ROLES TRANSFORMATION WITHIN A SOFTWARE ENGINEERING MASTER BY IMMERSION

Vincent Ribaud, Philippe Saliou

ROLES TRANSFORMATION WITHIN A SOFTWARE ENGINEERING MASTER BY IMMERSION

V.Ribaud and P. Saliou

EA2215, Computer Science Departement, Faculté des sciences, 29285 Brest Cedex, France

E-mail:{Vincent.Ribaud, Philippe.Saliou}@univ-brest.fr

Abstract

Learning the software engineering (SE) profession is a difficult task. Most SE professionals will say that they learned it "by doing". Hence, the main paradigm used is teaching software engineering by doing. Most academic curricula address this issue through projects, but academic projects are not sufficient to achieve the goal. Beyond the SE learning, more general questions arise : how can we teach/learn engineering ? What is a "long-term education", particularly in computer science ? And in a general sense ?

Since September 2002, Brest university has offered a dedicated second year of Master in computer science, called "Software engineering by immersion". The main objectives are : mastering software engineering activities and skills, working in a team, coping with change. Except for English and Communication, no courses are offered. The plan of action is built on a 6-month team project, lead and tutored by an experienced software professional. A real corporate baseline (by the courtesy of a software services company) has been tailored to the apprenticeship paradigm and now sustains engineering activities and products delivery. The apprenticeship process is achieved in two iterations. During the first iteration (4 months), students are swapped around the different tasks needed by engineering activities and strongly guided by the tutor. During the second iteration (2 months), roles are fixed within each team and teams are relatively autonomous when completing the project, the tutor performing mainly a supervising and rescuing activity. The assessment process is essentially formative, due to the permanent feedback of tutoring.

After a very positive first year, the outcome of the second year is rather disappointing. This year, we met with unexpected difficulties and this may be due to the roles transformation needed in the immersion paradigm, for the students as well as for the tutors. This paper attempts to present the new roles. Teachers have to deal with several kinds of tutoring activities : coordination between teams, software project management, individual and collective apprenticeship during the first iteration; peer review, sustaining the student acting as software manager, programme management during the second iteration. Students have to learn new skills but above all play new roles namely : citizen, builder, explorer, team member.

Assessing the role playing process formally is very difficult, first in a purely quantitative manner and second in such a short time. Moreover, we are aware of very few similar experiments. So, this paper tries to evaluate the new roles within the constructivism paradigm, especially the apprenticeship paradigm described by Jacques Tardif. The teacher and student roles defined in this paradigm are compared with those defined in our immersion paradigm. Some problems are reported and possible improvements are drafted.

As a conclusion, if the immersion system is exciting and innovative, on the other hand the system is fragile if both teachers and students are unable to play the new roles required. In addition to these first conclusions, the professional insertion and career evolution of students need to be observed over several years in order to evaluate the real benefits of this system.

1 Introduction

Among possible definitions of software engineering, we kept the following : "Software engineering includes the different types of knowledge, processes, scientific and technical experiences used for the manufacturing of software. This starts with the software order from the contractor and ends with its exploitation".

Teaching software engineering is hard, but learning software engineering is much harder. Lethbridge and al. [8] surveyed software professionals to learn which educational topics have proved most important to them in their careers and to identify the topics for which their education or current knowledge could be improved. The survey indicates that employees are likely to lack skills and knowledge in fundamentals areas of software engineering such as requirements gathering/analysis, human/computer interfaces, project management, configuration management and in people skills (attitudes) such as leadership and negotiation. Most SE professionals will say that they learned it "by doing". Hence, the main paradigm used is teaching software engineering by doing. Moreover, several analyses of software engineering teaching emphasise the benefit of a long-term team project (one semester or a year) [1] , [9], [11]. Beyond the SE learning, more general questions arise : how can we teach/learn engineering ? What is a "long-term education", particularly in computer science ? And in a general sense ?

Brest University has been providing since September 2002 a second year of Master in software engineering. The opportunity to offer a dedicated year allows us to cope with some missing skills which are however well-identified by the professional branch. The main objective is mastering software engineering activities and skills. Moreover, among the characteristics and practices recommended for computer science graduates (Computing Curricula 2001, Computer Science Volume), we kept as additional objectives : working in a team, coping with change, and being able to appreciate a perspective in a whole [2]. The pedagogy used to achieve these objectives breaks with the usual teaching paradigm to rely on an learning paradigm [14].

The main idea of the system is to let professional realities into our university walls. Students work in teams to analyse, design, implement, test and document a software project relying on strong software engineering principles. The pedagogical system imitates as closely as possible real-world phenomena: a professional working environment, the client-supplier relationship, the application of a development baseline, the use of methods and associated tools, the cooperation within the team, ... We call this education system « Software engineering apprenticeship by immersion ». This system is presented in sections 2 and 3, a digest of [12].

During the first year of our Master in computer science, students experimented with the "learning by doing" paradigm through two mini-projects (2 weeks each) in distributed systems and in information systems. Each mini-project follows a classical development cycle starting with software specifications as defined by the teachers and ending with deliveries and a demonstration. But in the immersion paradigm, the whole year is devoted to the project. No courses are offered, except for English and Communication.

The need for significant changes in the character of engineering education emerged in the mid-1980s. As an example, from 1989, the National Science Foundation supported six major Coalitions of U. S. institutions to pursue the vision and goals outlined in their proposals [13]. The Université de Sherbrooke has totally redesigned its electrical engineering and computer engineering programs, following an original learning approach that combines problem-based and project-based learning [7]. However, we are not aware of similar experiments in a MS in software engineering. However, the good and bad results that we observed probably apply to various disciplines using engineering.

Yet, providing professional working environment and processes is much easier and cheaper in software engineering education than in other fields.

After a very positive first year, the outcome of the second year is rather disappointing. This year, we met with unexpected difficulties and this may be due to the roles transformation needed in the apprenticeship by immersion paradigm, for the students as well as for the tutors. New roles are depicted in section 4.

We were not able to assess the role playing process formally, firstly because the assessment is more qualitative than quantitative and secondly because it is too early to decide if the seeds we sowed are producing (or will produce) the desired fruits. So, in section 5, we try to evaluate the new roles within the constructivism paradigm, especially the apprenticeship paradigm described by Jacques Tardif. The teacher and student roles defined in this paradigm are compared with those defined in our immersion system. Some problems are reported and possible improvements are drafted.

We finish the paper with related work and a conclusion.

2 General description

2.1 Immersion principles

We structured SE activities around three main areas : software project management activities, specific software development activities and software development support activities. This division is the reference framework for the university (in a diploma-awarding perspective), for the pedagogical team and the students, and finally for the professional branch which will hire these young graduates.

Teaching these various activities is quite difficult following the teaching paradigm. Hence the idea to tackle these activities within an immersion in a real project, with the following specificities:

- the project goes on in the university in a dedicated fitted-out room;
- the ten students are divided into two separate project teams, that we call companies;
- companies will have to produce the same product but with different methods, tools and technologies;
- the project manager driving the project is a university lecturer, formerly a SE professional;
- the target, the software product, is only a pretext for learning software engineering activities.

In order to emulate a standard firm environment, we fitted out a laboratory room with a landscape room for each company (with own individual working post), a common meeting room and an Internet room.

The apprenticeship environment is made of at least four essential elements:

- a set of activities (or tasks) related to the software engineering profession,
- a software development process which organises the set of SE activities,
- a corporate baseline (by courtesy of Thales-IS) which defines good practices and capitalizes the company’s know-how,
- a working framework including common installations and tool suites (IBM, Oracle, Rational).

The whole apprenticeship plan of action is guided by the development process, which defines, among others, the role and the schedule of project stages. It reveals clearly three kinds of teaching activities:

- *Organising*. It is a matter of scheduling and elaborating work cards that define, at each stage, the work to be done and preparing pedagogical supplies (book, software, hardware, corporate baseline…) needed to carry out the work.
- *Tutoring*. For each work card, a tutor is available to students who are thus provided with continuous support and assistance.
- *Control*. Work cards constitute the assessment framework because they define form and content for the deliverables to be produced and delivered.

2.2 Apprenticeship and manufacturing card

First, the apprenticeship process strongly relies on apprenticeship cards; then students enrich them for their own software development (manufacturing) process. The card structure is standardized (see as example below). Its main elements are the activity (here requirements capture) tied up to the work; the role to play (here analyst) with students' name; the work description (here a consolidation task); the products (deliverables) to deliver (here a Software Requirements Specification, SRS); the supplied pedagogical resources (here a writing guide and real SRS samples); workload and lead-time information.

Number : 8	**Date** : 11-21-2002	**Origin** : Ph. SALIOU	**Action**	
Program : SI2EWP1	**APPRENTICESHIP CARD**		Analyst	**L. Delemazure** **S. Le Livec**
Activity : Requirements capture		**Name** : Requirements consolidation		

WORK DESCRIPTION

This work aims to consolidate in a single document the set of requirements. The purposes of this work are :

To be in accordance with the TEMPO-ILI baseline and the Two-Track Unified Process.
To bring together two requirements : functional and technical.

…

The expected result will be materialized with a Software Requirements Specification document (SRS). This specification is a reference document for the software design.

…

Pedagogical resources can be helpful in order to write the SRS :

Simplified writing guide for the software requirements specification (TEMPO-IGQ347).
SRS Examples

…

Products	**Version**	**Milestone**
Software Requirements Specification (SRS)	A	11-29-2002

WORK IN PROGRESS

Estimation				**Reality**				
Start date	**End date**	**Workload**		**Start date**	**End date**	**Used workload**		
11-25-2002	12-03-2002	3,5	3,5					
Date	**Used workload**	**Deliveries-Observations**						

Figure 1 : Apprenticeship card

3 The apprenticeship process

The apprenticeship process is guided by an unified software development process :

- It materializes as a sequence of stages with clear and controlled objectives at each stage.
- Each stage includes some work described with work cards that define precisely the work to be done and the apprenticeships to gain and/or the competencies to mobilize.
- Each work card is assigned to one or several students who should take up the job (or the role) inherent to the activity.
- Students are swapped around the roles from one stage to another stage.

For each work card, students can rely on tutor's support and assistance on a daily basis.

3.1 First iteration

The *first tutored apprenticeship iteration* (4 months) lets students acquire knowledge and skills needed for each stage. An incomplete version of the software is built, entirely driven and tutored by the company's tutor. All SE activities are put in practice within a complete software development cycle.

Each apprenticeship card give rise to one or several deliverables. Each deliverable is carefully examined and annotated by tutors, then the tutor feeds back comments to the authors together with improvements to bring about. This assessment and feedback process is iterated (at least twice) until that the deliverable is considered as good enough for its future exploitation (it should arise some problems when the final delivery is not judged as good enough).

Software project management	**Coeff.**	**Apprenticeship card**
Software project leading	2	Project set-up Progress report meeting accounting
Software quality programme	1	Quality programme set-up Quality insurance
Software configuration management	2	Software configuration plan elaboration Software configuration management Technical incident management Version management
Software development engineering	**Coeff.**	**Apprenticeship card**
Requirements capture	2	Functional requirements capture Technical requirements capture Requirements consolidation
Technical architecture	2	Technical architecture validation Development framework exploration Generic design Technical prototype
Analysis	2	Requirements analysis Human-computer interface mock-up
Design	2	General design Relational database consolidation Detailed design
Coding – Unit testing	2	Coding – Unit testing
Integration-Qualification	2	Software test plan elaboration

		Software test dossier elaboration Integration Internal validation Qualification
Software development support	**Coeff.**	**Apprenticeship card**
Technical support	2	Means set-up System and networking support Database administration Development technical support
Tools and methods support	2	Analysis environment definition and set-up Design tailoring Development support Software configuration tool mastering
User's documentation	2	Software user's guide writing Software installation and exploitation guide writing
Installation - Deployment	1	Target installation and configuration Software deployment

The array above describes the link of apprenticeship cards (the learning activities) with the SE activities breakdown structure (the competencies curriculum). This set of apprenticeship cards is the reference framework for all the stakeholders (university, teachers, students, professional branch). Each apprenticeship card has an assessment and a mark is given to.

3.2 Second iteration

The *second accompanied application iteration* (2 months) is intended to transform knowledge and skills into competencies. Each company is relatively autonomous in order to complete the project. The expected software product at this end of this iteration corresponds to the whole software product as defined in the contract. The progress of this iteration is similar to those of the first iteration, except that the company's tutor stands back in order to allow the company to stand on one's own two feet. The company's tutor has mainly a supervising and rescuing function.

At the end of the first iteration, students carried out (or saw their team mate at work) different engineering activities, and students used deliverables produced by other students during previous stages. We consider that they have a first "apprenticeship by doing" of the job and also that they know what they are able or not able to perform.

A fixed organisation of the team is set up for the second iteration, structured around roles. One student is acting as project manager while the others are carrying out all the others activities. During the second iteration, students will rely on the apprenticeship process acquired during the first iteration. The company has to transform this process in a production process.

For the second iteration, we did not want to measure individual competencies and performances. We gave three marks for this iteration : a mark given the project itself (and the work done), a mark given a collective viva voce examination, a mark given individual reports on the personal and the work done collectively. The mark given on the project relies on an formative and summative assessment of the essential deliverables of a software project.

Assessment	Coeff.	Deliverables
Viva voce examination	2	
Individual written report	2	
Work	4	Software Requirements Specification Software Analysis Document Software Design Document Software Test Plan Software Test Description Software User's Guide Software Installation and Exploitation Guide Delivered information system (the software product)

4 Roles

4.1 Teacher roles

In order to guarantee the good progress of the training course, preparation work and continuous maintenance is necessary. This work is accomplished by a "*Coordinator*" working all the year long. He/she has to:

- define the new project in which the students will be immersed. He/she has to write requirements, then a response to solicitation including a technical and commercial offer answering the expected needs.
- prepare the logistics necessary for the smooth running of the service: to make an inventory of company's resources and to sort them out, to reset development framework to zero, to forecast needs in hardware of software, to negotiate contracts with software providers, to supply with and receipt new devices, software, books, self-training media, ...
- consolidate and to tailor the apprenticeship repository in order to make it easier to use by the tutors and the students. It could be the integration of new books or self-training supports as well as enriching or updating the TEMPO-ILI repository, for example apprenticeship cards.

The smooth running of the first iteration relies mostly on the company's tutor, who has two functions.

The "*Software manager-Tutor*" is the authentic software manager inside the company, he/she leads the students' team, is in charge of the work progress, and refers to the "Coordinator" as much as necessary. The "Software manager-Tutor" coordinates students' work and adjusts the planning according to real work progress. On a daily basis (at least one hour per day), he/she is the privileged interlocutor for the students as well as other apprenticeship system staff members. He/she prepares and drives the weekly progress report meeting, which is a privileged time to exchange information within the team. Last but not least, the main objective in the first iteration is the apprenticeship : the "Software manager-Tutor" is, first of all, a teacher who wishes to transfer competences to his/her team mate.

The "*Apprenticeship-Tutor*" coordinates and regulates individual and collective students' apprenticeships, assists students on a daily basis, and finally assesses deliverables. For these purposes, he/she relies on the apprenticeship process (see section 3.1) and on apprenticeship cards

(see section 2.2). Thanks to this support and assistance, students are able to provide deliverables. The production process is continuously sustained, the "Apprenticeship-Tutor" annotates, corrects, make proposals, reorientates. Generally, two assessment iterations are necessary in order to guarantee a satisfactory result, in all case sufficient to carry on the project.

The tasks and the workload of the company's tutor are still important in the second iteration; he/she has now three functions.

The "*Regulator-Tutor*" can accompany individually each student. TEMPO defines a regulator as: "The regulator is an expert outside the project whose experience corresponds to the current project in order to give impartial advice on functional and technical features. He/she is working as a peer who can help the software manager or the team mate." So, the "Regulator-Tutor" has mainly a consolidation activity working on documents and deliverables provided at each Manufacturing Card, but it is also used as an assessment activity.

The "*Facilitator-Tutor*" is in charge of the project logistics, but mainly sustains the student acting as the important role of project manager. It is a daily meeting (from 10 minutes from 1 hour) browsing current aspects of the projects.

The "*Programme manager-Tutor*" arbitrates conflicts, regulates workload, indeed may give orders in case of production locking. As seen in the definition above, the "Regulator-Tutor" has no decision power on the team. The role play needs a dedicated hierarchical role in order to solve the problems which could arise. In the TEMPO repository, the immediate superior of the project manager is the programme manager. "The programme manager acts on behalf of the unit (the firm) and the customer to ensure contractual commitments (cost, lead-times and performance) are met. [15]". This hierarchical authority is entrusted to the company's tutor and allow him/her to ensure commitments but also to substitute for the project manager for any technical, functional, organisational aspects.

4.2 Student roles

The first role is called "*a citizen*". Each student lives with their team mates during 6 months, 5 days a week, 8 hours a day. The respect of others, of their work, of their own working place and of the common infrastructure is fundamental when working in a team. Unlike what happened during previous years, the student must realize that the respect for the environment and a few rules (working hours, computer security, deadlines) is not enforced by the institution but accepted by each student. A few students were not able to impose this discipline on themselves, but "it is discipline first which transforms animality in humanity" (I. Kant, [6]).

The main role is called "*a builder*". The student is in turn architect, project manager, manufacturer, inventor, artist. Sometimes he/she will be providing others with products or services, sometimes he/she will be using the other students' work. He/she must understand that his/her own work takes place within a structured set and that the success of each piece is necessary to the success of the whole project itself. The understanding of long term issues should always be kept in mind and regular and sustained efforts are essential. It breaks away from previous attitudes where efforts are mainly motivated by the examination deadlines.

Beside this building activity, the apprentice engineer must sometimes transform him/herself into "*an explorer*". Students cannot always find their way in the maze of pedagogical resources provided (manuals, tutorials, white papers) or in the explanations that they can get from tutors. So there is nothing to help them through the work they have to do. The student generally feels very uncomfortable and this situation is aimed at pushing him/her to deepen existing knowledge, to

discover new skills, to invent personal solutions. A few students liked this exploration/invention role so much that the tutor had to lead them back to the hard realities of the builder's life ...

Each of these three previous roles bring us back to the question of teamwork and hence to the definition of the role "*a team member*". Students, until now engaged in a rather individualistic learning process (although they are used to negotiating mutual services) must be aware of the fact that nothing can be done without others or more exactly, that they need the others to do everything. It is not possible to describe the attitudes that each team member will have, develop or discover in others. We think that, in a team, each personal quality or skill is useful to the whole and that diversity is guarantee of success. True cooperation is before all complementary rather than community [3].

5 Examining some problems in the light of the learning paradigm

Let us examine some problems we met the second year in the light of constructivism. Constructivism can be summed up with two fundamental statements [4] :

- learning is defined as an active process for knowledge building rather than a knowledge acquisition process;
- teaching is essentially aimed at helping students in this process rather than transmitting knowledge.

Among practices belonging to the constructivist stream (and cognitive psychology), D. Dwyer [5] and J. Tardif [14] define a learning paradigm, in opposition with the main teaching paradigm. The learning paradigm provides a framework which allows the school to constitute a learners' community for the pupils as well as the teachers and the other staff members.

5.1 Roles relationships

5.1.1 Teachers roles

J. Tardif defines teachers' roles as creators of pedagogical environments; interdependent, open-minded, critical professionals; development instigators; mediators between knowledge and students; coaches; collaborators for the students' success of a whole school. Relying on exhaustive definitions of these roles given in [14] p. 59-70, the table below establishes the relationships between these roles and the different kinds of teachers roles described in section 4.1.

Tardif's roles	**Apprenticeship by immersion teacher roles**
-1- creators of pedagogical environments	coordinator
-2- interdependent, open-minded, critical professionals	all roles
-3- development instigators	software manager, apprenticeship, regulator, facilitator
-4- mediators between knowledge and students	software manager, apprenticeship, regulator, facilitator
-5- coaches	all tutoring roles
-6- collaborators for the students' success of a whole school	coordinator

Table 1 : Teacher roles relationships

Let us examine how Tardif's roles are played in our immersion system.

The second role is obviously essential because the apprenticeship by immersion system comes from the professional world. The two tutors exchange daily their information, questions, doubts and the possible solutions. It works fine.

The first and sixth roles are nearly devoted to the coordinator. He/she is responsible for the immersion system and for putting it in practice.

The fifth role – coaches – is the heart of individual and collective apprenticeship. All kinds of tutoring roles are participating.

The third and fourth roles are shared nearly between all kind of tutoring roles.

Mapping Tardif's roles and our roles does not work. There are two main functions in our immersion system : coordinator which is supervising the system and company's tutor which is playing most of Tardif's roles.

5.1.2 Student roles

J. Tardif defines students' roles as investigators; co-operators sometimes experts; clarifying actors; strategic users of available resources. Relying on exhaustive definitions of these roles given in [14] p. 70-74, the table below establishes the relationships between these roles and the different kinds of students roles described in section 4.2.

Tardif's roles	**Apprenticeship by immersion students roles**
-1- investigators	"an explorer"
-2- co-operators sometimes experts	"a team member"
-3- clarifying actors	-
-4- strategic users of available resources	"a builder"
-	"a citizen"

Table 2 : Student roles relationships

None of Tardif's roles includes "a citizen" role. From the learning paradigm point of view, defined for one school, this role does not exist probably because the teacher stays in centre of the class and enforces the discipline.

Investigators and "an explorer", co-operators sometimes experts and "a team member", strategic users of available resources and "a builder" have very similar definitions. It is not surprising because we integrated in the design of our immersion system many models of constructivism.

Our immersion system did not emphasize on the clarifying role and its associated techniques (peer/tutors questioning, checking reformulation and comprehension). However, this kind of clarifying activities is the basis of four apprenticeship tasks. This role may be enforced in the next version of the system.

5.2 Weak points of the system

The main problems which jeopardize the system are pictured in the figure below, linked with the roles involved.

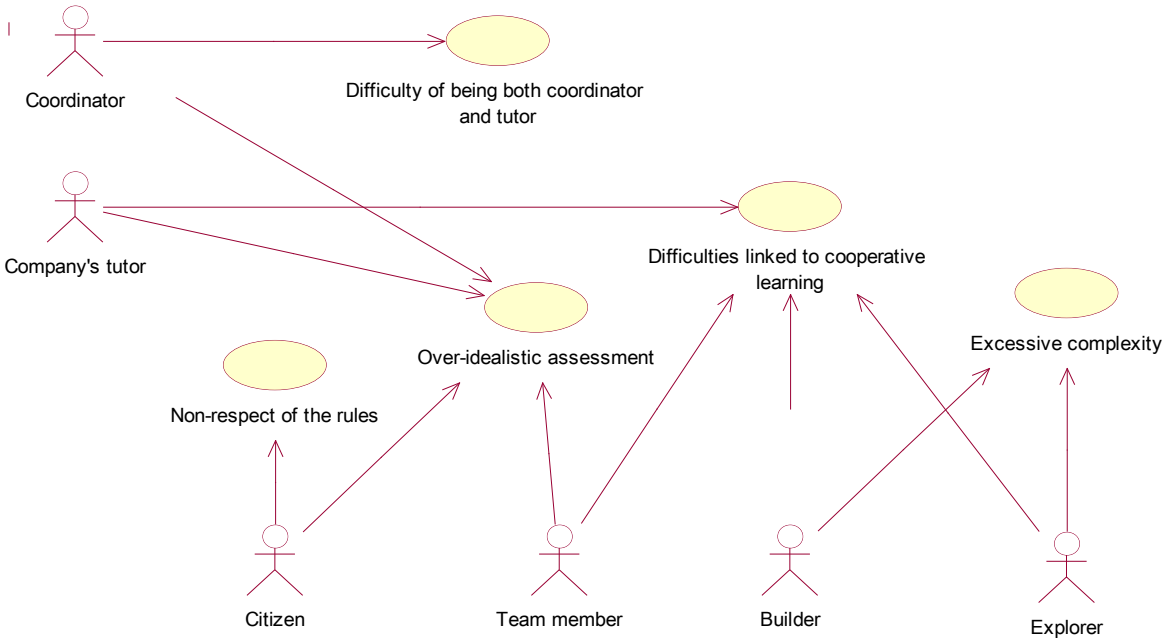

Figure 2 : Relationships between roles and main problems

Each problem is briefly described and possible solutions are drafted. We are aware that these solutions could diverge from the constructivist approach but they are essentially pragmatic.

Difficulty of being both coordinator and tutor

The current system relies on the two authors, each managing a company. The second author also assumes the role of coordinator. It is hard to supervise the system – preparing, coordinating, controlling and accompanying the other company's tutor – while tutoring a company at the same time. Supervising a company's tutor can be compared to a form of teacher tutoring : it requires that activities are well-defined, structured and organized and that events are anticipated. The coordinator has to adjust the general framework according to real work progress. It consumed time and efforts and made it difficult for the coordinator to share the everyday concerns of his own company. Students did not seem to notice it but it caused a lot of inconvenience for the coordinator.

Ideally, the coordinator should not be simultaneously a company's tutor. But the lack of teachers makes it impossible. Hence our ambitious system has to be slightly revised and the interdependence between companies should be diminished by increasing the autonomy and relative freedom of the company's tutor.

Difficulties linked to cooperative learning

Our education system uses a cooperative learning model. The class is divided in teams, each team member builds a competency within a group of 2 or 3 (rarely alone). Each group owns a piece of the puzzle and students are strongly bound together because of the collective results required.

During the first year, we motivated all the students for shifting from compartmentalized knowledge acquisition and sanction-assessment to the process of personal and social construction of SE knowledge intended to be used in a professional environment. We fired them with enthusiasm for this new kind of education. So, we were mistaken in believing that each student would agree with the immersion principles : constructing long-term competencies in SE while being the main actor of his/her education. During the second year, some students moved with difficulty (indeed could not move) from previous attitudes to the requirements of the problem-based and project-based learning (Université de Sherbrooke reported the same difficulty for some students). This could be due to a lack of involvement of students who prefer supporting roles rather than leading roles or it may come from the difficulty to accept tutor's feedback as an essential building element rather than a criticism. The learning process is blocked : what could be done with students who refuse this approach ? This stalemate (mainly for "a builder" attitude), even for a single student, can involve the whole team due to group phenomena. The big role play will then become very difficult for all actors.

We do not have any real solution to this problem. Improving the recruiting process aims to minimize the risk : cooperative learning principles will be emphasized as well as the required attitudes; individual recruitment techniques may be useful.

Over-idealistic assessment

Our assessment process relies on work cards. Cards are essential elements in the tutor(s)-student(s) relationship. The objectives of the card are to describe the work to be done as well as the pedagogical resources supporting apprenticeship. Resulting products should be considered as good enough for their future exploitation. This is a formative assessment of competencies which diagnoses and regulates apprenticeships. Summative assessment is accomplished by giving a mark to each product. We consider that product evaluation is appropriate to judge knowledge and skills acquisition independently from the learning framework.

Assessment has a strong influence on the students' motivation for learning. So it seems important to us to break away from the traditional evaluation process. We privilege collective assessments and we do not communicate any mark to students during the whole year in order to focus on the formative assessment. Selected students graduated from a 4-year technological education system, we assume that everyone will succeed in a traditional system and we wish to offer a transition year between the university and the professional world.

During the first year, this assessment system has been a determining factor in the cooperative learning dynamics. Students quickly understood that learning was intended to reuse competencies in order to diagnose, explain and solve problems rather than restitute knowledge during examinations. During the second year, some students did not play "a team member" role and took advantage of the collective assessment system to have a quiet year, relying on other team members' efforts.

We do not want to fundamentally modify our assessment system. It belongs to the « authentic assessment » stream : this considers moments and contexts where students learn as the privileged place to gather information on assessment itself. Nevertheless, it should be essential to impose sanctions to the students who are insufficiently involved in their apprenticeships :

- *the first iteration, "Tutored apprenticeship", will not change;*
- *the second iteration, "Accompanied application", will aim to enforce the student's autonomy and to individualize summative assessment. Students should keep in mind that knowledge and skills acquired during the first iteration will be put into practice but will also include greater emphasis on individual performance.*

Excessive complexity

Confronting students with complexity is one major goal of the system. This leads to learning situations which are intentionally complex and which some students will not assume. Students could be paralysed with a problem they are not used to solving and could accept no responsibility for this failure and will blame someone else (the education system, tutors and so on).

We must be able to include just the right amount of technological complexity. We agree with Monash teachers who are circumspect about the complexity of using industrial CASE environments [10]. Students can get stuck trying to master commercial tools, rather than learning appropriate abstractions, developing and applying well-suited principles and processes.

Complexity is an essential principle of our education system. We will stress the need for adopting "a builder" and "an explorer" attitudes during the recruitment as well as during the year. Technological complexity should be reduced. Our goal is to come back to initial objectives, i.e. acquire strong competencies in software engineering without getting lost in the complexities of a technological environment.

Non-respect of the rules

This education system is intended for graduate students and relies on students' self-management including the respect of elementary rules such as assiduity, punctuality, deadlines. The necessity of "a citizen" role arose only the second year due to the attitude of one student. Although it happened only once, not respecting the rules is detrimental to the company and hence to the other teams. Our cooperative learning relies on, among other things, the "learning together" idea. Students have to organise their work, to assign tasks, to dispatch responsibilities and to help one another, that is difficult when somebody does not respect the rules.

Next year, we will set up safety and control measures. The students who will not follow the rules repeatedly will be sent back to a traditional system.

6 Related work

The Department of Electrical and Computer Engineering at the Université de Sherbrooke has totally redesigned its electrical engineering and computer engineering programs. Based on recent research advances in cognitive science as applied to student learning, these curricula have led to new instructional models. The programs were built on a competency-based framework, following an original learning approach that combines problem-based and project-based learning (PBL). PBL is the principal mode of knowledge acquisition. Each problem is formulated so that the solving process leads students to discover what of their existing knowledge can be used, what they need to learn, and what skills are required to manage the situation effectively. Competencies are evaluated in terms of behaviours that can be demonstrated and observed in a professional context [7].

Besides formal courses, any curriculum will include software projects. B. Meyer advocates the long-term project as an essential technique, which students should develop over more than a standard quarter or semester – typically over the course of a year. It should be a group project that includes aspects of analysis, design and implementation. And it should involve the reuse, understanding, modification, and extension of existing software [9].

Meyer pointed out that a group of enthusiastic teachers at Monash University, under the direction of Christine Mingins, has been doing exactly that over the past few years (1996-1999). The Monash solution takes place in the second year of an undergraduate program, after that students have already completed two semesters of introductory programming in C++. Monash objectives are :

understand the importance of software engineering, OO design as a software engineering method, motivation, experience a large-scale software engineering project, use an industrial strength CASE environment. For some topics, teachers teach the ideas first; for other topics, however, teachers let the students follow an experimental approach. Mingins and al. believe they have achieved a balance between the conflicting requirements of dealing with the complexities of 'real-world' software engineering based on large projects with the need to establish a firm personal software process which students can use as a basis for reflection and further professional development [10].

7 Conclusion

The education given in the software engineering apprenticeship by immersion system relies on the following principles :

- to be centred on the competencies to be developed and on the fruitful apprenticeship situations, rather than be content with the teaching of knowledge, subject by subject.
- to develop an active and cooperative pedagogy based on the project and the role play: students' immersion in a 6-month project imitating as closely as possible a project in a firm.
- to work in a team, to communicate about the work done, to cooperate with colleagues.
- to elaborate and maintain an apprenticeship by immersion repository.

The first year, there was great enthusiasm coming from students and the two tutors as well. This dynamics helped to solve most of problems.

The second year was rather disappointing. Some of the phenomena we expected did not happen and part of the previous system did not work. The role definitions as stated in section 4 did not allow us to diagnose all problems but the classification of Jacques Tardif let us discover some roles which were played incorrectly or missing, either by tutors or students. We have now to enhance the system in order to prevent these possible failures and improve the role play.

As a conclusion, we can say that if the immersion system is exciting and innovative, on the other hand the system is fragile if both teachers and students are unable to play the roles required. In addition to these first conclusions, the professional insertion and career evolution of students need to be observed over several years in order to evaluate the real benefits of this system.

8 References

[1] Computing Curricula, Computer Science Volume, chapter 10, IEEE and ACM, 2001

[2] Computing Curricula, Computer Science Volume, chapter 11, IEEE and ACM, 2001

[3] Dameron S., 2002, La dynamique relationnelle au sein d'équipe de conception , Le travail humain, Vol. 65, n°4, Oct-Déc 2002, pp 339-361.

[4] T. M. Duffy, D. J. Cunningham, Constructivism : Implications for the design and delivery of instruction, *In* Handbook of Research for Educational Communications and Technology, MacMillan 1996

[5] D. Dwyer, Apple Classrooms of Tomorrow : What we have learned, *In* Educational Leadership, vol. 54, num. 7, 1994

[6] Emmanuel Kant, Réflexions sur l'éducation, Introduction, Vrin, 1998

[7] G. Lachiver, D. Dalle, N. Boutin, R. Thibault, J.M. Dirand and all, Redesign of Electrical and Computer Engineering Programs at Université de Sherbrooke, *In* Proceedings, Canadian Conference on Engineering Education, August 2001, Victoria.

[8] Timothy C. Lethbridge, What knowledge is important to a software professional ?, IEEE Computer, May 2000

[9] Bertrand Meyer, Software Engineering in the Academy, IEEE Computer, May 2001

[10] C. Mingins and al., How we teach software engineering, JOOP, Feb 1999

[11] Gilda Pour, Martin L. Griss, Michael Lutz, The Push to Make Software Engineering Respectable, IEEE Computer, May 2000

[12] Vincent Ribaud, Philippe Saliou, Software Engineering Apprenticeship by Immersion, International Workshop on Patterns in Teaching Software Development, ECOOP 2003, University of Darmstadt, Germany, 2003

[13] Progress of the Engineering Education Coalitions, SRI International, May 2000

[14] Jacques Tardif, Intégrer les nouvelles technologies de l'information – Quel cadre pédagogique ?, ESF, 1998

[15] Thales Information System Glossary, Thales Information System, 1997