# Software engineering apprenticeship by immersion

Vincent Ribaud, Philippe Saliou

▶ **To cite this version:**

Vincent Ribaud, Philippe Saliou. Software engineering apprenticeship by immersion . International Workshop on Patterns in Teaching Software Development, ECOOP 2003., Jul 2003, Darmstadt, Germany. pp.130-142, 10.1007/978-3-540-25934-3_13 . hal-01451171

**HAL Id: hal-01451171**
**https://hal.univ-brest.fr/hal-01451171**

Submitted on 3 Feb 2017

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/254441119

# Software engineering apprenticeship by immersion

**Conference Paper** · July 2003

CITATIONS
7

READS
26

**2 authors:**

Vincent Ribaud
Université de Bretagne Occidentale
**78** PUBLICATIONS **96** CITATIONS
SEE PROFILE

Philippe Saliou
Université de Bretagne Occidentale
**41** PUBLICATIONS **71** CITATIONS
SEE PROFILE

**Some of the authors of this publication are also working on these related projects:**


Analyse des données de simulation pour le diagnostic : application à la validation formelle des modèles de systèmes View project

All content following this page was uploaded by Vincent Ribaud on 02 July 2014.

The user has requested enhancement of the downloaded file.

# Software engineering apprenticeship by immersion

V.Ribaud and P. Saliou
with the collaboration of F. Le Borgne-Uguen[2]
*EA2215, Computer Science Departement, Faculté des sciences, 29285 Brest Cedex, France*
*[2] EA3149, Sociology Research Workshop, Université de Brest, 29285 Brest Cedex, France*
*E-mail:{ vincent.ribaud, philippe.saliou, francoise.leborgne-uguen}@univ-brest.fr*

**Abstract**

**Inside the french university, there are two different education systems: a 5-year academic system and a 4-year technological system. At Brest university and in the field of computer science, both systems teach the same curriculum. A dedicated fifth year was designed and offered to the 4 year-system graduates. The main objectives are : mastering software engineering activities and skills, working in a team, coping with change. The plan of action is built on a 6-month team project, lead and tutored by an experienced software professional. The real-world environment is imitated as closely as possible. A contract defines the customer-supplier relationship. A real corporate baseline (by the courtesy of a software services company) sustains engineering activities and products delivery. This corporate baseline is tailored with a software development process, called 2-Track Unified Process, relying on UML models. The students are divided into two companies of 5 persons. Each company has its own office with individual working post and shares other common installations. Each company uses a different and complete software engineering tools suite (Oracle and Rational/IBM Websphere). 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 to complete the project, the tutor performing mainly a supervising and rescuing activity. The heart of the apprenticeship process is the iterations breakdown into work cards. Each work card defines the products to be delivered, the precise nature of services required, the helpful resources provided, the expected workload and planning. The assessment process is essentially formative, due to the permanent feed-back of tutoring. Nevertheless, the awarding of a diploma needs a parallel formal assessment process. The expected abilities and skills are compared with those effectively reached. In conclusion, 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

Inside the French universities, there are two different education systems, a 5-year academic education system (including two-year general scientific studies) and a 4-year technological education system (including at most one-year general scientific studies). The computer professional branch prefers to hire young graduates with a 5-year curriculum. Hence, students graduating from the technological education system often try to integrate the last year of the 5-year academic education system, generally a one-year course called DESS (*Diplôme d'Enseignement Supérieur Spécialisé*, Specialized High-level Education Diploma).

In the field of computer science, Brest university offers both education systems. Both systems cover roughly the same curriculum. The knowledge and skills acquired at the end of these two kinds of education systems are more or less the same. Brest university has been providing since September 2002 an adaptation of the DESS for the graduates from the technological education system. The opportunity to offer a dedicated fifth 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 [CC2001-11].

The pedagogy used to achieve these objectives breaks with the usual teaching paradigm to rely on an apprenticeship paradigm [Tar98]. We can simplify this paradigm by saying that we try to replace activities linked to the verb « to teach » with activities linked to the verb « to learn ». Moreover, several analyses of software engineering teaching emphasise the benefit of a long-term team project (one semester or a year) [CC2001-10] , [PGL00], [Mey01].

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 ».

The paper is organised as follows : the section 2 presents a general description, the section 3 describes the apprenticeship by immersion system, the section 4 shows the progress of the year and we finish with perspectives and a conclusion.

# 2 General description

The year is divided into three periods: a 4-month tutored apprenticeship period, a 2-month accompanied application period and a training period of 4-6 months in a firm (see figure 1). Apprenticeship is mainly related to software engineering and completed with English and communication courses.

## *2.1 Objectives*

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".

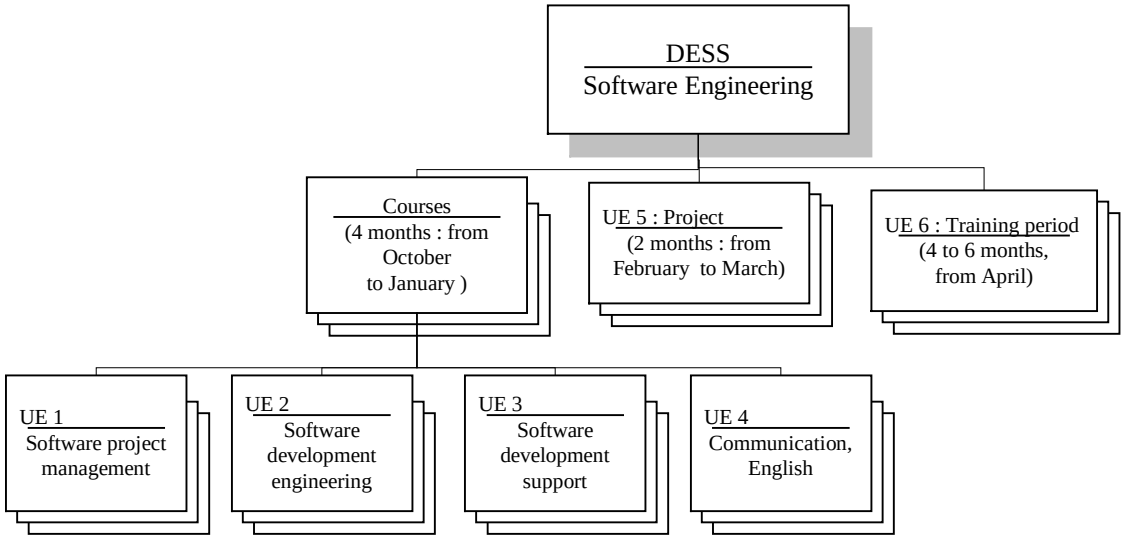


**Figure 1 : Structure of the training course**[1]

As an engineering discipline, software engineering includes a wide range of activities, some of which can be assimilated to actual professions (or specialities). The software engineers we are training may perform these occupations on a short, mid or long-term. The primary authors of this paper structured these 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. This paradigm often leads to teach only technical software engineering. Hence the idea to tackle these activities within a different learning process:

- within a real project,
- and emphasing the software manufacturing process whose activities are divided into the three main areas. These areas are assimilated to manufacturing subprocesses and, hence to specializations in the software engineering profession.

## *2.2 General organisation*

Students are immersed in a real project, which happens exactly as in a firm, with the following specificities:

- the project goes on in the university in a dedicated fitted-out room,
- the project manager who is driving and tutoring the project is a university lecturer,
- the target, the software product, is only a pretext for learning software engineering activities.

During the 2002-2003 academic year, the experimentation concerned ten students, graduates from the technological education system. These students are divided into two separate project teams, that we call companies, and they should have different roles (organisational, quality insurance, technical, support, …) in a collaborative and cooperative context related to working in a team. The two companies will have to produce software for the same order but with different methods, tools and technologies.

The whole apprenticeship plan of action is guided by the development process, which defines, among others, the role and the schedule of project stages. A first iteration of the process lets students acquire knowledge and skills needed for each stage, a second iteration is intended to transform knowledge into abilities and finally the last iteration permits to apply this in a firm.

This apprenticeship plan of action relies on the primary authors of this paper. Before joining the university, both were software project managers for several years within the same software services company. New lecturers or intervening professionals should reinforce this pedagogical team. This new form of teaching reveals clearly three different kinds of 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.

The students' apprenticeship process is elaborated and tailored according to real work progress and each team's specificities.

A different technological framework is provided to each project team. Technical architecture is essential in the building of current computer systems. Hence, each company has a rather complex professional framework (IBM, Oracle), which students master only partly as the beginning of the project.

[1] *UE, Unité d'Enseignement* : Course Unit

In order to emulate a standard firm environment, we fitted out a laboratory room (see figure 2) with:

- a landscape room for each company,


- a common meeting room and an Internet room.


**Figure 2 : Room fit out**

Each student has his/her own individual working post.

# 3 System of apprenticeship by immersion

Since June 2002, we have been elaborating a first version of an apprenticeship repository for the software engineering profession.

The apprenticeship guideline is the project. A project fits into an environment 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 activities allowing to transform users' requirements into software products,
- a corporate baseline which defines good practices and capitalizes the company's know-how,
- a working framework including common installations and tool suites intended for the manufacturing and the documentation of software products.

Apprenticeship by immersion reproduces as much as possible the real environment of a software project and provides an apprenticeship plan of action for the required competencies. This plan of action then consists of an apprenticeship process relying on an apprenticeship repository. This repository is built on and tailored according to pedagogical choices relying on the four essential fundamentals of the environment.

Before describing this apprenticeship plan of action, let us define the immersion environment.

## *3.1 The project immersion environment*

### 3.1.1 Software engineering activities

Main engineering activities tackled during the year are distributed in three main areas as depicted in figure 3.

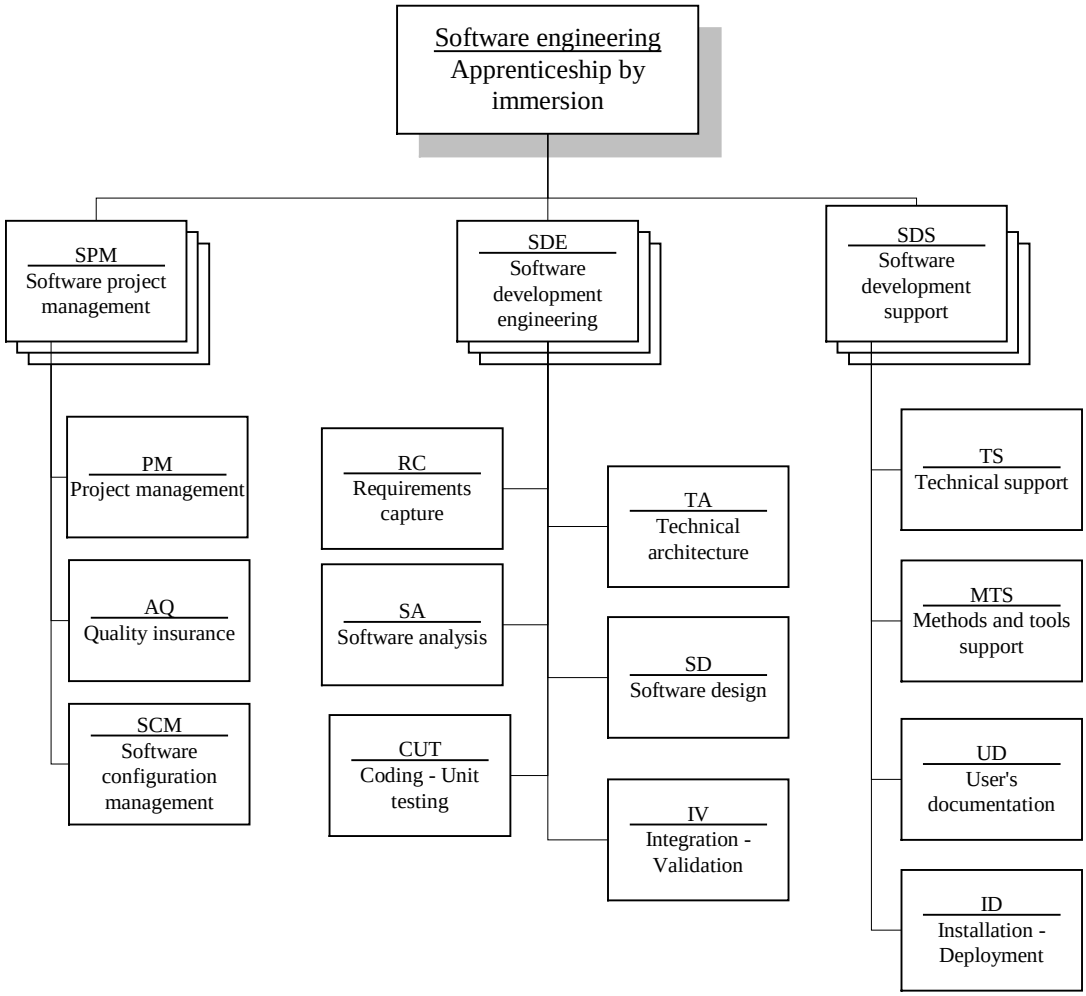


**Figure 3 : Breaking down activities**

### 3.1.2 Corporate baseline

« TEMPO is the foundation of the company corporate culture and is the basis for sustaining good working practices. It is a set of procedures, guides and instructions defining how the company operates, and how it is organised, providing a framework for programme management, software development and system integration activities. » [TIS02].

Thanks an agreement with Thales group, we can use a part of this corporate baseline and more precisely all guides and procedures related to software engineering for pedagogical purposes.

From a development point of view, TEMPO defines a very general life cycle (requirements analysis, preliminary design, detailed design, coding and unit testing, integration and integration testing, software testing). TEMPO is also a documentary repository that provides, for each document we have to deliver at each phase of the life cycle: a standard document frame, a writing guide, and sometimes examples. TEMPO defines as well activities and resulting products for the programme management, the software configuration management, and the software quality programme.

The agreement established with Thales compels us to restrict this corporate baseline use to software engineering. So, we tailored TEMPO in a TEMPO-ILI[2] baseline that we enriched with documents originating from real projects together with references to other baselines (type ISO 9001/9002).

### 3.1.3 Development process

The development process belongs to the unified processes family. « The unified process is first and foremost a software development process ... The unified process uses the Unified Modelling Language (UML) in order to create elaboration and building plans of the software system ... Nevertheless, the truly specific features of the unified process are as follows: use case driven, architecture-centered, iterative and incremental. » [JBR99].

The process we kept is the 2-TUP process, which means 2-Track Unified Process. « The 2-TUP is an unified process ... 2 Track literally means that the process follows two ways. There are the functional way and the technical architecture way, which correspond to the two change axes set to a computer system ... The outcome of the evolution of the functional model and of the technical architecture leads to merge the two-way results, so that the system is realized. This merge leads to the achievement of an Y-development process, as depicted in figure 4 » [RV02].

[2] *ILI, Ingénierie du Logiciel par Immersion*, Software Engineering by Immersion

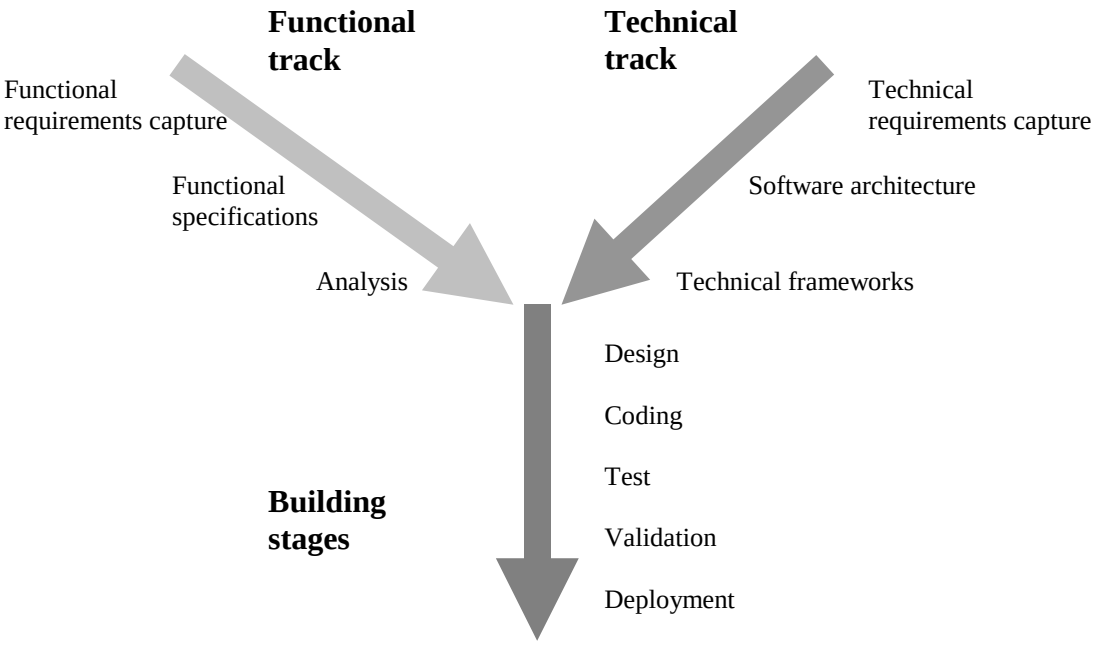


**Figure 4 : 2 Track Unified Process**

The right track of the Y-process allows us to deal with technology complexity.

### 3.1.4 Working framework

Last but not least, the apprenticeship system includes a different technical framework for each project team.

The choices of frameworks was made with several pedagogical objectives in mind:

- To provide tools and technical frameworks that are expected by the market in order to build and to deploy information systems relying on an N-tier architecture.
- To confront students with the complexity of development and deployment infrastructures.
- To supply modelling uppercase tools, high-level development tools relying on technical frameworks allowing a model-driven development process.
- To supply office tools as professional tools for development support such as configuration management tools, planning tools, requirements management tools ...

With few exceptions, UML is the link between these tools.

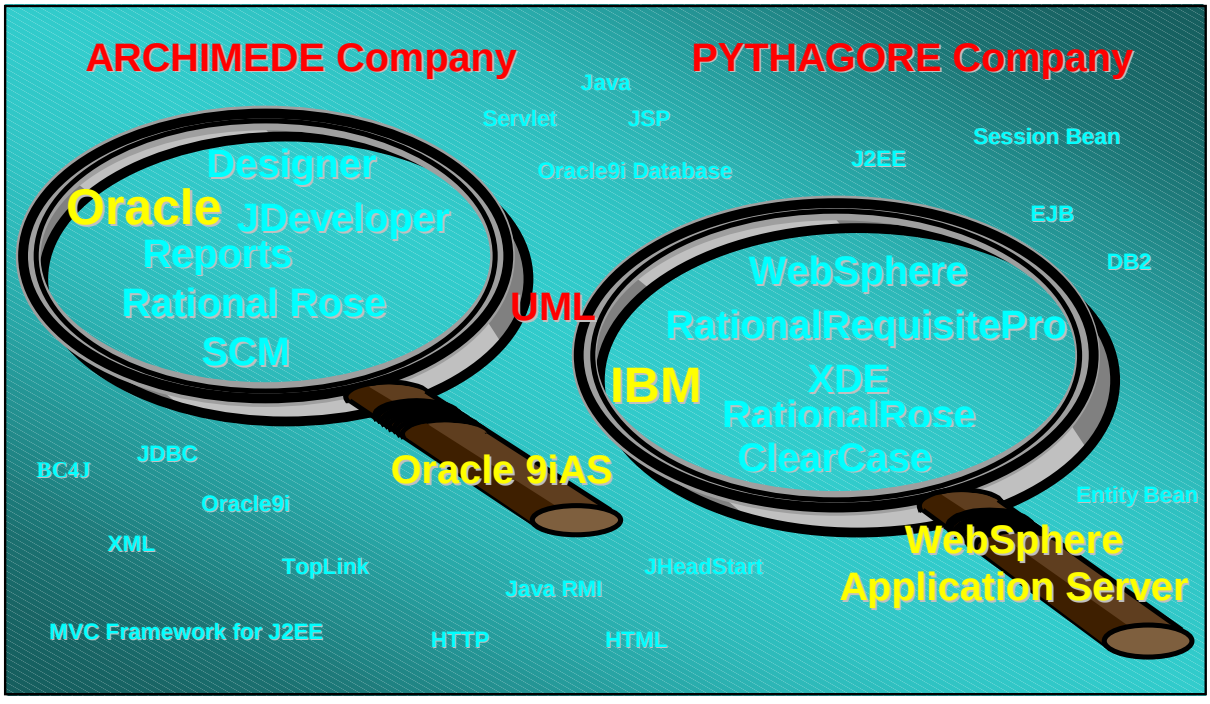


**Figure 5 : Technical framework**

This year, one project team was immersed in an Oracle universe, loosely coupled with Rational tools while the other one had to tackle the IBM/Rational world. For the future academic years, we consider adding two new companies, the former relying on Microsoft .NET architecture (we are waiting for mature uppercase tools), the latter working with the “free” world.

## *3.2 Apprenticeship by immersion*

Our objective is to define a reference framework for all the people involved in this education system.

### 3.2.1 Apprenticeship repository

We merged all the fundamental environment elements in order to build a first version of an apprenticeship repository for the software engineering profession. The apprenticeship system relies partly on the year's division in three periods (iterations), and partly on the apprenticeship process applied during the first two iterations.

The apprenticeship repository includes at this time:

- TEMPO-ILI, a dedicated corporate baseline, including:
  - Elements issued from Thales-IS baseline tailoring.
  - References to other ISO 9001/9002 baselines.
  - Examples of documents produced for real projects.
- Use supports for the technical frameworks :
  - Installation and user's guides
  - White Papers
  - Red Books
- Books as well as self-training media.
- The description of the apprenticeship process consisting mainly of :
  - Work cards defining stage by stage, the work to be done.
  - Pedagogical resources at students' disposal as the project is being carried out.

The manufacturing process of this apprenticeship repository is iterative, which means that the reference framework elaborated during this academic year will be enriched and tailored during the next "Software Engineering by Immersion" years. We expect the repository to be stabilized in 3 to 5 years. Another interesting aspect of these thought processes is that students themselves, with their own work, will bring new elements or new adaptations to the apprenticeship repository.

Elaborating this repository implies that we work in the field of pedagogy as well as in the field of research in methodology.

### 3.2.2 Apprenticeship process

The apprenticeship process is guided by the 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 depends on one of the activities defined in section 3.1.1.
- Each work card is assigned to one or several students who should take up the job (or the role) inherent to the activity.
- There is an important role play because at one stage some students can hold the responsibility of project manager while others are analysts, designers, architects, system engineers … Students are swapped around the roles from one stage to another stage.
- Each stage provides pedagogical supplies (standard document frame, guides, books, self-training supports). For each work card, students can rely on tutor's support and assistance on a daily basis.
- Each work card describes form and content for the expected deliverables. Work cards constitute the principal assessment framework, see section 4.7.

# 4 Year progress

## *4.1 Temporal organisation*

The year is divided in three periods, called iterations (see figure 6):

- a tutored apprenticeship period allows students to acquire software engineering knowledge and skills,
- an accompanied application period has to transform knowledge and skills into competencies,
- and finally a training period to put this in practice in a firm.

The guideline for the two first iterations is the software development project, in which students will be immersed. Each company is associated with a company's tutor (a lecturer), who plays different roles in the first and second iteration. The objective of the project entrusted to each company is to manufacture an information system, which answers the real needs of the computer science department lecturers. This year, it was about automating the course assessment process, as it is already done in some computer science courses (distributing poll forms, gathering students' anonymous answers, computing statistics).

The training course starts after the response to solicitation phase. The role play partly consists in simulating the client-supplier relationship. Other colleagues will play the role of contractors and clients. We have to write requirements, then a response to solicitation including a technical and commercial offer answering the expected needs. These documents are very similar to real documents. They are only different in order to incorporate needs induced by our apprenticeship system:

- Two work packages are expected in order to map the two first iterations.
- Lead-time and cost are adapted to the size of teams and the time available for the training course (4 then 2 months).
- Technical constraints imposed by the client correspond to the objectives and means of the course.

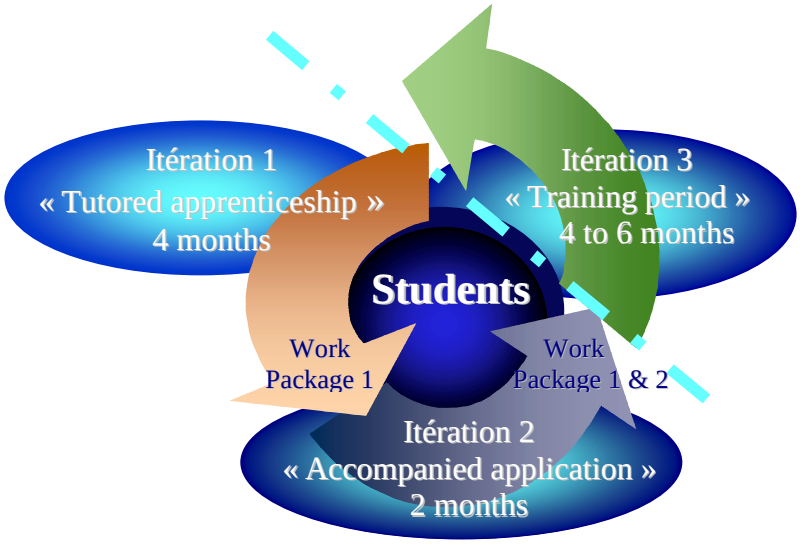


**Figure 6 : 3 apprenticeship iterations**

## *4.2 Iterations*

**During the first tutored apprenticeship iteration** (4 months), an incomplete version of the software is built within a framework entirely driven and tutored by the company's tutor. All software engineering activities are put in practice within a complete software development cycle. The expected software product at the end of this iteration corresponds to the first work package as defined in the requirements document.

**During the second accompanied application iteration** (2 months), each company is relatively autonomous in order to put in practice the knowledge, skills and competencies acquired during the first iteration. The expected software product at this end of this iteration corresponds to the whole software product as defined in the requirements document. During this iteration, the company is still accompanied by the company's tutor. The tutor has mainly a supervising and rescuing function : he/she is an observer, a referee in case of conflicts, he/she can suggest solutions and can answer any explicit demand from students. The tutor should more specifically sustain the student acting as project manager : progress report meeting preparation, planning, task preparation and assignment …

**Finally, the third iteration, i.e. the training period**, should ensure the continuity of the training course. Inside a firm, a software project (from A to Z) could be entrusted to the student according to the development process learned during his/her education. Practising of the software engineering activities is an alternative for the training period, for example managing software configuration, capturing users' needs, analysis or design modelling … For this alternative, we have to deal with job specialisation which can take place in environments and/or with methods and tools different from those used during the apprenticeship period.

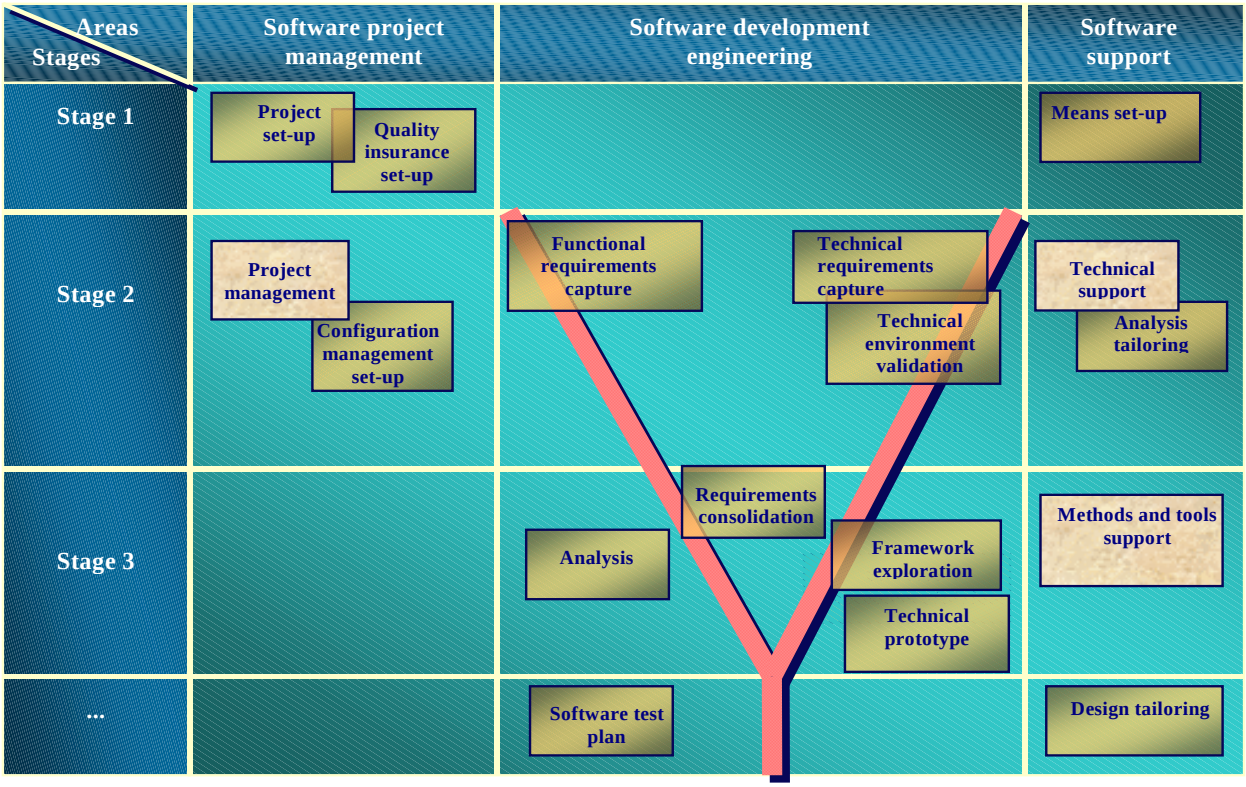


**Figure 7 : A part of the apprenticeship process**

## 4.3 Relationships between activities, apprenticeship process and time

Our apprenticeship process is guided by the Y-software development process. This Y-process is a staged manufacturing process from the contractor's order to the software exploitation.

This apprenticeship process is represented within a two-dimension array (see figure 7).

The three columns (which can be assimilated to lanes) represent the three main activities areas defined in section 3.1.1. These three lanes can be viewed as three complementary and tightly coupled subprocesses.

The vertical axis shows the time. It allows us to materialize stages and milestones scheduling, essential to software engineering apprenticeship. For each stage, there are precise and controlled objectives. A stage is during 2-4 weeks (rarely 5) depending on the activities performed in the stage. This schema is not a task planning, but only a frame representing a macroscopic abstraction of the work that the students will have to tackle during the training course.

## 4.4 Apprenticeship stage structure

Each stage includes different work. For example, in the third stage (see figure 7), there are some distinct work as requirements consolidation, analysis and so on. With this work model and the progress reality as well as with the company's specific constraints, the company's tutor defines the set of tasks at the beginning of the stage. He/she relies on apprenticeship cards of the stage. The tutor's company schedules and assigns tasks to students, at least during the first iteration. The planning is updated at each progress report meeting on a weekly basis. As an example, the third stage planning of one company is given in figure 8.

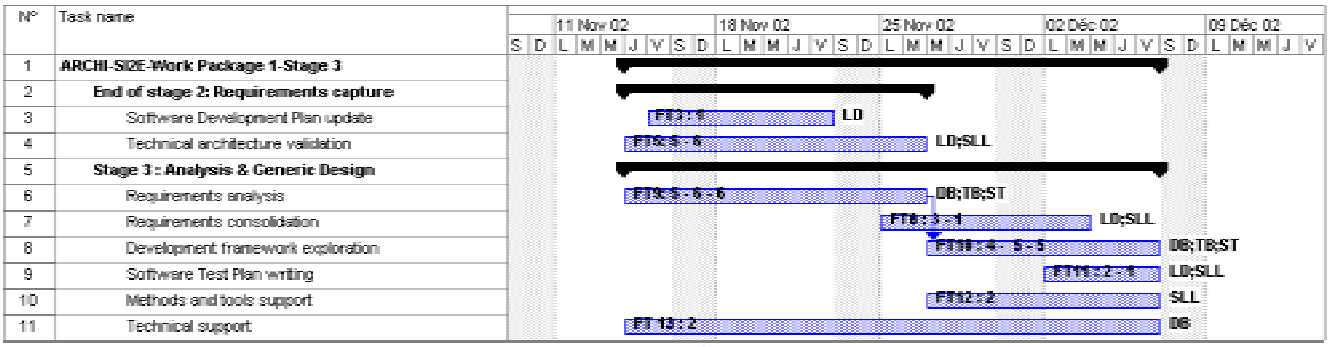


**Figure 8 : Technical framework**

Each task is situated within its activity lane (or area) of the figure 3. Each task is assigned to one or several students who have to take up the job (or role) inherent to the concerned activity. In the example above, there are the software manager role , the analyst role and so on.

## 4.5 Apprenticeship card

The apprenticeship card is an essential element in the tutor(s)-student(s) relationship. The objectives of the card is to describe the work to be done as well as the pedagogical resources supporting apprenticeship.

During the iteration « tutored apprenticeship », the apprenticeship process strongly relies on apprenticeship cards (see a simplified card in figure 9) predefined and tailored by the company's tutor according to the project specificities that he/she is driving.

During the iteration « accompanied application », students start from these apprenticeship cards in order to fit in and to enrich them for their own software development process. This leads to obtain manufacturing cards for the different tasks required. It is a tailoring activity which is running as long as the project is carried out.

| **Number** : 8 | **Date** : 11-21-2002 | **Origin** : Ph. SALIOU | **Action** | |
|---|---|---|---|---|
| **Program : SI2E-WP1** | **APPRENTICESHIP CARD** | | Analyst | **L. Delemazure**<br>**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 10 : Apprenticeship card**

The apprenticeship card structure is standardized. 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 supplied pedagogical resources (here a writing guide and examples),
- the products (deliverables) to deliver (here a Software Requirements Specification document),
- workload and lead-time information.

## *4.6 Implementation of the apprenticeship process*

### 4.6.1 First iteration

During the year 2002-2003, we defined about 40 apprenticeship cards. Apprenticeship cards are distributed along the Y-process axes, within their activity lane and time. The apprenticeship stages occur according to a predefined structure. This theoretical apprenticeship process has to be continuously tailored by the company's tutor, depending on real progress and the project specificities (hardware, software, organisation, ...).

When a stage is completed, new work is entrusted to each student, corresponding to changes along several axes:

- **time**, which allows the student to tackle a new software engineering activity, relying on previous activities and work, hence only making sense at that moment of the process;
- **apprenticeships**, because those needed by this activity do not ensure a continuity with the previous activities;
- **activity area**, from software management to development support, and so on;
- **job specialization**, the role play changes, a student goes from a software configuration management role to a design role, or from analyst to system engineer, ...

Each student has his/her own apprenticeship process. Figure 10 shows the individual advance of a student during this year.

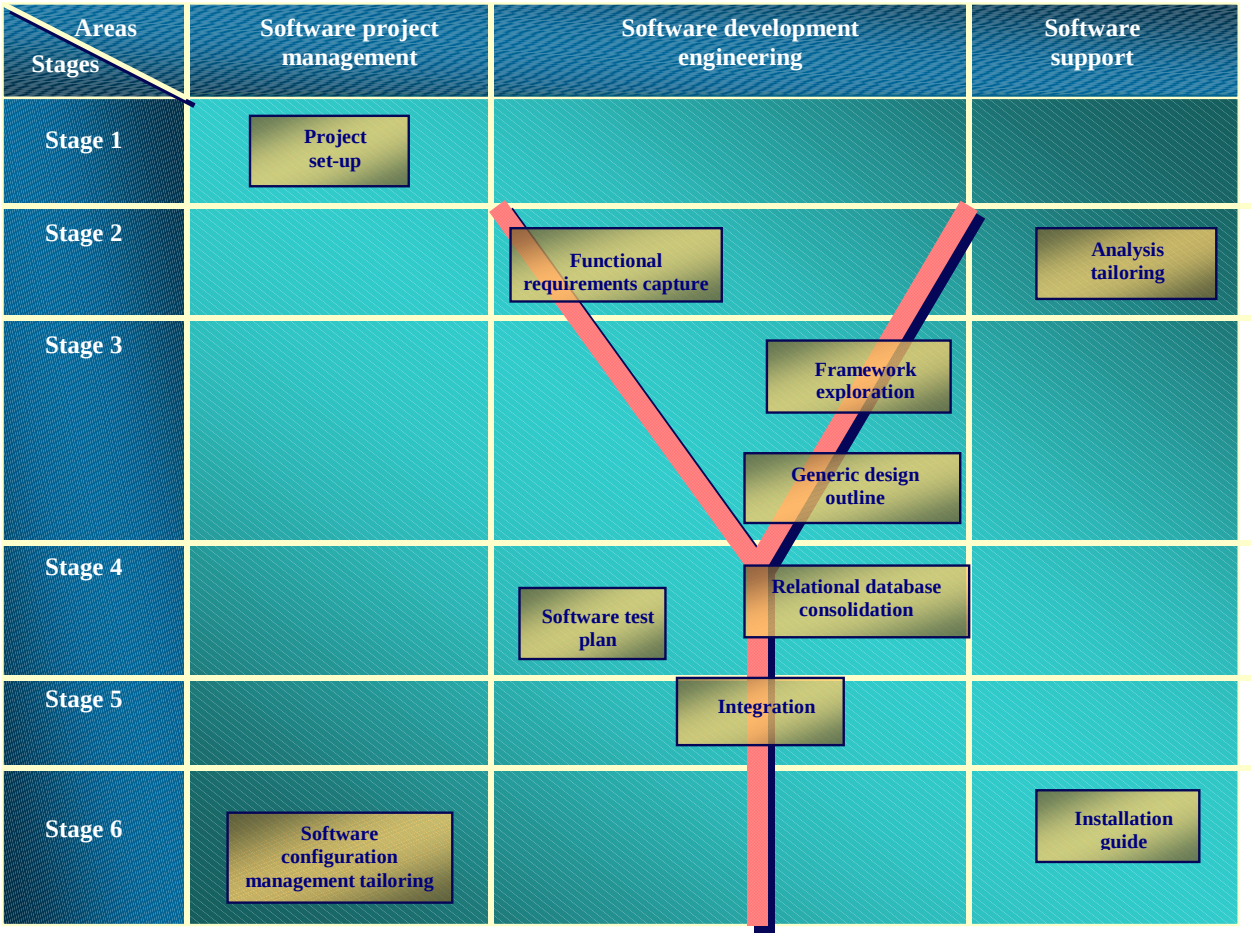


**Figure 10 : An individual apprenticeship advance**

### 4.6.2 Second iteration

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. This year and each company, we find the following roles :

- a project manager,
- an assistant project manager specifically in charge of analysis,
- an architect,
- two developers.

Each company could propose a role distribution, either together (if there is a consensus) or individually. Taking advice into account, the company's tutor defines his/her own role casting which represents the reference authority.

During the second iteration, students will rely on the apprenticeship process acquired during the first iteration. This apprenticeship process becomes a development process according to the technical architecture choices already made, to the development tools used and also according to the deadline constraints coming from the client. This iteration should end with the delivery of the required information system, qualified according to the validation protocol which was elaborated by the project team.

The progress of this iteration is similar to those of the first iteration, except the following facts:

- the company's tutor stands back in order to allow the company to stand on one's own two feet. The company's tutor does not become only an observer, he/she is still supervising the good progress of the project: delivering resources, accompanying students individually, arbitrating conflicts and particularly sustaining the student acting as project manager.
- the student acting as project manager takes on the responsibilities for all management project activities (planning, tasks assignment, progress report and so on).
- from the Apprenticeship Cards (AC), the student acting as project manager elaborates Manufacturing Cards (MC).
- with the company's tutor's help, the student acting as project manager prepares and drives the weekly progress report meeting.

When delay constraints are too high, some manufacturing cards may be not be written. In this case, the progress report meeting account document describes accurately the work assigned to team mate as well as the planning of work.

This second iteration is also an opportunity to develop some engineering activities. This year, deepening focuses on software configuration management. Each company has defined configuration management rules and procedures and has automated a part of these rules through the use of ClearCase (Rational) or Software Configuration Manager(Oracle) tools.

## *4.7 Evaluation*

### 4.7.1 First iteration

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).

The array above describes the link of apprenticeship cards with the activities of the structure depicted in section 3.1.1. (see figure 3 and recall that a UE, *Unité d'Enseignement*, is a Course Unit as well as an activity area). Each apprenticeship card has an assessment and a mark is given to.

| **UE1** | **Software project management** | **Coeff.** | **Apprenticeship card** |
|---|---|---|---|
| | Software project leading | 2 | Project set-up<br>Progress report meeting accounting |
| | | 1 | Quality programme set-up<br>~~Quality insurance~~ |
| | Software configuration management | 2 | Software configuration plan elaboration<br>*Software configuration plan elaboration*<br>Technical incident management<br>*Version management* |
| **UE2** | **Software development engineering** | **Coeff.** | **Apprenticeship card** |
| | Requirements capture | 2 | Functional requirements capture<br>Technical requirements capture<br>Requirements consolidation |
| | Technical architecture | 2 | Technical architecture validation<br>Development framework exploration<br>Generic design<br>Technical prototype |
| | Analysis | 2 | Requirements analysis<br>*Human-computer interface mock-up* |
| | Design | 2 | General design<br>Relational database consolidation<br>Detailed design |
| | Coding – Unit testing | 2 | Coding – Unit testing |
| | Integration-Qualification | 2 | Software test plan elaboration<br>Software test dossier elaboration<br>Integration<br>Internal validation<br>Qualification |
| **UE3** | **Software development support** | **Coeff.** | **Apprenticeship card** |
| | Technical support | | Means set-up<br>System and networking support<br>Database administration<br>Development technical support |
| | Tools and methods support | 2 | Analysis environment definition and set-up<br>Design tailoring<br>Development support<br>Software configuration tool mastering |
| | User's documentation | 2 | Software user's guide writing<br>Software installation and exploitation guide writing<br>~~Training support writing~~ |
| | Installation - Deployment | 1 | *Target installation and configuration*<br>*Software deployment* |

This year, some apprenticeships (~~crossed~~) were partly treated (sometimes not at all) and some others were developed (*in italics*) only during the second iteration.

### 4.7.2 Second iteration

The objective of the second iteration is to put in practice the knowledge and skills acquired during the first iteration. We did not have enough time 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 sommative assessment of the essential deliverables of a software project.

| UE5 Projet | Coeff. | Deliverables |
|---|---|---|
| Viva voce examination | 2 | |
| Individual written report | 2 | |
| Work | 4 | Software Requirements Specification<br>Software Analysis Report<br>Software Design Report<br>Software Test Plan<br>Software Test Dossier<br>Software User's Guide<br>Software Installation and Exploitation Guide<br>Delivered information system (the software product) |

## *4.8 Tutor roles*

### 4.8.1 Before operations beginning

In order to guarantee the good progress of the training course, preparation work is necessary. This work is accomplished by a "**Coordinator-Tutor**" 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.

### 4.8.2 First iteration

The smooth running of the first iteration relies partly on the "Coordinator-Tutor" concerning the apprenticeship system, but mostly on the company's tutor, who has two functions :

- Software manager-Tutor,
- Apprenticeship-Tutor.

The "**Coordinator-Tutor**" is pursuing his/her previous activities (see previous section). He/she guarantees the apprenticeship repository consistency and is the privileged interlocutor for company's tutors. He/she manages common resources, adjusts the repository to the companies' requirements, tries to anticipate the next year. A part of his/her activities are still running during the second iteration to a lesser degree.

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 on a daily basis, and refers to the "Coordinator-Tutor" as much as necessary. His/her intervention begins at the first apprenticeship stage. Unlike what happens in a firm, he/she will delegate to his/her students' team the responsibility of the project organisation and management. Hence, writing the Software Development Plan is entrusted to three students who have to define work organisation, the schedule and the milestones, the human and material resources, the methods and techniques used, the tools, ...

Once the project starts, 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 people who are kept in touch with the apprenticeship system : the computer science department manager, system and networking engineers, other colleagues. He/she prepares and drives the weekly progress report meeting, which is a privileged time to exchange information within the team. In turn, each student will be entrusted with the meeting accounting, under the supervision of the "Software manager-Tutor".

In addition to strong technical knowledge and skills and significant experience in software engineering activities, the "Software manager-Tutor" role needs rigour, method and an excellent organisation. For the big role play, he/she should have good relationships, communication (indeed theatrical) capabilities as well as be able to improvise.

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 3.2.2) and on apprenticeship cards (see section 4.5). 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. When the product is delivered, the "Apprenticeship-Tutor" writes an assessment card including a general assessment together with all the points to improve or to start over. The feedback is given in front of all the team mate, it allows the team to deepen, to discuss, eventually to contest remarks made by the "Apprenticeship-Tutor". Following this briefing, students have to update or start again their product, which will be assessed again. Generally, two assessment iterations are necessary in order to guarantee a satisfactory result, in all case sufficient to carry on the project. Even if the product is not so good at all, the "Apprenticeship-Tutor" should not emphasize too much the criticisms in order to keep the group dynamic (the "Apprenticeship-Tutor" may have to be moderated and to expurgate his/her remarks). However, there could remain situations where the poor quality of the deliverable will impact the whole project ...

### 4.8.3 Second iteration

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

- Regulator-Tutor,
- Facilitator-Tutor,
- Programme manager-Tutor.

These roles are not distinct but there are essential to the smooth running of the project in the theatre of operations which represents the Software Engineering Apprenticeship by Immersion.

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. It could be assimilated to a "peer review" activity, but it is also used as an assessment activity (firstly formative, secondly sommative). The consolidation is essential in order to correct weak deliverables which should compromise the coming stages of the project. These "peer review" can be seen as a kind of hierarchical approval, which makes the students more confident.

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. Il is a strong assistance for: progress report meeting preparation, planning, manufacturing card writing, task assignment … It has to sustain the project manager during the weekly progress report meeting when the student is in trouble regarding either technical or functional problems or a conflict that should be resolved as soon as possible.

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. However, there are some situations where a hierarchical power should be exerted on a part or on the whole team. Acting as a teacher and in an assessment context, the tutor has a partly hierarchical power, but this power is outside the scope of the project. 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. "Working with the sales 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. The programme manager is also responsible for ensuring that the programme achieves the expected results and meets financial forecasts. [TIS97]". 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.

# 5 Sociological aspects

The primary authors solicited a university department, called URAFF (Unité de Recherche-Action pour la Formation de Formateurs, *Research-Action Unit on Training Trainers*) in order to carry out action-research in the current experiment.

An action-research system using observing participation is set up with the ten students, the two primary authors and F. Le Borgne-Uguen, lecturer in sociology. Two students representatives are invited.

Data gathered and analysed are :

- observation of teams at work and observation of deliverables,
- four discussion meetings, during 2-3 hours, the first and the second being opened to the students representatives external to the system,
- a story written by each student accounting for his/her life during the first iteration.

At this point, at least two kinds of observations can be made, relying on the work described in [Dam02].

Firstly, this apprenticeship process can be transferred within the professions' environment because of the following aspects :

- The capability to move from a community cooperation to a complementary cooperation.
- The setting of a transferable competence into future work organisation : a thought process which consists in learning to learn.
- A relational dynamics coupled with a competency acquisition process which guarantees the capabilities to socialize and be integrated at work.

Secondly, the training course is built on apprenticeship and production stages, mobilized in both iterations.

This allows to :

- take in account the structuring role of the time and its importance at work.
- establish cooperative relationships among apprentices that lead to alternate between individual working time and real team work time.
- be situated in a community cooperation. It is based on a common identity because students share the common lot of apprentices. It is founded on common objective references (to pass the examination, to aim at the same jobs), and on sharing common objectives.

- leave this kind of community cooperation which is in fact juxtaposed. For the students, it is about becoming involved in a complementary cooperation, based on a negotiation among participants and on sharing resources. It mobilizes a calculated rationality, which is based on mutual confidence regarding the necessary contributions of each student to the project.

# 6 Perspectives

We still have not treated the whole information gathered this year and the third iteration will still be running until September 2003. Following the young graduates' professional insertion will provide a significant assessment of the system, particularly by comparing it with the professional insertion of students graduating from the classical system.

The system is kept for the next academic year (2003-2004) with a challenge : the individual and collective assessment, for the apprenticeship, competencies and performances. The board of directors of Brest university has just approved the future assessment framework. We have now to put it in practice.

Major issues remain to explore. For example, rather than building software from scratch, we would like to start the year from previous projects in order to deal with legacy system and re-engineering.

At the end of next year, we feel that we will be able to use this experimental system to serve as a basis for a master of software engineering, when Brest university will integrate the new higher education European space (called Bachelor-Master-Thesis system).

# 7 Conclusion

The fifth year should not overload the acquired learning with disciplinary and didactic knowledge, if there is no time to learn how to integrate and mobilize them.

The important point is to define competencies and work knowledge as resources rather than final goals. It is not about shifting from disciplinary or methodological knowledge, but to use them concretely. This approach applied to software engineering leads to revising the education system.

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

- to be centered on the competencies to be developed and on the fruitful apprenticeship situations, rather than content us 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 a apprenticeship repository, the foundations of the system presented in this paper.

This year, there was great enthusiasm coming from students and the two tutors as well. This dynamics helped to solve most of encountered problems. Students are very satisfied with the training course and begin a "by word-of-mouth" advertising which is a proof of success.

The weak point is the individual assessment, for knowledge, learning, competencies and performances as well. Our challenge for the next academic year will be to tune individual supervision and measurement tools.

# 8 References


[CC2001-11] Computing Curricula, Computer Science Volume, chapter 11, IEEE and ACM, 2001

[CC2001-10] Computing Curricula, Computer Science Volume, chapter 10, IEEE and ACM, 2001

[PGL00] Gilda Pour, Martin L. Griss, Michael Lutz, The Push to Make Software Engineering Respectable, IEEE Computer, May 2000

[Mey01] Bertrand Meyer, Software Engineering in the Academy, IEEE Computer, May 2001

[Tar98] Jacques Tardif, Intégrer les nouvelles technologies de l'information – Quel cadre pédagogique ?, ESF, 1998

[TIS02] TEMPO, la maîtrise du développement de systèmes informatiques, Thales Information System, 2002
*[TIS02] TEMPO, information systems development under control, Thales Information System, 2002*

[JBR99] Ivar Jacobson, Grady Booch, James Rumbaugh, The Unified Software Development Process, Addison-Wesley Longman, 1999

[RV02] Pascal Roques, Franck Vallée, UML en action, Eyrolles, 2002

[Dam02] 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.

[TIS97] Thales Information System Glossary, Thales Information System, 1997




View publication stats