational processes. That is, theories, techniques, and principles, may be learnt in a context of an educational process similar to a software engineering process. How this has been approached is further discussed at the section Project Based Education.

SOFTWARE ENGINEERING PROCESS MODELS VS. CDIO

While Software Engineering process models have its origins in software crises, the CDIO initiative ([13]) origins from a desire from industry for more mature students. Maturity does here relate to capabilities of putting core engineering knowledge into practice in real world projects. Besides from real world problems, real world projects also concern a process of working phases to move from originally stated problems to fulfilled solution under operation and maintenance. Typically, at an abstracted level there are four main phases. The first phase concerns setting the business case, catching the main requirements, and estimating required resources, in terms of time, money and personal. The second phase typically stands for high level design, the third for more fine granular design, and constructing the product. Finally the forth phase stands for deploying the product, putting it to an operating state, and eventually maintaining it.

The innovative CDIO initiative provides an educational support to meet the industry desires for more appropriately prepared students. Here the letters CDIO, stands for Conceive – Design – Implement – Operate, that in turn in much corresponds to the mentioned four phases. For more detailed information on this please see [13].

Now, returning to the previously described Software Engineering Process Models, the same four main phases have also been outlined in that context. No matter what model is used we can see the same core activities, however, more or less parallelized. Main goal of the phases of the RUP model are (for more detailed information, please see [10]):

- Inception, Establish the business case for the system.
- Elaboration, Develop an understanding of the problem domain and the system architecture.
- Construction, System design, programming and testing.
- Transition, Deploy the system in its operating environment.

Again we can see the close correspondence between the concepts. So far they may be interchangeable. That is, so far, following a Software Engineering Process Model for software development in education could as well be done by following a plan for using CDIO. Now, CDIO have been put into rather exhaustive evolvement, and tested out in itself, and may therefore meet the mentioned desires from industry. The conclusion should therefore be that following a Software Engineering Process Model for educational purposes should correspond to similar desires from software industry. That is, in the sequel, when such models for project based learning are discussed, they could as well be seen as CDIO based projects.

Actually, a more detailed analysis is required for comparisons. As an example, assessing the work is done in different contexts for different purposes and because of this, probably looks different; however, this is put outside the scope of this contribution. In the sequel it is assumed that assessing project based educational work is done according to how to measure progress of Software Engineering Process Models.

PROJECT BASED EDUCATION

According [6], project- and problem based learning forms are attending more and more interest in Swedish educational systems. Pupils of elementary school more and more work in contexts of specific themes, where teams of pupils drive their own process to get and present the appropriate information. This in turn will probably contribute to a sense of familiarity for those later on, when it comes to project based learning in teams of higher education.

Motivations

In [5], several examples on project- and problem based learning at Swedish universities have been pointed out. Furthermore, [5] presents a number of motivating points for developing software systems in project teams, and especially reflecting that on higher education.

- First, if systems are supposed to be interesting enough, those are seldom developed in a scope of one or two persons. Furthermore, real world problems normally inherently have a scale and complexity that is not possible to manage for only a few persons. You may need ten, twenty, or even hundreds of people to fulfil a complex task.
- Second, in software industry most often projects are performed in teams of developers. Providing practice in this should therefore also be essential in higher education for preparing students for software industry ways of working.
- Third, the Higher Education Ordinance of Swedish National Agency for Higher Education points out several desired competences engineering students should have that probably may be developed though project based educational forms ([5]).
- Forth, projects often involve needs for investigating techniques and knowledge that in advance may be unknown for the developer. Thus, project based learning in itself

supports overcoming the challenge of taking responsibility to actively find new solutions to problems. This is perhaps especially essential in software development where we can see an increasingly variety of proposed techniques for an increasingly amount of different purposes.

The forth point above illuminates on the impossibility in the ambition of providing a complete picture of all parts of computer science to students, perhaps especially because of constant changes in techniques. All we can offer are restricted overviews, and in depth studies of some selected concepts. The ambition we can have is to provide an appropriate fundament for further investigations. In [3] this is pointed out through:

”Universities are supposed to enable their students to engage in effective action in situations they are going to encounter but as the future is increasingly unknown, these situations are impossible to define in advance. What universities have to offer is knowledge and therefore you have to prepare for the unknown by means of the known”.

Finally, the background and the reasons behind CDIO ([13]), with impact and desires from industry on engineering studies, strengthen the motivations for project based learning even more. That and the growing interest in CDIO in itself may serve as great inspiration in developing project based learning forms further.

Project based learning at home department

The main focus on [5] lies on studies of project based learning forms outside of Sweden, especially China, and Arabic countries. The result shows a significant discrepancy between Sweden and those countries, in the use of such learning forms. The reason behind the studies of [5] is the quite great amount of students from those countries that have been, and are studying at our home department. This in turn, and also with respect to the above pointed out reasons for project based learning, has inspired us to introduce project based learning at several courses, starting already at the first semester.

At the section below, a case study with a project based course is provided. Typically students of that course have been involved in projects as listed below, where the author of this contribution has been more or less participating as a supervisor or teacher.

- Project at the end of course in fundamental programming, about 5 hp, first semester. The project is done with an iterative and incremental process style (see section on Software Engineering Process Models, for more on this). There is one project group meeting with a supervising teacher each week, during a five week’s period. For each meeting the students shall upload and discuss documents on requirements, risks, design, and project plan. Progress in documents and software product shall be shown, and that also forms the basis on which student’s course grade is assessed. A project group involves about four participants.
- Project at the end of course Object Oriented Programming, about 5 hp, second semester. This corresponds very much to the first semester project, but with impact from more detailed teachings in design principles, and programming techniques.
- Software Engineering, about 7.5 hp, second semester, year 2. The project is based on an eXtreme Programming (XP) process style (see section on Software Engineering process models, for more on this). It does not have one meeting per week between teacher and project group. Instead, there have been full days of working XP-like. Principles and techniques are covered through the project.
- Software Projects, on 15 hp, first semester, third year. In contrary to previous courses projects are here performed in large project groups. Project group meetings take place every twice week, where a number of documents should be uploaded and

discussed. Principles and techniques are covered through the project. More information on this course is provided in the Case Study section below.

Normally a project ends with full days of project presentations where the result of the work of the project groups are shown and discussed. In some cases this is done in front of a common audience and with invitations of newspapers ([9]). Grade is generally based on process, presentation, final software product, and project report.

A CASE STUDY

The case study is based on a course that covers 15 hp, and has been running at the first semester (autumn semester) of the corresponding students' third year, which also is the last year of those. The number of students was about 60. Each group were quite large, about 20 students, which in turn were divided up into sub groups of about 4 students. There were originally 3 groups, one of those was split into two halves, because of too hard cooperation problems. This paper will focus on the two remaining groups of about 20 students. Group 1 is a group with solely Chinese students. Group 2 is mixed of a number of nationalities, such as, from Sweden, Arabic countries, Cameroon, and more. That first division into project teams were done by teachers, where the decision was based on experiences from previous desires from students. The course was chiefly provided by one main teacher, which is the author of this paper (the course teacher). Besides from that, there were three co-teachers mainly assisting at developing and participating at some of the course labs.

System structure

The project itself concerns aspects of a so called *interactive house*. The technicalities of the project were presented at *STCC, Hangzhou China, 2010* ([4]).

In short the interactive house project should consider several distinct parts, including an embedded server and database, web based interfaces, remote controllers in terms of mobile phones, a physical simulation of a house (including light, fan, radiators …). Parts of an interactive house should, furthermore, communicate through well defined communication protocols. Figure 4 illuminates on the core system structure.

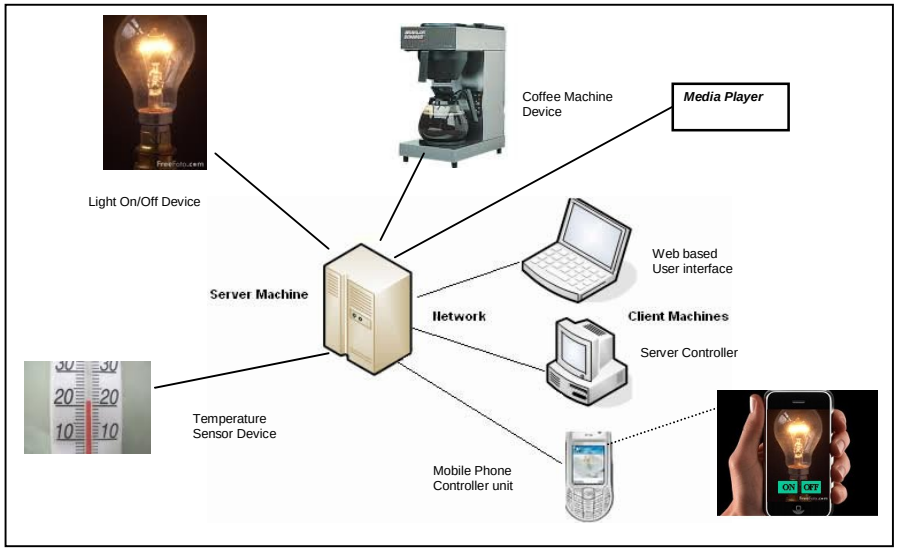

Figure 4. Illumination of system of the Case Study ([12]).

Communication towards the interactive house may be performed from both computers, and mobile phones, at the same time, and there may be any number of those. Mobile phones may be the most modern smart phones, or more traditional mobile phones. A media player could be built as a piece of software with a graphical user interface (GUI) from which pieces of music are selected and played. The simulation of the house is done through a small scaled physical house model where communication with light, door locks, fan, sensors for temperature, etc, is provided through an Arduino micro controller ([1]). Figure 5 illuminates on the latter. Other devices such as coffee machines, etc. are completely simulated in software.

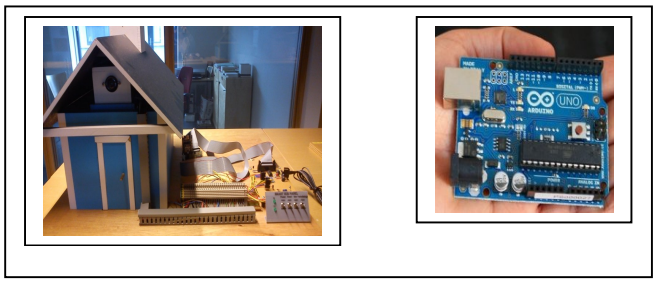

Figure 5. House simulation with Arduino micro controller board ([1]).

Project team structure

The project team was mainly divided into sub groups with respect to sub tasks of the project as a whole, see Figure 6. That was more or less decided from the start by the course teacher. That is, one group handled the server with a database, two groups handled the house devices, one for software simulation, and another for the physical house. Furthermore, the control units were developed by two groups, one for a web based user interface at a computer, and the other one for the mobile phone. Sub tasks developed in sub groups aims for parallel developments, however, that also requires well defined communication interfaces between functionalities. Therefore one group was especially dedicated to that task. Finally, one students was selected by the teacher as the Project Manager, with a leading role, and one student was elected Requirements Manger, with specific responsibility in communicating the system requirements between teacher and project team members. Moreover, each sub group had their own sub group leader.

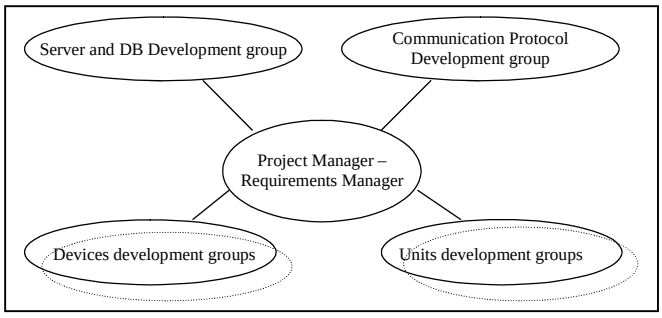

Figure 6. Project group structure.

All in all there were put high demands on the student groups to manage communication and negotiation to fulfil their main project task. Communication with course teachers was mainly

done through the dedicated Project Managers. Running the working process should here correspond close enough to well known Software Engineering process models where the teacher also plays a role of a customer or “super boss”.

Techniques, Theories, and Practices

To be able to develop a system of the case study, some theoretical and technical knowledge is needed. Moreover, knowledge of practices is desired to work in teams. First, at the point of the course start, experiences that students should have had from previous courses, and are crucial for the course, include:

- General programming skills
- Use of IDE (Integrated Development Environment) for developing software
- Software design and use of tools to develop software design
- Communication protocol for general distributed communication (such as, TCP/IP based communication)
- Digital techniques
- Running an iterative and incremental project (see e.g., Subsection *Project based learning at home department*)
- Configuration management, to handle repositories of developed pieces of software, and different versions of those.

The course will give opportunities for practising those subjects even more. Besides from those, the project of the case study also put demands on knowledge in further subjects, as listed below. Even more, the course in itself stands for a step in the educational progress where students are exposed to more professional tools, and ways of working. Further subjects taught at the course include:

- A more advanced and professionally used IDE, with plugins for integrating software design and programming.
- Developing software for mobile phones
- Developing software for web based interfaces
- Handling the physical house model
- Software Engineering subjects. More on requirements, qualities, design, etc. Also more on working processes.

The last point above was typically covered at lectures and practised during the course. Experiences on the first four points were typically provided through labs. The three middle points should be practised by all students. Still, while the labs provide a preliminary knowledge, sub groups especially covering those subjects need to dig more into needed implementation details, by themselves.

Outcome – students

An interesting point here is the big variation of nationalities amongst the students, in several cases with backgrounds in education systems where project based work seem to be quite uncommon. However, at the course of the case study, the students should have some skill in working with projects in teams (see, Subsection *Project based learning at home department*). This, together with use in English as the main educational language should aim for more or less fluent communication.

Another interesting point is that in several cases, students of one and the same sub group were living several tenth of miles from each other and only seldom with natural in person meetings. Yet another interesting point was that the groups should in much control their work

by themselves, where main control was performed by a Project Manager and delegated sub group leaders. Communication is therefore significantly critical.

Initial phase

At an early full class meeting the students were divided into project teams, with dedicated Project Managers. After that a Requirements Manager should be selected, the project manager should have a main role in this selection. Then the project teams should divide themselves into sub groups, everything should be controlled and steered by the project manager. Still, sub groups of Group 2, were typically very much geographically correlated. Swedish students formed one sub group, Cameroon students formed one sub group, and so on.

Very early the students agreed upon a technical basis from communication. Group 1 established common email lists to distribute information. Group 2 chose a Google based configuration management tool. For each comment uploaded at a common repository, each member of the project group should automatically get an email about the discussed subject. For the sub groups, communication was typically performed on more personal email basis.

All in all there was an active and interesting start. The first meetings with teachers were active with many discussions on design on parts of system, as well as on system as a whole. Much of the discussions concerned communication protocols between system parts. Those discussions were needed to be able to develop the system parts as independently as possible.

As time goes

After the first about seven weeks of the course, progress for both Group 1, and Group 2, seemed to decrease. Technical difficulties were handled, but especially communication between sub groups seemed to cease. Project Managers seemed to loose control over sub groups, and their activities, and little or even no result was shown.

Typical causes of the problems seemed to be: higher priorities put on parallel ongoing courses, low authority from Project Managers point of view, lower interest in the course than initially, a few students even decided not to follow the course.

Meetings with teacher seemed not to be as valuable as previously. Mostly the still active students thought they didn't get feedback enough from teacher, and that they were forced too much to tackle the problems themselves.

Finally, to handle the situation, the course teacher had to give an order to the whole project groups to attend at mandatory meetings led by the course teacher. Not only had the sub groups have to fulfil their own sub task, but also, everything should be integrated to fulfil the project as a whole, a challenge that is not trivial in itself. By the end of the course Group 1 seemed to be safe, while Group 2 was at the edge of failure.

Project presentation

The presentation of the project is a part of the examination, and is done by each student, from each sub group, from each project team. That is, one project team presents its result from the view of the results of the sub groups. A presentation is provided to all other co-students of the course, and covers the work that is done, design, qualities of the system, etc. Each sub group should also prove that their part work in isolation, that is, without involvement from the rest of the project parts. That is typically done by simulating the context,

and run the part in the simulated context. Thereafter a project team should show a fully integrated working system.

Group 1 have a good presentation, in detail as well as at a whole project level. The following discussions with audience and course teacher are also good. The general judgement is however that the limits could be pushed even further.

Group 2 were risky to the end. However, through hard and intense work they managed to fulfil their tasks. The result was surprisingly good at every level, and from several aspects. The final run of the complete integrated system was actually even surprisingly impressing.

Outcome – course teacher

In several aspects the course was new both to the students and the course teacher. Techniques that previously had not been used in education at the course teacher's home department were introduced in this course. The project group sizes, and project managers and sub group leaders, were new to both the course teaches and students. Leaving more responsibility to the students in this way introduced more work to the course teacher. Conflicts had to be handled, and one of the originally three groups had to split because of impossibilities in cooperation. This on the other hand led to special treatments for those groups later.

At course start a quite great amount of course material was exposed to the students. Lectures on techniques, Software Engineering, and the purpose of the project itself had to be provided to start the students' work as soon as possible. After the first weeks the students had a higher grade of self responsibility for their work, and feedback from course teacher was delivered through the project group meetings.

As described above, initial meetings and also lectures, were intense, active and exiting, not only to students, but also to the course teacher. After the first weeks, progress and interest from students' point of view, seemed to cease. After those first weeks, the course teacher also had fewer amounts of lectures, and a conclusion may be that this contributed to lower grade on activity and higher sense of insecurity from the students' point of view. Challenges of teachers typically relate to encouraging groups, and especially project managers, and group leaders to carry on. When the situation seemed to become more critical the course teacher found that the upcoming crises had to be handled with the whole project teams. Even though those meetings seemed to encourage the students further, the crises were still there. The partially successful outcome at the final project presentation was both a sense of success and relief also for the course teacher.

Further challenges lay in the evaluation of the student work. Also here the working process plays an important role since this should be integrated in that process. Each twice week students should upload documents on design, test plans, etc., to show progress of work. The documentation upload is done per sub group, and correspond to an iterative and incremental way of working (see Subsection *Iterative and Incremental Models*), where such documents are used to validate the state of the process. In the context of the course such documents provides a basis for evaluating student work and set the grades. Moreover, at the project group meetings those documents are discussed.

The result of uploaded documents mirrored the state of the course. Initially documents showed progress, however later there was a lack on progress or in some cases there were no uploads at all. From the course teachers point of view a situation turned out where there a balance had to be found between passing students, and failing them completely on the basis of a failed working process. Fortunately, through uploaded complementary documents later, the course teacher avoided the situation of failing students on that basis.

A conclusion may be that the course teacher should have taken a more authoritative role, more leading, and provided more careful comments on work done by students. Furthermore, the course schedule should be changed in favour of lecture contents more spread over the whole semester. Even more, project meetings should not only concern project managers and sub group leaders, but at least occasionally involve all project team members.

Course evaluation

There are mainly two ways of performing a course evaluation, a formal investigation, and an informal investigation. At the home department of the author the formal investigation is done through an electronic interact able document that is sent to all course participants. Thereafter, after the form is filled in, it is automatically returned and compiled together with other filled in forms. This has also been carried out at the current course.

Moreover, after the project presentation, there was a final mandatory meeting between course teacher and the different project groups. At that meeting several discussions on the outcome of results as well as on the course itself were taking place. All in all, the project and the course were considered very exiting. Below some outcomes of the discussions are pointed out.

On the positive side:

- Fun and challenging project
- The course aimed for studies of new techniques
- The course provided new insights on working in large groups

On the negative side:

- To big groups. Even though this was a good experience, the group sizes still were considered too big.
- Not enough feedback on work from course teacher.
- The authority of the student project manager was not strong enough. Support from course teacher was needed.

The formal course evaluation was in much reflecting the same comments on the course as the informal discussion did. However, the number of answers was actually too low for a significant course evaluation and validation.

Besides from the course itself, the course teacher could see that many bachelor thesis works could extend the core of the project in several directions. Such extensions concern security, usability, and more. A list of proposals for bachelor thesis projects was developed by the course teacher. The reactions from students that had taken the course were positive, and about all of the proposals were selected by students for bachelor thesis work.

Possible improvements

Both positive and negative criticism should be reflected upon. The course will take place again autumn 2011, and modifications should be considered. The outcome of bachelor thesis works will probably bring even more substance to the project.

The project in itself may be further specified, that is, the system requirements should be more precisely described. The course teacher should supervise the project more, at a level of system requirements, and qualities, i.e., should stress the requirements and system qualities even more.

Groups should be smaller; still there is probably a limit at about twelve to fifteen members of a project team. Below that number it may be hard to fulfil the goals of the project. The course teacher should probably have more meetings with the whole project groups to avoid dips and support ongoing progress. At such meetings the course teacher may clarify aspects of the project as well as on the work.

SUMMARY

To meet the requirements of engineering industry there is a need for substantial experiences in core knowledge, i.e., theories and techniques. Still, industrial engineering processes require more than that. Demands are put on flexible ways of using such knowledge appropriately. Moreover, fulfilling complex tasks necessitates analysing, and designing such tasks before they may be implemented. That is we also have to consider ways of working with problems that furthermore most often will be done in teams of developers.

Project based educational forms may be motivated by at least three reasons. First, they provide basis for practicing core knowledge in appropriate contexts. Second, when projects scale up in size and complexity, those are by necessity solved in teams of developers. Therefore, the process of developing the project requires ways of working in teams. Practicing this means maturing students to approach similar ways of working in industry. Third, projects normally involve activities or techniques originally unfamiliar to the developer. That will put further demands on the developer to meet those challenges. Meeting such challenges means taking responsibility for updating your own experiences that in turn means that you are prepared to meet the changing future and correspond to evolving techniques, and disciplines.

This contribution has put a focus on project based work. In Software Engineering education such ways of working may be inspired by well known Software Engineering process models, as well as by the innovative CDIO initiative. Software Engineering process models aims to guide software developer teams through a qualitative process, to qualitative results. CDIO aims to give valuable practice to engineering students for those to more appropriately fit into product development of engineering industry.

The contribution shows several similarities between Software Engineering process models, and the CDIO initiative. The conclusion is therefore that those approaches are more or less interchangeable, that is, using a Software Engineering process model for educational purposes could be seen as following the CDIO initiative.

A project based case study has also been provided and discussed. That case study is based on a course implemented with a Software Engineering process model, but could as well be seen as implemented through the CDIO initiative approach. Furthermore, the case study is shown to give support for putting core knowledge in a context, as well as provide practice in working processes, and provide for students to explore new techniques to solve the goal of the project.

ACKNOWLEDGEMENTS

Acknowledgement to colleagues, and students, at my home department for participating and contributing to project based courses. Special thanks to colleagues and students of the project based course of this contribution's case study. Also thanks to Fredrik Frisk for cooperation in the work of [5]. Furthermore, great thanks to the anonymous reviewers of the 7th International CDIO Conference 2011, for valuable comments on the abstract of this contribution.

REFERENCES

[1] Arduino micro processor at: http://www.arduino.cc/. Picture of micro processor from that link, picture of house from home department.

[2] Bass L., Clements P., and Kazman R., Software Architecture in Practice, Addison-Wesley, 1998.

[3] Bowen J. and Marton F., The University of Learning – Beyond Quality and Competence, Routledge, 1998.

[4] Einarson D., "Internet of Things in Education", STTC, China Zhejiang International Science and Technology Cooperation Conference – part Internet of Things (IOT) at Jiaxing, Hangzhou China, 2010, (No paper, only presentation).

[5] Einarson D. and Frisk F., "Projektet "Projektstyrt Projektbaserad undervisning"", A study on project based learning especially for international students, Kristianstad University, 2007, in Swedish.

[6] Egidius H., "Pedagogik för 2000-talet", Natur och Kultur, 2003, in Swedish.

[7] Ghezzi, Jazayeri, Mandrioli. Fundamentals of Software Engineering, Prentice Hall, 2002.

[8] Naur P. and Randell B. (Eds.), Software Engineering: Report of a conference sponsored by the NATO Science Committee, Garmisch, Germany, 7-11 Oct. 1968, Brussels, Scientific Affairs Division, NATO (1969) 231pp.

[9] News on project presentations of student's first semester, in Swedish, Kristianstadbladet: http://www.kristianstadsbladet.se/hassleholm/article917896/Ung-ljuddesign-i-kulturhuset.html Norra Skåne: http://www.nsk.se/article/20100114/HASSLEHOLM/701149726/1125/*/de-vill-designa-nasta-storsaljande-dataspel

[10] Rational Unified Process in Education, UPEDU, http://www.upedu.org/

[11] Sommerville I., Software Engineering, Addison Wesley, 2010.

[12] System structure, free pictures and photos from the internet.

[13] Worldwide CDIO Initiative: A Framework for the Education of Engineers, http://www.cdio.org/

[14] XP, on eXtreme Programming, http://xprogramming.com/index.php

Biographical Information

Daniel Einarson has a PhD in Computer Science and has several years of experience in teaching Computer Science and Software Engineering. Furthermore, the author has been experimenting with several different forms of software process models as models for educational forms. Moreover, with inspiration from the CDIO initiative the author will strive after developing the educational forms for Software Engineering even further.

Corresponding author

Dr. Daniel Einarson
Kristianstad University
Norra stationsgatan 8A
281 48 Hässleholm, Sweden
+46 -44 203177
daniel.einarson@hkr.se