Learning by doing software engineering

Philippe Saliou, Vincent Ribaud

LEARNING BY DOING SOFTWARE ENGINEERING

Philippe Saliou
Département informatique, U.B.O.
C.S. 93837
29238 Brest Cedex 3
Philippe.Saliou@univ-brest.fr

Vincent Ribaud
Département informatique, U.B.O.
C.S. 93837
29238 Brest Cedex 3
Vincent.Ribaud@univ-brest.fr

ABSTRACT

Learning software engineering is often performed 'by doing'. Shifting to the constructivism paradigm as far as possible, we have designed an education system called «Software engineering apprenticeship by immersion» entirely based on a 7-months project, performed by a 6-students team within a virtual company and tutored by an experimented software engineer. The immersion system is skill-oriented and it may be difficult to classify system's units in a topical-oriented book of knowledge. Some individual and organizational consequences are drafted.

Keywords

Software engineering, learning by doing, immersion.

1. INTRODUCTION

Most software engineering (SE) professionals will say that they learned the profession "by doing". Hence, the main paradigm used is teaching software engineering by doing. Most academic curricula address this issue through projects [1]. Since September 2002, the second year of Brest University Master in software engineering is entirely based on a 7-months project, performed by 6-students team within a virtual company and tutored by an experimented software engineer. This paper briefly presents this education system and states the gap between knowledge-oriented and skill-oriented education systems. The need for an individual and an organizational shift is drafted.

2. SOFTWARE DEVELOPMENT

Software engineering can be minimally defined as the set of activities involved in developing, operating and maintaining software. Software development is organized around a project which involves people guided by a software development process. The process organizes the set of activities allowing to transform users' requirements into software products. Products are artifacts (or deliverables) that are created during the life of the project, such as specifications, architectural design, source code and documentation.

3. SOFTWARE ENGINEERING BY IMMERSION

Industry complains that graduates take at least one year to become productive once hired. So, 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» [2].

The year is divided into three periods: a 5-month tutored apprenticeship period, a 2-month accompanied application period and a training period of 4-6 months in a firm. Except for English and communication courses, no lectures are given. During the first period, students are swapped around the different tasks needed by engineering activities and strongly guided by the tutor. During the second period, 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.

4. KNOWLEDGE AND SKILLS

A recognized profession must have an established body of knowledge and skills that its practitioners understand and use consistently. Building curricula and body of knowledge are typical activities supported by professional societies such as the Association for Computing Machinery (ACM) and the IEEE Computer Society and results are available to the public. On the other hand, typical software engineering usages and skills are gathered in corporate baseline which defines good practices and

capitalizes the company's know-how but diffusion and use are restricted to the company.

4.1 A book of knowledge

Achieving consensus by the profession on a core body of knowledge is an essential goal of the Software Engineering Body of Knowledge (SWEBOK) project. The product of the SWEBOK project will not be the body of knowledge itself, but rather a guide to it. This Guide will seek to identify and describe that subset of the body of knowledge (BOK) that is generally accepted [3]. The Guide organizes the BOK into several Knowledge Areas (KA). Each KA is decomposed into a set of topics.

4.2 A corporate baseline

The authors both worked several years at Thales Services, a software services company. The authors led projects and developed several management information systems under the control of TEMPO, Thales corporate baseline. « TEMPO is a set of procedures, guides and instructions defining how the company operates, and how it is organised, providing a framework for project management, software development and system integration activi-ties [4]».

4.3 An apprenticeship baseline

Thanks an agreement with Thales group, we use this corporate baseline as a part of an apprenticeship baseline. The decomposition into activities was used as the structure of the curriculum. Each activity includes several tasks. Each task is described with work cards that define precisely the work to be done, the resources needed and the results (deliverables).

4.4 Linking knowledge and skills

There is a strong relationship between Knowledge Areas and topics of the SWEBOK and activities of a corporate software engineering baseline. An interesting issue arises when we worked to establish the diploma supplement for our immersion curriculum. For each unit of the curriculum, we were required to establish two lists: one for the knowledge covered in the unit and the other for the abilities linked to the unit. We established easily the abilities list from the description of activities and tasks found in the corporate baseline and in the work cards of the curriculum. It was more difficult by far to extract the knowledge really covered by the unit, even with the help of the SWEBOK. Let us present an example with the "Quality Assurance (QA)" unit. During this activity, students perform quality assurance control established in the QA plan, mainly technical reviews and product inspections. Such are the abilities of the QA unit. Let us have a look on the SWEBOK guide about the KA "Software quality". This KA is divided in 11 subtopics, including the "Reviews and Audits" subtopic. Practically, students learn this subtopic and only this subtopic by doing, but through performing this review activity there are linked to several other subtopics of the KA.

5. SHIFTING THE IMMERSION PARADIGM

At the individual level, shifting to the immersion paradigm means a new way of thinking for tutors and students. The immersion system belongs to the constructivism approach, which can be summed up with two fundamental statements: 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 [5].

At the organizational level, the shift involves at least two dimensions. First, lectures are replaced with student learning. Secondly, the whole immersion system should operate as a learning organization.

In the first dimension, "teachers" perfom three different kinds of activities: - Organising - It is a matter of scheduling and elaborating work cards that define, week after week, the work to be done and preparing pedagogical deliveries 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; - Continuous assessment - Each deliverable is carefully annotated by tutors, together with improvements to bring about. This assessment and feedback process is iterated at least twice.

In the second dimension, our system re-unifies the place, the time and the means of apprenticeships thanks to the immersion in a realistic (but not real) project. Learning is cooperative, collaborative and supportive. Assessment emphasises on generating questions and learning from errors.

The main goal of the system is to educate skilled but also reflective practitioners (Donald Schön).

6. REFERENCES

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

[2] Vincent Ribaud, Philippe Saliou, Software Engineering Apprenticeship by Immersion, International Workshop on Patterns in Teaching Software Development, ECOOP'03, 2003.

[3] Guide to the Software Engineering Body of Knowledge http://www.swebok.org/overview/

[4] TEMPO, information systems development under control, Thales Services, 2002

[5] 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