# Good practices for Educational Software Engineering Projects

Louwarnoud van der Duim
*University of Groningen*
*l.a.van.der.duim@rug.nl*

Jesper Andersson
*Växjö University*
*Jesper.Andersson@vxu.se*

Marco Sinnema
*University of Groningen*
*mail@msinnema.nl*

## Abstract

*Recent publications indicate the importance of software engineering in the computer science curriculum. In this paper, we present the final part of software engineering education at University of Groningen in the Netherlands and Växjö University in Sweden, where student teams perform an industrial software development project. It furthermore presents the main educational problems encountered in such real-life projects and explains how this international course addresses these problems. The main contribution of this paper is a set of seven good practices for project based software engineering education.*

## 1. Introduction

For more than ten years, the computer science studies at University of Groningen in the Netherlands and at Växjö University in Sweden include courses where students learn software engineering in practice. Since two years, we have been offering a joint course called ISEP (International Software Engineering Project). In this project course, about fifty Dutch and Swedish students work together in international teams. Last year, each team contained about six students from each university. Each team performed a software project for a company, located in the Netherlands or in Sweden.

The primary goal of this course is to learn software engineering by resembling an industrial setting as closely as possible. To this extent, we analyzed the main aspects of software development and integrated these aspects in our ISEP course. These main aspects encompass working in teams, the size of the projects in terms of members and hours, distributed development, customers from industry, bidding on requests for information, focus on quality and delivery, as well as the combination of greenfield and maintenance projects.

Although introducing these real-life aspects into software engineering education does improve the learning experience of the students, it also introduces some educational problems. These problems mainly involve the amount of complexity combined with a lack of structure the students are facing, in particular compared to other computer science courses.

In this paper, we discuss problems with project-based teaching, based on previous publications and our course setup. We furthermore present educational measures that alleviate the identified problems and how we integrated these measures into ISEP. We conclude this paper with its main contribution, i.e., a set of good practices for educational software engineering projects. These practices have been tested and evaluated during sequential course instances.

## 2. Software Engineering Projects

We start our discussion on teaching software engineering by presenting the projected learning outcomes and the project setup in ISEP.

### 2.1 Learning outcomes

To prepare students well, they have to learn and gain experience from working in an environment that resembles an industrial situation as closely as possible. We derived the aspects directly from the student's future working environment, i.e., software development in industry. Such development requires many technical and social skills, e.g., working with the latest technology and cooperating in large teams.

**Cooperation in teams** – Industrial software projects are rarely based on individual effort. Instead, engineers work together in teams to achieve a specific goal. This requires students to learn to cooperate in teams, which implies dividing tasks and interacting with other team members. These social aspects make teamwork completely different from performing individual tasks.

**Project type** – In industry, there are two types of software engineering projects, i.e., greenfield development and maintenance projects. *Greenfield development* represents projects where engineers develop software from scratch, applying a full development cycle from requirements to testing and delivery. Students therefore have to learn this greenfield development, since it is here where all

software engineering techniques they learn in other courses are combined and applied.

While software engineering education is most frequently based on this greenfield development, in industry, software is only rarely developed from scratch. Instead, when the project starts, there is often already a software system in place and the goal of the project is to enhance and optimize the system, i.e., *software maintenance* [4]. Figures show that maintenance is the most expensive activity in the software life cycle. These, in combination with the fact that most courses focus on greenfield development, increase the importance of teaching these matters.

**Project Management** – A well-managed project successfully delivers its results within the given constraints and resources. Students have to learn the social and engineering aspects of project management.

**Distributed Environment** – In large software development projects, engineers are typically not situated together in one location. Instead, many projects are geographically distributed, over several locations within a country or even distributed internationally over multiple countries. Performing a project with distributed teams, prepares students for working in such development projects.

**Flexible to Technology** – In an industrial setting, the customer defines the technology requirements. Customers consider different options and select a technology that in many instances is not in line with the development team's preference. Students must experience elements of uncertainty in their education and learn how to cope with them. The uncertainty forces students to bring in experiences from other software engineering courses. They have to set up projects within their project that investigate the specific technologies and train the team.

**Software Quality** – Engineers develop software products for a market of demanding customers. We judge project success on the quality of the products they deliver. Quality cannot be achieved in an ad-hoc manner. On the contrary, it requires defined processes and lots of additional work. In our experience, the majority of students does not understand this very well. Therefore, it is important to teach good software quality practices.

## 2.2 ISEP Setup

The design and management of an international course with approximately 50 students requires detailed planning and careful organization. Over the years, the project setup has evolved, first at each university and later in cooperation. The ISEP projects last for almost six months. During these months, groups of students from the Netherlands and Sweden work together in a development project defined by an industrial partner. We depict the ISEP course organization in Figure 1 and describe the course setup and its organization in detail below.

**Project and project team** – Each ISEP project consists of two subprojects, one greenfield project and one maintenance project. Each subproject spans 10 weeks. The project teams consist of approximately 10 students (5 from each country). To further resemble a real project, we redistribute all students over the teams and customers when the first subproject is completed. The new teams have to finish and improve the product developed by the first team.

Although we give the teams a lot of freedom in arranging the structure within their project, we do require some management roles. Examples of those roles are a project manager, a requirements manager, and a quality manager. In addition to these roles, the students learn project management from the subcontracting aspect of ISEP. To that extent, the team has to isolate a part of their work to students from another software engineering course. The ISEP team manages these subcontractors, considering the dependencies with their own project as well as time constraints. Additional external resources are architectural consultants. They come in and evaluate the projects. Here the teams have to provide the consultants with adequate information on their projects. We use external customers in ISEP, instead of preparing and developing an internal assignment for the students. Dutch and Swedish companies define realistic projects targeted for project teams of 10 members. We do not give them any restrictions concerning technology or problem domain.

**Supervision and Evaluation -** In Figure 1, we depict the supervision and evaluation roles played by the faculty. Two Heads of Department (HoD) are assigned to each team, together with project supervisors. The HoDs are available for the students on a daily basis. They have a meeting with every individual student in their team at least 6 times during the project. PhD-students or Teaching Assistants performs the role of HoD. Teaching Assistants are selected on the basis of their performance in software engineering courses and projects with emphasis on their social skills. The supervisors meet students on a weekly basis. The evaluations performed by HoDs and supervisors focus on individual and team effort, and team interaction. To teach students the basics and importance of quality assurance and perform a continuous evaluation, ISEP contains a quality team. With respect to quality assurance in ISEP, SPICE (Software Process Improvement and Capability dEtermination) [11] plays a central role. The quality team performs a weekly SPICE assessment on all
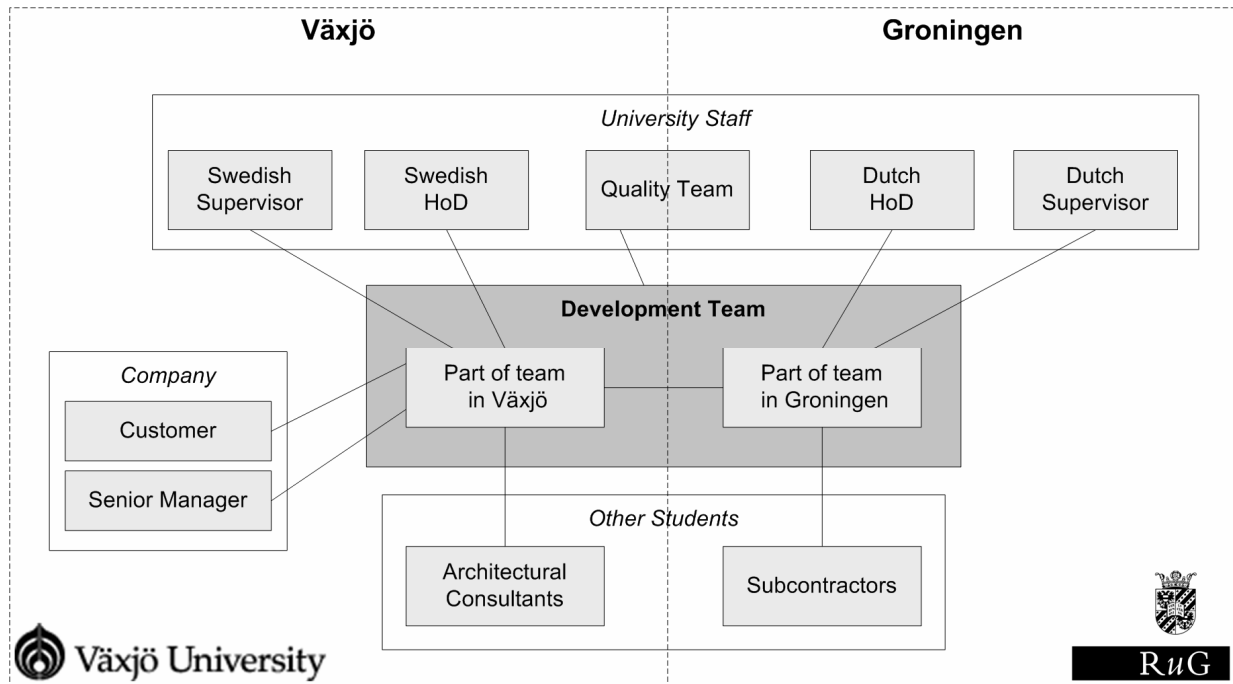
**Figure 1 Development Team Interaction – Internal and External**

development teams. Each weekly assessment focuses on three or four areas, e.g., Requirements, Testing or Deployment. In this way, we provide feedback on team performance with respect to software quality and in which ways it can improve.

## 3. Problems in educational software engineering projects

Several problems have been reported about educational software engineering projects. In this section, we select three problems that we think are the most important based on our own experience:

- Complexity problem
- Focus on technology problem
- Free rider problem

### 3.1 Complexity problem

The main problem reported in literature is that real life projects are too complex and demanding for undergraduate students [5]. Authors argue that it is hardly possible to set up a project that gives students the same structure they are used to from their other coursework and that this is necessary for "surviving" such a project. Connected to this problem is the fact that students have problems with relating software

engineering theory to the problems they encounter in real projects.

*Our experience with the complexity problem*

In ISEP (and the projects we did prior to ISEP), we also experienced the complexity problem. Our students have repeatedly reported problems in connecting the theory from different software engineering courses they take to the practical application of the theory in ISEP and they tend to be overwhelmed by the complexity of the project. Some students also reported another problem related to complexity which was the difficulty of selecting the right tooling to use in their projects.

### 3.2 Focus on technology problem

Another reported problem, is that the "Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering" and the Software Engineering Body Of Knowledge emphasize "the engineering in Software Engineering" [5], thereby squeezing out the social and human dimensions of software engineering. On top of this, most teachers involved with educational software engineering projects have a technical background and therefore are likely to rate technical problems as more important than social/human problems.

*Our experience with the focus on technology problem*

Our experience is indeed that students have problems with the social aspects involved in a software engineering project. We can see this for instance in the lack of traceability throughout the documents describing the relevant SE-disciplines and between product documents and the implementation. Most of the time the reason for the lack of traceability lies in insufficient communication between team members, for instance team members prepare design documents without looking at the requirements.

Sometimes, there is an uneven distribution of tasks throughout the student team. On top of that, several students have a rigid view on what their task is and what is not. Finally, customers frequently report problems in communication with the students. Students often have inappropriate ways of communicating with industrial customers and tend to forget that the customer has more to do then talking to the student team.

### 3.3 Free Riders problem

One of the problems commonly associated with team work is that of free riders. Those are students who do not really contribute to the team's effort, but try to pass the course anyhow. This problem typically occurs when students know that the teacher does not have a real insight in the individuals' efforts. The problem of free riding has its effects on the team as a whole because it tends to lead to lower group performances [9]. This problem is also reported for educational software engineering projects [1,10].

*Our experience with the free riders problem*

In the first years of working with educational SE-projects we had little insight in the individual achievements of students and during evaluations free riding was reported by students. Students report several reasons for free riding: lack of interest, the work load in the SE-projects as compared to other courses, and lack of social skills.

## 4. Educational Measures in ISEP for the reported problems

When teaching software engineering during the last ten years, we came across the problems we described in section 3 at both universities. We decided that we want to keep projects with the complexity that students will face in their future professional life. In this section, we discuss how to handle the complexity problem, the focus on technology problem and the free

rider problem. For these problems, we also describe what we have done in ISEP to address them.

### 4.1 Educational measures for the complexity problem

Translated to educational measures, the problems described in 3.1 underline the need for what is known as advance organizers and other measures to promote transfer of learning. Although the problem is real and attention should be given to reducing it, we also believe that communicating high expectations, which is one of the principles we will discuss in section 5, reduces this problem.

**Advance Organizers -** An advance organizer is information that is presented prior to learning and that can be used by the learner to organize and interpret new incoming information [2]. It can be used by students to connect new information to existing knowledge. Advance organizers also provide support for new information. If used properly, they "direct your attention to what is important in the coming material; they highlight relationships among ideas that will be presented; and remind you of relevant information you already have" [6]. In the software engineering field, there are a lot of standards (e.g., for software processes and quality processes) that can be used as advance organizers.

**Transfer -** Transfer of learning occurs when learning in one context enhances (positive transfer) or undermines (negative transfer) a related performance in another context [3]. The positive transfer that is needed is the transfer of applying theoretical knowledge from courses to solve practical problems in projects. The negative transfer one wants to avoid is (most of times) transfer from the students own software engineering practice (hacking away in hobby projects) and the application of such practices in software engineering projects.

*The ISEP approach to the complexity problem*

As described, the main solution for reducing the complexity problem lies in using advance organizers and increasing transfer of learning. In ISEP, we use four main advance organizers:

- UPEDU
- Request for Information
- SPICE/working with a Q-team
- The ISEP tooling platform

**UPEDU -** The first advance organizer is the development process we prescribe for the teams, the unified process for education (UPEDU) [7]. In essence, UPEDU, which is based on the Unified Process, gives

the student an introduction, concepts, workflows, activities, artifacts, and guidelines to all disciplines a software development organization uses. This is done within the framework of the phases a software development process usually has. We think that working with a prescribed development process is in line with our approach to present students with real life projects, because many companies also work with some standard development process. For students, this means that they do not have to select a process model. We think this is reasonable because the problems that the companies present to the students are in themselves complex enough.

All development in ISEP has the characteristic of geographical distribution and that quality control is emphasized. Therefore, we adapted the UPEDU by specifying that the students performing the role of an analyst and a project manager should be based in the country where the company they work for is based in and by adding the role of a quality manager. In some cases, we also specify that some named student needs to perform some specific role.

Students also tailor the UPEDU to the specific needs of their project. We developed educational material that supports students to make a project specific adaptation that best fits the specific properties of the team. In this material, we help the students to detect the forces in and around their project that call for process adaptation.

**Request for Information -** Every customer in ISEP writes a request for information (RFI) in which his problem is presented and where the customer poses question he likes the students to answer. All teams initiate their projects by writing a proposal for two requests for information. The RFI is a document with three purposes. As an advance organizer, it is a pre-structured document that guides the students to important activities and decisions of the inception phase of their project. What students must deliver here can be compared to the vision document of UPEDU. The second purpose is that it forces the team to investigate the problem and, based on that, derive a realistic set of requirements along with an initial project plan. Finally, it is a document that customers use to decide, which student team they would like to work with.

**SPICE/Working with a Q-team -** The third advance organizer is working with a "set of practices fundamental to good software engineering": SPICE. By selecting relevant base practices for the two parts of the projects (greenfield and maintenance), we focus the student attention, instead of letting them drown in the more than a hundred available base practices. The base practices of SPICE provide students with checklists for what to do for the relevant disciplines of UPEDU. We

believe that working with SPICE also increases the transfer of learning. To a large degree, the principles behind SPICE are identical to the principles in UPEDU. This point is emphasized in our SPICE lectures to help the students to understand that the same principles that are useful in a development process can also be useful in a quality improvement process.

We try to get the students to work on level 3 of SPICE. Working on level 1 means that they have to perform the base practices we have selected for them. Working on level 2 means that they have to manage (plan and track) the base practices. Working on level 3 means that they have to tailor the standard process (UPEDU) for the selected base practices.

**The ISEP-tooling platform -** For each project, this platform encompasses a file repository, a bug tracking system, a forum, a WIKI, and a calendar. The versioned file repository allows the students to start immediately with a robust system for source code and document storage. Similarly, the bug tracking system can be used from the beginning of their project. The forum, the WIKI and course calendar facilitate and stimulate team communication, communication between different teams, and between teams and their supervisors. Because all the teams must use the tooling platform for all their artifacts, it also functions as a reporting mechanism from the team to the HoDs and supervisors. Apart from these tools, we also provide means for videoconferences to the students. As an advance organizer, these tools emphasize that a project is about communication and cooperation.

## 4.2 Educational measures for the focus on technology problem

Most computer science students have little intrinsic motivation for the social aspects of software engineering. That means that, in order to focus their attention on these aspects, one will have to raise their extrinsic motivation. One way of doing this is by emphasizing these aspects in their grading. This should be told to students right from the start and during the turn of the course they should be given prompt feedback on their achievements in this aspect. This implicates that one must involve teaching staff on a daily basis.

Involving teachers with a non-technical background can also reduce this problem. In the same way, introducing students with less technical background can also be of use as van Vliet points out [5].

*The ISEP approach to the focus on technology problem*

As described, the most promising solution for reducing the focus on technology problem lies in stressing the social aspects of the project for the students and involving staff with a non-technical background in the project. In ISEP, we have 5 measures to do this:

- Grading
- Request for Information
- Heads of Department
- Customers/senior managers
- Need based lectures

**Grading -** In the general introduction document we give to students when they start the project, we describe the grading criteria. Some of the things we stress are that teamwork, relationship with the customer (who sends in his own evaluation about working with the team) and cooperation within the team will be an integral part of the evaluation for the individual grade.

**Request for Information -** The request for information as described above is not about the technique but about introducing your team and its qualifications to the customer. Presenting an attractive solution to the customer for the problem described in the RFI is also part of this procedure.

**Head of Departments (HoDs) -** The HoDs are primarily selected on their social and not on their technical skills. This reduces the focus on technology problem. By reading the teams contributions' on the tooling platform, the HoD is up-to-date on the team's performance on a daily basis and is able to provide project specific advice on software engineering issues. In this respect, a staff person whose main focus is not the technology used continuously evaluates students.

**Customers/senior Managers -** People from the companies we work with are not technicians. Usually they are project managers in their companies. In our customer preparation, we tell them not to focus too much on the technology.

In addition to functioning as a customer of an ISEP team, companies may also act as a senior manager to another team. The senior manager meets the project manager of the ISEP team on a weekly basis and uses industrial expertise as project manager mentor. The focus in this coaching of project managers is also on the social aspects.

**Need based lectures -** Where we identify general problems and difficulties in the teams, we organize lectures for all the students. These lectures frequently have topics about the social aspects of software development projects and project management.

## 4.3 Educational measures for free riders problem

In general, this problem should be handled by introducing measures that reduce the anonymity in the project. Students should know that their individual achievements are taken into consideration. There is research that points in the direction that free riding increases if the person who is supervising the team takes on an expert role and that free riding decreases if the supervisor performs the role of a process guard [9].

*The ISEP approach to the free riders problem*

As described, the most promising solution for reducing the free rider problem lies in reducing the anonymity for students and in letting the staff operate as process guards instead of experts. In ISEP, we have 3 measures to do this

- Time logging
- Continuous evaluation
- Working with pre-defined processes

**Time logging -** Each team member keeps track of all hours spent and the tasks performed within these hours. They have to keep the time log on the basis of the activities that are described in UPEDU for the roles they are performing. Every week they have to submit their time log to their project manager and to their HoD. The team and HoD check whether they approve with the reported hours or not.

**Continuous evaluation -** HoDs meet with every individual project member for an evaluation meeting during the project. Students meet with the Q-team once a week or more frequently if required.

The tooling platform is not only meant for students but also for the staff to see what the discussions are about and to see if there are students who are not partaking in the discussions.

**Working with a pre-defined process -** As said, we use pre-defined processes in ISEP. All involved staff members emphasize the need to work according to these processes. They do not provide clear-cut answers to any problem but they do remind the student that, for instance, that it is part of the process to let the customer prioritize the requirements.

## 5. Good practices for Educational Software Engineering Projects

In section 4, we demonstrated that it is possible to address the problems described in section 3. In this section, we introduce seven principles for good practices in undergraduate education that can be applied to educational software engineering projects. After introducing the good practices, we describe how

we worked with them to turn them into good practices for educational software engineering projects. For every principle we give:

- A short description what the principle is about
- Our implementation of the principle
- Observations from our practice for this principle

In 1987, Chickering and Gamson published "Seven Principles for Good Practice in Undergraduate Education" [8]. They describe seven principles in education with sufficient empirical evidence that they have a positive effect on learning outcomes of students:

1. Good Practice Encourages Contacts Between Students and Faculty
2. Good Practice Develops Reciprocity and Cooperation Among Students
3. Good Practice Uses Active Learning Techniques
4. Good Practice Gives Prompt Feedback
5. Good Practice Emphasizes Time on Task
6. Good Practice Communicates High Expectations
7. Good Practice Respects Diverse Talents and Ways of Learning

In ISEP, we actively use these principles to structure the projects.

## 5.1 Encourage Contacts between Students and Faculty

*Description*

Let the project setup encourage and reward frequent contacts between students and staff. Provide formal and informal communication channels on different levels.

*Our implementation of the principle*

- The Heads of Department have contact with the teams on a daily basis.
- There are weekly Q-team meetings where students discuss relevant versions of documents and/or implementations with the faculty.
- The tooling platform is not only for the student teams but also for the faculty who for instance partake in forum discussions.

*Observations from our practice for this principle*

Our experience is that the HoD's role as an observer and coach promotes the interaction with the faculty. The HoD frequently forwards issues and problems the team faces to the senior staff. We deal with some of the

issues and problems through lectures. The continuous evaluation requires that senior staff together with the HoDs take a more active role earlier in the course as opposed to a more traditional evaluation at the end of a course. This increases the staff involvement, which in turn encourages students during the course of their project.

The Q-team meetings, where students and faculty work together to improve documents and implementation, are highly valued by students. The students learn a lot from faculty who do little else but asking questions.

The tooling platform supports asynchronous communication, which increase the availability of both staff and students. Students and staff may post questions and respond to questions at any time. A positive side effect is that this reduces the stress. Teachers and students have other commitments that are equally important as the project. This is a big difference compared to a real-world project. The asynchronous communication in forums and the WIKI mitigates these problems.

## 5.2 Develop Reciprocity and Cooperation among Students

*Description*

The project setup must encourage and reward student cooperation. This must be done in such a way that students respect individual differences.

*Our implementation of the principle*

- The problems customers present to students can only be solved by teamwork. Together with the customers, we try to set them up in such a way that it is possible to expand or in some other way adjust the problems during the project. This is important since we must have the option to do this, for instance to react to student teams that progress more rapidly then anticipated.
- The UPEDU activities prescribe cooperation between specific roles in the project.
- The tooling platform gives students several means for communication and cooperation. The tooling platform in itself signals that the project is about communication and cooperation.

*Observations from our practice for this principle*

Because the project setup is distributed, students are forced to think about the best way to cooperate in a team where team members are not at the same location.

The tooling platform is used extensively. In the last year the teams produced 802 topics and 5201 posts in the discussion forum, an average of 100 posts per student.

In a real-world project, there is always some element of uncertainty. As a manager or developer, one must stay alert and be able to quickly react to changed project conditions. An unstable and changing environment is much more challenging for students and leads to increased cooperation

## 5.3 Use Active Learning Techniques

*Description*

Students themselves must take active measures to lookup, interpret, and apply new knowledge in their project.

*Our implementation of the principle*

- Working with a RFI in the first week of the project forces students to start cooperating in the team and to start negotiating with the customer. The questions in the RFI force students to actively research the problem and to think about the cost to solve it.
- The SPICE base practices that we have selected for the students specify all kinds of activities that students have to perform. An example is ENG 2.4 Evaluate requirements with customer.
- The tooling platform is set up in such a way that they can access all their materials and communicate with each other anytime and anywhere.
- Students do not need to reproduce knowledge in ISEP but they have to actively construct new software using their software engineering knowledge.
- Students are furthermore encouraged to actively propose topics for the "need-based" lectures.

*Observations from our practice for this principle*

Letting the student teams compete with each other at the start to get a specific project has the effect of a jump-start. Students are active in the project from the very first day.

Because of the advance organizers UPEDU and SPICE, students are seldom in doubt if work remains. If students think they are finished, there is always the Q-team to remind them that you can also improve and perform your work on a higher quality level.

The "Need based lectures" cover topics students are currently working with. This means that they are usually more interested in the lecture than in other lectures, where the content has no immediate practical application. Still, the number of students actively proposing lecture topics is low.

## 5.4 Give Prompt Feedback

*Description*

Develop tight feedback loops on different levels within the project.

*Our implementation of the principle*

- Reviewing is an important activity in UPEDU. It is part of all activities in all disciplines. This is emphasized at Q-team meetings and students are asked to show the review results.
- The quality management approach forces students to continuously evaluate their work.
- The tooling platform provides them with the means for immediate response.
- The Head of Department fills an important role, giving feedback in weekly team meetings and contacting individual students frequently. On another level, the evaluation mechanisms, e.g., quality assurance and senior advisors, provide more in-depth but less frequent feedback. We provide feedback in different formats, for instance in-meeting feedback and formal-evaluation forms.

*Observations from our practice for this principle*

The first reviews that students perform are not much more then an advanced spell check. After feedback in the Q-team meetings (and by showing them more interesting reviews performed by the Q-team members), the quality of the reviews usually improves.

The advance organizers we see that support this practice best are the different tools we provide for students. A forum and WIKI can be used for less formal feedback, for instance by responding to suggestions made in forum discussions. The file repository is better suited for the formal evaluations.

Students are perfectly capable to give each other adequate feedback, especially, when there is also frequent feedback from the staff. This point is important to prevent student feedback that is not based on a good software engineering insight.

The better students get extra motivation from taking part in a project where there is a constant flow of feedback that enables them to improve their skills.

## 5.5 Emphasize Time on Task

*Description*

The project setup, the evaluation criteria, and advanced organizers shall emphasize time on task, and encourage individual and group reflections based on effort and result.

*Our implementation of the principle*
- The RFI procedure ensures that students get to work on the project immediately. After this initial jump-start, the daily availability of the HoDs as well as the weekly meetings with the customer and Q-team make certain that the students keep on working.
- We make sure that students take the planning activities from the project management discipline of UPEDU very serious. The planning must be detailed enough for all students in the team to know on which tasks to spend their time in the upcoming weekly iteration.
- All students have to keep a time log in which they specify how they have used their time in the past week.

*Observations from our practice for this principle*

The project is a 280 hours course for students. 70% of the students spend more or less that amount of time. About 10% of the students spend less time on the project, which are usually students that fail the project. The other students spend significantly more time on the project.

Students often look for the easy way out but, with our mechanism that compares performance with quality in place, such situations are identified and dealt with accordingly. The students take the planning activities very serious. Several teams use the functionality of the bug tracking tool to distribute, track and close detailed tasks on an individual level. The time log is effective to keep track of student activity throughout the project and it is easy to identify underperforming students.

Although students tend to complain about the workload during the project, in the final evaluation, most students think that they have learned a lot. So it is their own experience that by spending a lot of time on a subject they also learn a lot.

## 5.6 Communicate High Expectations

*Description*

During the course of a project, the project setup and faculty communicate high expectations to team members.

*Our implementation of the principle*

High expectations are a good way of positive coaching. Positive coaching aims at improving learning outcomes by high expectations, delivering the message "you are a good student, you can do this". From the students' perspective, it is important to know that the faculty has confidence in their capabilities and commitment. Still, it is important with realistic expectations, otherwise the effect will be negative. We show high expectations in several ways:
- By letting them work on realistic projects for real industrial customers and starting with a competitive bidding procedure (RFI).
- By letting them work in a distributed environment.
- By aiming at reaching level 3 of SPICE and by the quality standards we set throughout the project.

*Observations from our practice for this principle*

The introduction of RFIs at the start of the course sometimes leads to a competition. If a project looks more attractive and fun, two groups end up in a race-for-the-contract. They will push their promises to the customer as far as possible to get the project. This is an indirect way of setting high expectations. The team commits to the customer and project by signing a contract.

Students are able to work in a distributed environment without any real problems. We send out the message to them that they should be able to do this and the effect usually is that they are.

Students feel challenged by the goal of reaching level 3 of SPICE. Some student teams actively start to monitor on which level they perform on the base practices of SPICE.

## 5.7 Respect Diverse Talents and Ways of Learning

*Description*

The project setup and team organization shall support diversity in team member skills and learning preferences.

*Our implementation of the principle*

- UPEDU specifies 10 different roles within the development team. For every role, there are different activities. Students can choose the role that suite them best. On top of the 10 roles in UPEDU, we added the role of Quality manager, which is a combination of activities from different UPEDU roles.
- The tooling platform is set up in such a way that the students have a lot of flexibility in how to use the tools. This leads to very different ways of using, for instance, a WIKI or bug tracking system.

*Observations from our practice for this principle*

Students' previous experiences and talents are a double-edged sword. On the positive side, some students are better suited for certain roles in the project. On the negative side, this can result in sub-optimal learning outcomes. Students should learn and improve. Thus, it is sometimes necessary to, independently of previous experience and skill levels, assign "less qualified" students to specific roles.

Students report that they think it is motivating to be able to choose their roles in the project. For some roles (e.g., the project manager), there is always some competition in the teams.

## 6. Conclusion

In this paper, we described the educational software engineering project we teach at University of Groningen in the Netherlands and at Växjö University in Sweden. The primary goal of this educational project is to learn software engineering by resembling an industrial setting as closely as possible. The main industrial aspects of this education encompass working in teams, the size of the projects in terms of members and hours, distributed development, customers from industry, bidding on requests for information, focus on quality and delivery, as well as the combination of greenfield and maintenance projects.

We explained that, although introducing these real-life aspects into software engineering education does improve the learning experience of the students, it also introduces some educational problems. This paper therefore discusses the problems, as reported in other relevant publications, connected to such software engineering education, and summarizes measures we applied in our software engineering project to alleviate these problems. The main contribution of this paper is a set of seven best practices for software engineering education, derived from seven best practices for general education [8], i.e., (1) encourage contacts between students and faculty, (2) develop reciprocity and cooperation among students, (3) use active learning techniques, (4) gives prompt feedback, (5) emphasize time on task, (6) communicate high expectations, and (7) respect diverse talents and ways of learning. We finally show how we implemented these seven best practices and present our observations from the last two years. Our experience is that realistic software engineering projects are a good candidate for teaching many aspects of software engineering, which are difficult to address in regular ex-cathedra teaching. Project based learning is extremely time-consuming, for students as well as faculty. Still, the results of our evaluations are encouraging. The positive results make the extra effort required worthwhile.

## 7. References

[1] Hazzan, O. and Dubinsky, Y. "Teaching a software development methodology: the case of extreme programming," In *Proceedings of 16th Conference on Software Engineering Education and Training, 2003. (CSEE&T 2003),* pp. 176- 184, March 2003

[2] Mayer, R. "Learning and Instruction". New Jersey: Pearson Education, Inc. 2003

[3] Perkins. David N. and Salomon, G. "Transfer of Learning", Contribution to the International Encyclopedia of Education, 2nd Ed. Oxford, England: Pergamon Press, September 2, 1992

[4] Collofello, J.S. "Teaching practical software maintenance skills in a software engineering course", In *Proceedings of the Twentieth SIGCSE Technical Symposium on Computer Science Education*, 1989, Barrett R.A. and Mansfield, M.J. Eds. SIGCSE '89. ACM Press

[5] Vliet, H. van, "Reflections on Software Engineering Education", IEEE Software, Vol 23, no 3 (2006), pp 55-61.

[6] Woolfolk, A. "Educational Psychology", 8th Ed. Boston: Allyn and Bacon, 2001.

[7] Robillard, P. and Kruchten, P. "Software Engineering Process with the UPEDU", Addison-Wesley, 2003.

[8] Chickering, A.W. and Gamson, Z.F. "Principles for Good Practice in Undergraduate Education", The Wingspread Journal, June, 1987

[9] Ruël, G.Ch, Bastiaans, N. and Nauta, N. "Free-riding and team performance in project education," Research Report 03A42, University of Groningen, Research Institute SOM (Systems, Organisations and Management), 2003.

[10] Smarkusky, D., Dempsey, R., Ludka, J., and de Quillettes, F. "Enhancing team knowledge: instruction vs. experience", *SIGCSE Bull.* 37, 1 (Feb. 2005), pp. 460-464.

[11] Dorling, A. "SPICE – "Software process improvement and capability determination", Software Quality Journal, Dec 1993. Vol. 2, no. 4, pp. 209-224