Building an Observatory of Course-of-Action in Software Engineering: towards a Link between ISO/IEC Software Engineering standards and a Reflective Practice

François-Xavier Bru, Gaëlle Frappin, Ludovic Legrand, Estéban Merrer, Sylvain Piteau, Guillaume Salou, Philippe Saliou, Vincent Ribaud

Building an Observatory of Course-of-Action in Software Engineering: towards a Link between ISO/IEC Software Engineering standards and a Reflective Practice

François-Xavier Bru[1], Gaëlle Frappin[2], Ludovic Legrand[1], Estéban Merrer[1], Sylvain Piteau[3], Guillaume Salou[4], Philippe Saliou[5] and Vincent Ribaud[5]

[1] Thales Airborne System, 29283 Brest Cedex 2, {François-Xavier.Bru, Ludovic.Legrand, Esteban.Merrer@thalesgroup.com}
[2] Teamlog, Rue Fulgence Bienvenüe, 22300 Lannion, Gaëlle Frappin @teamlog.com
[3] Direction des Constructions Navales - DCNS, route de la corniche, 29200 Brest, Sylvain.Piteau@dcnsgroup.com
[4] Groupe Arkéa, 32 rue Mirabeau 29480 Le Relecq Kerhuon, Guillaume.Salou@arkea.com
[5] University of Brest, CS 93837, 29238 Brest Cedex, France
{Vincent.Ribaud@univ-brest.fr , Philippe.Saliou@univ-brest.fr}

Abstract. As a help to compete in an evolving market, small software companies may use an observatory of their course-of-action. The course of action considers the observable aspect of the actor's activity. Its analysis provides a description of actors' activity and it can express recommendations concerning both the individual situations and the collective situation. The observatory is an articulated set of data collecting methods supported with semantic wikis and a dedicated application. A case study, based on the activity of a team of 6 young software engineers, depicts some aspects of the building and the filling of the course-of-action observatory. As primary results of this work, we may think that observing and analyzing software engineer's activity help to reveal his/her theory-in-use – what governs engineers' behavior and tends to be tacit structures – That may help engineers to establish links between "Project Processes-in-use" and a simplified Process Reference Model and contribute to reduce the fit between a project-in-action and espoused SE standards.

Keywords: course-of-action, theory-in-use, espoused theory, reflective practitioner, software engineering processes.

1 Introduction

For many small software companies, software process improvement (SPI) is often out of reach due to prohibitive costs and lack of SPI knowledge. However, to survive in this competitive market, software developers must improve their productivity, time to market and customer satisfaction. A help could be provided through a reflective attitude (D. Schön [1]). A question occurs: "How to bring this reflective (and learning) attitude into organizations and everyday work?"

Theories of action study what an actor do, in a given situation, in order to achieve consequence or objectives. A distinction can be made between two kinds of theories of action. Espoused theories are those that an individual claims to follow. Theories-in-use are those that can be inferred from action [2]. Espoused theory and theory-in-use may be inconsistent, and the agent may or may not be aware of any inconsistency. By definition, the agent is aware of espoused theory. Theories-in-use can be made explicit by reflecting on action [2]. In the software engineering field - and especially in Very Small Enterprises – the horizon of standards or the corporate baseline of processes and practices constitute the espoused theory, since it is what engineers claim to follow. Although an emerging standard "Software Engineering - Lifecycle Profiles for Very Small Enterprises (VSE)" [7] may facilitate the use of SE standards in a VSE, what engineers do (and this action is designed and do not "just happen") may reveal a different theory-in-use. We believe that making explicit theories-in-use may help software engineers to learn more suitable theories-in-use, thus contributes to improve productivity and performance.

In this perspective, after several years of informal methods to analyze and improve software engineers' activities, we are now using the course-of-action analysis in order to understand the structural coupling of a software engineer with his/her environment and especially lifecycle software processes. Let us cite a short definition of course-of-action: "the activity of one (or several) specific actor(s), engaged in a specific situation, belonging to a specific culture, which is significant for the latter, in other words, that can be related or commented by him (or them) at any moment [4]." The course-of-action analysis is based on an observatory that we consider in this introduction as a system of data collecting methods. The data necessary to study the course of action includes continuous observations of the behavior of action and communication in a work situation as well as different traces of other elements such as interpretations, feelings, and judgments [4]. The analysis of this data produces a decomposition of the global dynamic in terms of smaller units and the relations of sequencing and embedding between these units. The results of this analysis may (i) help to design better interactions or corrective situations; (ii) facilitate the reconstruction by the actor of his/her own activity, i.e. going from "pre-reflective consciousness" towards a reflective attitude [1].

This paper is organized as follow. Section 2 presents the course-of-action framework and its application to software engineering. Section 3 drafts some related work. Section 4 discuss about the observatory of course-of-action of software engineers. Section 5 present excerpts of a case study. We finish with perspectives.

2 Course-of-action Applied to Software Engineers' Activity

2.1 The Course-of-action in a Nutshell

Pinsky and Theureau, ergonomists, initiated the theoretical and methodological framework of "course-of-action", summarized in one directing idea, that of the necessity of an analysis of the actual operators' activities in real work situations for

the design of new work situations [5]. An important theoretical hypothesis is that the course-of-action framework states about human activity, is that human activity is dynamically situated, i.e. always appeals to resources, individual as well as collectively shared to varied degrees, which stem from constantly changing material, social, and cultural circumstances. The course-of-action analysis add to various theories of "situated activity" the consideration of the domain of experience, i.e. that of the agent's course-of-experience, of the constructing process of this experience at any moment, and takes an interest in the articulation between the cognitive domain and the course-of-experience. Theureau in [6] defines the theoretical object called "course of action" as follows: *"what, in the observable activity of an agent in a defined state, actively engaged in a physically and socially defined environment and belonging to a defined culture, is pre-reflexive or again significant to this agent, i.e. presentable, accountable and commentable by him/her at any time during its happening to an observer-interlocutor in favourable conditions".*

2. 2 The Observatory of Course-of-action

This paragraph is reproduced from [7].

The course-of-action analysis is based on an observatory that allows to specify the material conditions of situated recall (time, place, material elements of the situation), the follow up and the guiding of presentations, accounts and commentaries by the agents as well as the cultural, ethical, political and contractual conditions that are favorable to observation, interlocution, and creation of a consensus between the agent and the observer-interlocutor [6].

A methodology has been developed to collect data on the courses-of-action. It connects continuous observations and recordings of the agents' behavior, the provoked verbalizations of these agents in activity (from the "thinking aloud" for the observer-interlocutor to the interruptive verbalizations at privileged moments) and the agents' comments in self confrontation with recordings of their behavior [6].

Continuous observations and recordings together with verbalizations and self-confrontation let us access to a representation of dynamics of the structural coupling between the actor and his/her situation (including other actors) [9]. A "semiological framework" [6] provide us with a theory of activity allowing to describe the activity in abstract terms expressing hypothetical invariants. Explaining and using this theory is out of the scope of this paper focused on the observatory of course-of-action. It is sufficient to tell that this semiologic stems from the hypothesis that any period of course-of-action may be described in smaller units. This description of the intrinsic organisation of the course of action articulates two complementary descriptions: a description of its global dynamics, characterising the units of the course of action and the relations of sequencing and embedding between these units; a description of its local dynamics, characterising the underlying structure of the elementary units [5].

2.3 An Observatory of Software Engineers' Activity

The intervention of an ergonomist in an organization intended to produce software concern the analysis of human-system interaction – of the software engineer with his/her organization's processes – and the design of the system in order to optimize human well-being and overall system performance. In our case, we use the theoretical and methodological framework of course-of-action in order to analyze the activity of software engineers within Very Small Enterprises (VSEs, up to 15-25 employees).

Recall the definition of the course-of-action in §2.1: what, in the observable activity of an agent […] is pre-reflexive or again significant to this agent, i.e. (i) presentable, (ii) accountable and (iii) commentable by him/her at any time during its happening […]. Software workers do not achieve complex technical gestures or do not have to progress along a detailed procedure. So (i) presentations to an observer are quite difficult to reproduce and presentable artifacts that are most notable and representative of the job are the outputs of software activities and tasks. (ii) Accounts are easier to collect and observe because a minimum of traceability and reporting is performed in any organization and if it is not sufficient, accounting can be provoked without significantly modify the course of the activities. (iii) Comments are not natural objects and have to be provoked: reports, self competency assessment (§ 4.3).

The course-of-action framework proposes self confrontation as an indirect means to document actor's experience or pre-reflective consciousness or immediate understanding of his/her activity at every instant t; the fact is highlighted that the experience at instant t differs from what is called the reflective consciousness, which concerns particular and situated periods of the actor's activity, when he/she considers his/her past activity with a given purpose [8].

However, considering these two levels of consciousness, we may think that there are two different levels of description of software processes. The first level – on which this paper is focused – is concerned with the day-to-day course of a software project and its associated activities while the second level – on which most Software Engineering standards are focused – is concerned with a description of these activities. We believe that the first level is related with theories-in-use, those that can be inferred from action [2]. And we think that the second level is related with espoused theories, those that an individual claims to follow. The purpose of our work is to provide an observatory of existing processes and practices that could help to situate project processes and practices in-use regarding to espoused standards.

2.4 Application for Software Engineers in VSEs

The semiological framework of course-of-action makes it possible to describe the courses of action in general structural terms, expressing underlying regularities. It allows on the one hand, such a description of the global dynamics of the courses of action, and on the other hand, such a description of their local dynamics. It also links these two descriptions. As we discuss in §5.3, the smaller units, based on individual courses-of-action, describe the carrying out of all or part of software engineering base practices. Hence, the global dynamic, which is related to the composition of these performed practices, is a description of what we may call process-in-action.

The course-of-action analysis operates on what, in the observable activity of an agent, is presentable, accountable and commentable by him/her. A sound analysis may work only with sound collected data and, because most accurate data are collected by the team itself, it requires the team commitment to this self-observation. This team commitment can only be effective if the team is the main beneficiary of this overwork, collectively - with a valuable result on team processes-in-action - and individually - with an added-value on competency development -.

Thus, as presented in figure 1, this analysis shall lead to (i) help to specify the modalities of engineers' interaction with project processes leading to the design of better interactions or of corrective situations; (ii) contradict or support the reconstruction by the engineer of his/her own activity, i.e. going from "pre-reflective consciousness" of the actor towards a reflective practitioner attitude [2]. Both results have a valued impact on the project processes.

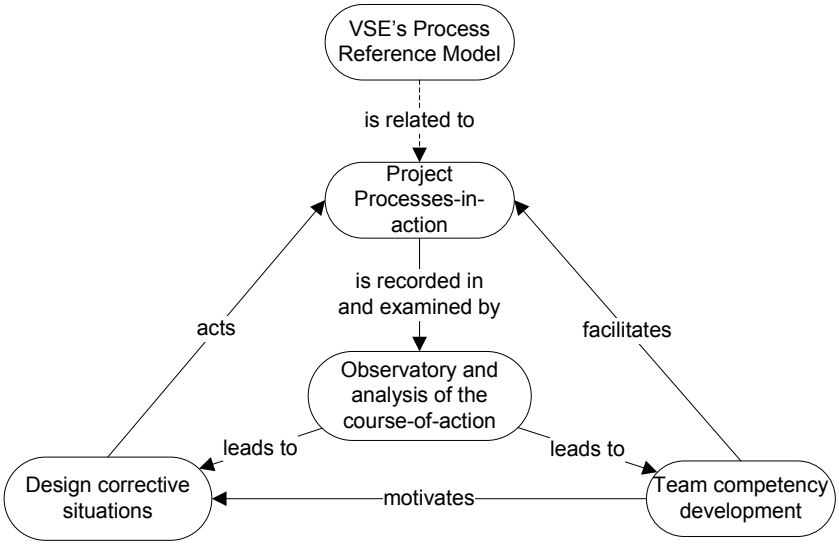

Fig. 1. The project's observable activities are self-recorded by team members. The analysis of the project-in-action provides a decomposition of the global dynamic in terms of smaller units and the relations of sequencing and embedding between these units. Two benefits are expected: (i) a reflective consciousness of competency maturity level; (ii) a support to design corrective actions. Both consequences may improve and facilitate the project processes.

3 Related Work

The "course-of-action" research framework [6] consists in several empirical and technological research programs in various domains (work analysis [4], traffic control [5], sport [8], and music composition [21]). The work described in this paper uses plentifully results of these research programs.

It would be impossible to reference all the research work that has been inseminated by Argyris and Schön's theories [10]. In the software engineering field, Halloran [11] investigates the relationship between a software process assessment and improvement model and organizational learning. This work points out the difference between "engineer's espoused theory" and his/her "theory in use" but it does not develop this matter as we did and rather focuses on the use of organizational learning to promote a proactive approach culturally to continuous improvement and learning procedures.

Many propositions have been made for Process Improvement or Process Assessment in small software companies ([12], [13], [14]). Many small organizations are unaware of existing SPI& SPA standards and assumes that assessments conformant to these standards can be expensive and time consuming, difficult to perform in small companies. We think that while building the observatory of course-of-action, foundations are set-up that will facilitate further SPI & SPA programs. There are similitude with the SPA process proposed in [l] based on an initial self-evaluation and following structured interviews and the observatory as we use it.

4 Observing Software Activities

4.1 Software Engineering Standards

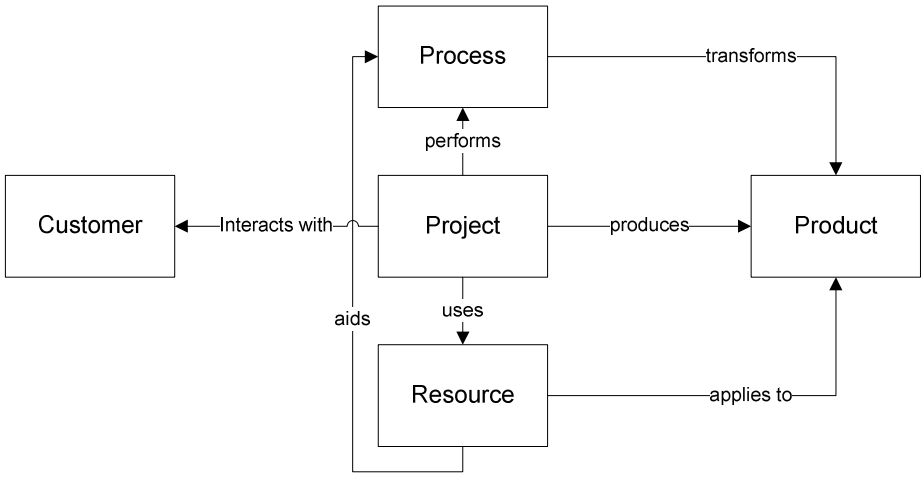

Fig. 2. The objects of software engineering, suggesting a categorization of standards in the subject areas of customer, process, product, and resource [15].

A very concise definition of the objects of software engineering is "a project uses resources in performing processes to produce products for a customer [15]." It gives a model in figure 2, centered on the software engineering project as the focal point for

applying software engineering standards. This suggests a categorization of standards in four major areas: customer, process, product, and resource.

For VSEs, each category contains a number of standards that put them out of reach. There is a need for an umbrella standard within each category. The IEEE/IEC 12207, Software Life Cycle Processes [16], provides this umbrella for all of the customer and process standards. An on-going initiative of ISO should provide lifecycle profiles for Very Small Enterprises (VSEs) [7].

4.2 VSEs Faced to the 12207

Confronted to the 12207, a software engineer in a VSE is at a loss ([1]"like a goose finding a knife" as French people say). First, this standard has received major changes since 1995: Amendment 1 in 2002, Amendment 2 in 2004, and a complete revision in 2008. Secondly, there are currently 43 processes in the 12207:2008 [16], organized in 7 process groups. As an example of the gap with the VSEs needs, the emerging standard "Software Engineering - Lifecycle Profiles for Very Small Enterprises (VSE)" [7] contains 2 processes: Project Management (PM.1) and Software Implementation (SD.1). PM.1 is subdivided in 4 sub-processes (Project Planning, Project Plan Execution, Project Assessment and Control, Project Closure) and SD.1 is subdivided in 6 sub-processes (Software Implementation Initiation, Software Requirements Analysis, Software Architecture and Detailed Design, Software Construction, Software Integration and Tests Product Delivery).

It is not sure that a software engineer in a VSE share the same meaning of these 10 names of sub-processes (from Project Planning to Software Integration and Tests Product Delivery) with a client or a colleague of a major company engaged in any SPI program such as ISO/IEC 15504 or CMMI. However, they will try to communicate and may sign a contract, but they don't speak about the same things. This lack of understanding illustrates the existence of two theories of action – for a software engineer as for any practitioner -, as defined by Argyris and Schön. They have established a distinction between those theories that are implicit in what we do as practitioners and managers (theories-in-use), and those on which we call to speak of our actions to others (espoused theory). "When someone is asked how he would behave under certain circumstances, the answer he usually gives is his espoused theory of action for that situation. This is the theory of action to which he gives allegiance, and which, upon request, he communicates to others. However, the theory that actually governs his actions is this theory-in-use [10]." We may ask question about the extent to which theory-in-use fits espoused theory. Reflection may be a help to discover the theory-in-use and to reveal the nature of the 'fit'. We believe that the observatory of course-of-action – adapted to the software engineering field – may support this process.

4.3 What Can Be Observed?

This significant activity for the actors includes action and communication, but also other elements: interpretations, feelings, judgments, …The data necessary to study the course of action must include continuous observations of the behavior of action and communication in a work situation as well as different kinds of instigated verbalizations from the actors which would provide access to other elements [4].

Software development never uses a repeated scheme and it may be difficult to interrupt a software engineer at work and to provoke a verbalization of what he/she is doing and why. In §2.3 we gave an overview of what, in the observable activity is (i) presentable, (ii) accountable and (iii) commentable by the actor.

Products and documentary resources are main objects of (i) presentation as they describe the inputs and outputs of the activity. The "historical" context of resources' use and products' production has to be recorded too. This can be described in terms of events and processes, involving occurrences of agents (people) and artifacts (products and resources) meeting in space (in case of distributed cooperation) and time. As a first stage, we may consider individual courses of action of the various participants. At a second level, a collective action involves parts of several individual courses of action which take place synchronically or sequentially. We need to divide individual course-of-action in smaller units, that we call course-of-action unit. Each event of interest has to be (ii) accounted in an instance of Course-of-action Unit in relation with people and artifacts involved. It provides a kind of project journal. A journal may be seen as a kind of reflective practice that is a device for working with events and experiences in order to write (iii) comments and extract meaning from them.

5 A Case Study

5.1 Introduction

In spring 2007, local employers in Brest decided to implement a recent French law on professional training. This law requires that 3% of employees be under 'sandwich' (or work placement) conditions. A lot of companies choose to use a system called "Contrat de professionnalisation" (professionalization contract) over a period of 12 months. During these 12 months, the full-paid employee is attending university for certain periods. For contracts involving our computing department, we dedicated an innovative program called "Software Engineering by Immersion" ('Ingénierie du Logiciel par Immersion'). The main feature of this last year of the Masters programme is to learn software engineering by doing, without any computing course but with a long-term project as the foundation of all apprenticeships. Alternating employees are attending university in 9 periods of 2 consecutive weeks and work in team of 6 in order to build a complete information system.

The program's rhythm is based on the lifecycle of a project organized into stages. Each stage was arbitrary sized to 2 weeks due to the constraints of alternation. The

cycle is: Stage 0: Warm-up; Stage 1: Project set-up; Stage 2: Requirement capture; Stage 3: Requirement analysis; Stage 4: Design; Stage 5: Software construction; Stage 6: Software construction; Stage 7: Integration and Verification; Stage 8: Qualification and Deployment.

This case study is based on the activity of a team of 6 young software engineers (the six formers authors) accompanied with the two latter authors acting as participants-to-observe: one having a direct contact of the team members, sharing their environment and taking part in the activities of the team, the other one conducting reviews and formal assessments as they happen. This case study depicts some aspects of the building and the filling of the course-of-action observatory.

The whole observatory is supported with several electronic tools such as semantic wikis, content management system and dedicated applications. Semantic wikis offers a lightweight authoring plate-form and will be used to record most events of the day-to-day life in the project journal.

5.2 The Horizon of Software Engineering Standard

As told in section 4.1, the 12207:2008 standard acts as a standard umbrella and was used during the introductory stage to define the framework of a software engineer's activity. The 12207:2008 was preferred to CMMI because the former (used jointly with the 15504 standard [17]) separates processes and capability levels in two dimensions while CMMI handles them in one dimension. This separation was preferred because it defines processes "(set of interrelated or interacting activities which transforms inputs into outputs" [16]) independently from base practices ("an activity that, when consistently performed, contributes to achieving a specific process purpose [17]").

The 43 processes are too many and complex to be used as the reference model and we concentrate on 16, those related to the software development cycle, that is: 6.2.2 Infrastructure Management, 6.3.1 Project Planning, 6.3.2 Project Assessment and Control, 6.4.1 Stakeholder Requirements Definition, 6.4.4 Implementation Process replaced by 7.1.1 Software (SW) Implementation Process and its 6 sub-processes, 7.2.1 SW Documentation Management, 7.2.2 SW Configuration Management, 7.2.3 SW Quality Assurance, 7.2.4 and 7.2.5 SW Verification & Validation, 6.4.7 SW Installation. Processes are grouped into process groups (five 12207 group processes are concerned that we regrouped in three).

The 6 young engineers chosen for this case study have a Bachelor in Information Technology (4-year studies in the field) and they work in large companies with a structured corporate baseline. However, there is a need for a common reference of the terms used, either because they have different significations in the different companies, or because their signification is unknown or fuzzy. We choose to use the ISO/IEC FCD 24765, "Systems and software engineering – Vocabulary [18]".

We dispose of a PDF version of the 12207:2008, licensed by ISO and of a electronic version of the 24765, copyrighted by ISO but free of use as long as the copyright is cited. As the project goes along and its events are recorded in the project journal, and in order to facilitate links between the project journal and Software

Engineering standards used at the horizon, the whole team filled two semantic wikis with a subset of the two standard used :

- the 12207 wiki (http://oysterz.univ-brest.fr/12207) is an hypertext reference of the ISO/IEC 12207:2008 for the process level : title, purpose, list of outcomes and process decomposition in activities and tasks;
- the 24765 wiki (http://oysterz.univ-brest.fr/24765) is a subset of the ISO/IEC 24765 vocabulary, it is actually under reengineering but on-line SEVOCAB is provided by ISO (http://pascal.computer.org/sev_display).

The structure of these two semantic wikis is given in figure 4.

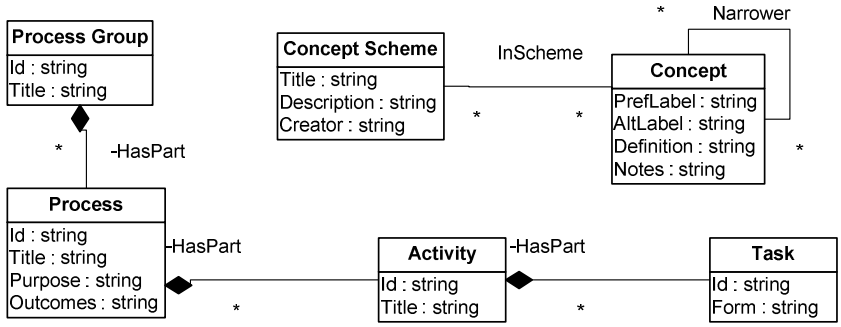

Fig. 4. A model of 12207 and 24765 semantic wikis.

5.3 The Project in Action

The two latter authors both worked for nearly ten years at Thales Information System (formerly Syseca Inc), a software services company. They led projects and developed several management information systems under the control of Thales Information System corporate baseline.

The authors have defined an apprenticeship/production framework called ILI (*Ingénierie du Logiciel par Immersion*, Software Engineering by Immersion), based on a reference model, a development cycle and a typical WBS (Working Breakdown Structure: a deliverable-oriented hierarchical decomposition of the work to be executed by the project team to accomplish the project objectives and create the required deliverables. It organizes and defines the total scope of the project [18]).

The Process Reference Model (PRM) is adapted and simplified from ISO/IEC 12207; we are using 3 *process groups* organizing 13 processes: *Software Development Engineering* (Requirements capture, Software Requirements Analysis, Software Architectural Design , Software Detailed Design, Software Construction, Software integration; Software qualification testing); *Software Project Management* (Project Management, Quality Assurance, Configuration Management); and *Software Development Support* (Infrastructure Management, Life Cycle Model Management, Documentation Management, Installation-Operation).

We use a Y-shaped life cycle that separates resolution of technical issues from resolution of feature issues [19]. First, the cycle is divided into two branches (tracks): a functional track and a technical track. Then these two tracks amalgamate for the realization of the system.

The WBS has a structural and a temporal decomposition. Each process is structurally decomposed in Software Engineering activities (to distinguish it from the activities in the 12207 sense) that may have slightly variation from a project to another. Each Software Engineering activity is further decomposed in sub-activities that can be fully specified or just named, depending of the scope and goals of the project. The WBS is temporally organized in stages (in our case, 9 of 2 week each). The planning of each stage is divided in several work scenes that carry on SE activities. Scenes will be performed by team members and ought to produce artifacts.

The course-of-action forms a whole that is concerned with all aspects described in previous paragraphs but we need to divide the continuous development of the course of action into significant units (cf. §2.3). We decide to divide the whole course-of-action by replying to the question: "What is this about, from the point of view of the engineer?" This division is recorded through the central event Course-of-action Unit. Complex or collective interactions require an intermediate level, called Step-of-action sequencing and embedding Course-of-action units. Links with PRM are provided.

A picture of all these interlinked concerns is given in figure 4.

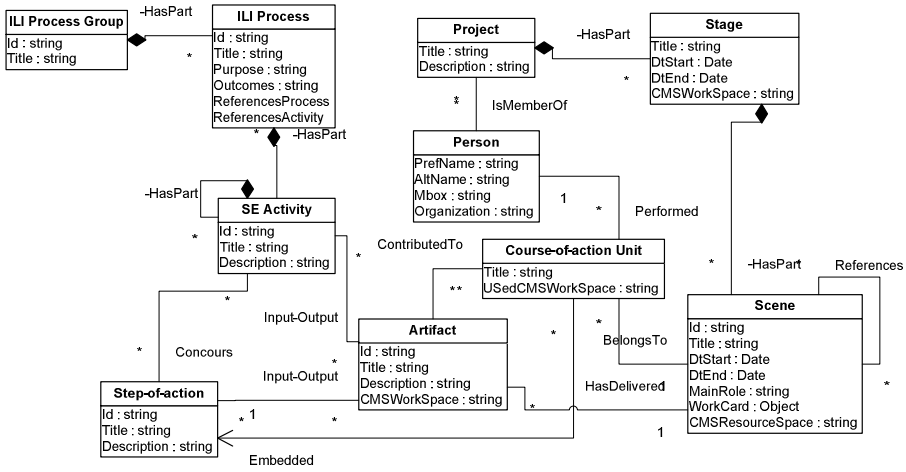

Fig. 4. A model of Process Reference Model -PRM- (on the left) and WBS (on the right). Artifacts are shared between PRM and WBS. The Course-of-action Unit is used as central link. Steps-in-action characterize the relations of sequencing and embedding between these units.

The project journal uses a semantic in order to record the progress of the project. The project manager initially fills and updates the WBS of his/her project. Team members can record events as they happen but have to systematically fill the wiki at the end of each phase. Semantic wiki is the most flexible tool in order to record and shape a structured content. Properties (modifying the underlying data model) can be added, updated or deleted as the project goes along. Information (data) can be recorded in a bulk mode and the typesetting performed later. Things to do or to report

are created in one Wiki word to indicate that they have to be filled. Information can be temporary missing or incomplete.

5.4 Recording Assessments

Several kinds of assessment occur in the life of a project. Assessment may be focused on products or services, on processes or on persons. Assessment itself provides information on action performed but many other elements significant for the actors and the course-of-action analysis: interpretations, feelings, judgments, actors' commitment to the situation and their use of past experience in the course-of- action.

Recording project assessment. The project has to record artifacts produced by project progress: lecture notes, progress meeting report, peer review reports which constitute valuable inputs for further analysis.

Recording project assessment. We argue that personal capability determination (rather than process capability determination) is more suitable to VSEs because employees may perceive it as a valuable benefit. Using the 2-level structure of our Process Reference Model (on the left part of figure 4), we analyze carefully SE activities in order to define abilities mobilized (or competencies: "the ability of a person to act in a pertinent way in a given situation in order to achieve specific purposes [20]"). For each process, we defined a family of competencies constituted with a list of knowledge topics and a set of abilities or skills required to perform the process (see an example in table 1).

Table 1. An example of a competency family: "Software detailed design'.

Knowledge topics	Abilities or skills
Software Design Fundamentals : concepts and principles, design role in a development cycle, top-level and detailed design	To use design methods and tools (in relation with requirements) to produce design documents: system and software architecture and detailed design
Software decomposition configuration item, software component, software unit	To implement methods and modeling tools of various aspects of a system (architecture and decomposition software, data structure)
Software architecture through different views: conceptual, dynamic, physical, data.	To implement J2EE development and technology of associated framework
UML diagrams to describe static and dynamic views	To implement DBMS concepts, techniques and tools
Object-oriented design	

We believe that a first step in competency development should be made by the engineer him/herself through a self-assessment of abilities at a maturity level. The assessment scale grows from 1 to 5: - 1: Smog - 2: Notion - 3: User - 4: Autonomous - 5: Expert. Each young engineer is required to periodically fill the 13 competency families while auto-analyzing the tasks performed and him/her achievement level

with the abilities defined in the family. This periodic inventory is supported by eCompas, a tool intended to manage development, assessment and value-added of competencies over the course of a curriculum or a professional career.

The eCompas tool is intended to store artifacts that may be interesting to illustrate the ability determination. Each time a software engineer self-assesses a process's ability level, he/she has to write an entry associated with the process and may link this entry with artifacts stored. It constitutes a rudimentary portfolio, but sufficient for our purposes. This tool needs to be reengineered to work with the wikis' architecture.

5.7 Focus on a Process: the Design Process

Recording the project in action. According to ISO/IEC 12207, outcomes of the 7.1.3 Architectural Design and 7.1.4 Software Detailed Design Processes are: a) a software architectural design is developed and baselined that describes the software items that will implement the software requirements; b) internal and external interfaces of each software item are defined; c) consistency and traceability are established between software requirements and software design and d) a detailed design of each software component, describing the software units to be built, is developed.

For the Design Process, 12207 recommended tasks and 15504 base practices are roughly the same:

1) transformation of the requirements for the software item into an architecture that describes its top-level structure and identifies the software components.
2) development and documentation of a top-level design for the interfaces external to the software item and between the software components of the software item.
3) development and documentation of a top-level design for the database.
4) development and documentation of preliminary versions of user documentation.
5) definition and documentation of preliminary test requirements and the schedule for Software Integration.

Our ILI framework, considered as representative of VSEs processes, decompose the Design Process in 3 SE Activities: Adjusting the Design, Exemplary Software Design, and Software Design (including Database Design as a sub-activity).

If we have a look at the information recorded in the observatory by team members, they performed two kinds of self-confrontations. The structure of self-confrontations of the former kind, performed at the end of the task, reflects the structure of recommended tasks as they may be found in the SE Activity description. For instance, for the Exemplary Software Design Activity, the description stresses the identification of Computer Software Components, the requirements allocation to the components and the components specification. So, each participant to this activity recorded its own participation in a Course-of-action unit kept to the Activity description. The latter kind of self-confrontation was performed as team members prepared the Software Design Process Review, a formal review. They have to create a synthetic description of the Design Process and to record it in its associated Work Scenes (see figure 4). Participants created Steps-in-action embedding individual Course-of-action units and established inter-wikis links with the corresponding 12207 Processes. It is

not sure that the 12207 outcomes and tasks were confronted to the performed actions, but it indicates an attempt to link the course-of-action at the horizon of SE standards.

Recording team competency development. Periodic inventories of team members are recorded within the eCompas tool. A copy (in a Word format) is stored into the observatory. Focusing on the Design Process, we may note that a team member has participated to the 3 SE Activities defined for the Design Process (see above). As the year started, he assesses himself at the maturity level - 1 - (or - none) for the process as a whole and for each associated abilities. Inside his company, he acts as a software developer and has very little opportunity to improve design skills. After the Software Design Process Review (6^{th} stage), he assesses himself to a maturity level of 4 - Autonomous - (level 2 - Notions - was reached at the end of the 3^{rd} stage, and level 3 - User – after the Exemplary Design Activity). The availability of accurate competency level provides valuable information for the project manager in order to assign tasks to team members.

Recording other assessments. The most valuable information is provided with the meeting report. They are recorded using a semantic wiki through a semantic form. Links to other resources (person, artifact, process ...) are very easy to establish and to update. It provides an ordering scheme and new navigation features.

6 Conclusion and Perspectives

We proposed to adapt the course-of-action framework to software engineers' activity in Very Small Enterprises (VSEs). An observatory collects the data necessary to study the course of action therefore including continuous observations of the behavior of action and communication in a work situation as well as different kinds of instigated verbalizations (transcript in a written form) from the actors which would provide access to other elements such as interpretations, feelings, judgments. As a case study, the activity of a team of 6 young software engineers accompanied with two participants-to-observe is currently recorded in the observatory. As units of courses of action are significant units for the actor, we choose to breakdown the whole course-of-action in units based on individual performed activities.

A further study will use these data to proceed with the analysis of course-of-action, using a theoretical framework, described as semio-logical. This framework will make possible to explain the global dynamics - or composition - of the courses of action units, their local dynamics - or generation - and the linkage between these two dynamics.

The current state of this work – the building and the filling of an observatory of the part of the agent's observable activity that is pre-reflexive (i.e. presentable, accountable and commentable) – let suggest that analysis will lead (1) to specify the modalities of engineers' interaction with life cycle processes leading to the design of better interaction or of corrective situations and (2) to contradict or support the reconstruction by the engineer of his/her own activity, i.e. going from "pre-reflective consciousness" of the actor towards a reflective attitude.

Thus, we may think that observing and analyzing software engineer's activity help to reveal his/her theory-in-use [10] - what governs engineers' behavior and tends to be tacit structures - that we may call Project Processes-in-use in a VSE. The unit breakdown of course-of-action is based on performed activities related to a simple Process Reference Model issued from the ISO/IEC 12207:2008 standard. We made the hypothesis that this standard constitutes the "espoused theory" of software engineers. So, the course-of-action framework may help engineers to establish a link between his/her "Project Processes-in-use" and "espoused Process Reference Model" and contribute to reduce the fit between a project-in-action and SE standards. When the upcoming standard "Software Engineering - Lifecycle Profiles for Very Small Enterprises (VSE)" [7] will be available, we will consider how this standard fits in this proposition.

Argyris and Schön explored the nature of organizational learning and defined two kind of learning: simple-loop learning and double-loop learning [22]. Then they set up two models (Model I and Model II) that describe features of theories-in-use that either inhibit or enhance double-loop learning. Further work is required to consider how course-of-action analysis is related with these organizational learning models and hence, on the VSE's ability to cope with innovations and changes.

References

1. Schön, D.: The Reflective Practitioner. Basic Books, New York (1983)
2. Argyris, C., Putnam, R., McLain Smith, D: Action Science, Concepts, methods, and skills for research and intervention. Jossey-Bass,San Francisco (1985)
3. Software Engineering - Lifecycle Profiles for Very Small Enterprises (VSE) -- Part 1 http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=51150
4. Theureau, J., Filippi, G., Gaillard, I.: From semio-logical analysis to design: the case of traffic control, communication. In Colloquium "Work activity in the perspective of organization and design", M.S.H., Paris (1992)
5. Theureau, J., Filippi, G.: Analysing cooperative work in an urban traffic control room for the design of a coordination support system, chapter 4. In: Luff, P., Hindmarsh, J., Heath, C. (eds.) Workplace studies, Cambridge Univ. Press, 68--91 (2000)
6. Theureau, J.: Course-of-action analysis & course-of-action centered design. In: Hollnagel E. (ed.), Handbook of Cognitive Task Design, Lawrence Erlbaum Ass., New Haven (2003)
7. Ribaud, V., Saliou, P.: Revealing Software Engineering Theory-in-Use through the Observation of Software Engineering Apprentices' Course-of-action. In: 4th International Multi-Conference on Computing in the Global Information Technology, IEEE Press, New York (2009)
8. Theureau, J.: Selfconfrontation interview as a component of an empirical and technological research programme. In : II° Journées internationales des sciences du sport, Paris (2002)
9. Varela, F.: Principles of biological autonomy. Elsevier, New York , (1980)
10. Argyris, C., Schön, D.: Theory in practice: Increasing professional effectiveness. Jossey-Bass, San Fransisco (1974)
11. Halloran, P.: Organisational Learning from the Perspective of a Software Process Assessment & Improvement Program. In: 32nd Hawaii International Conference on System Sciences. IEEE Press, New York (1999)
12. Cater-Steel, A.P.: Process improvement in four small software companies. Software Engineering Conference, 262-272, IEEE Press, New York (1999)

13. Grunbacher, P.: A software assessment process for small software enterprises. Euromicro 97. 'New Frontiers of Information Technology', 123-128, IEEE Press, New York (1997)
14. von Wangenheim, C.G., Anacleto, A., Salviano, C.F.: Helping small companies assess software processes. IEEE Software, 23, 91-98 IEEE Press, New York (2006)
15. Moore, J.W.: An integrated collection of software engineering standards. IEEE Software, 16, 6, pp. 51-57 IEEE Press, New York (1999)
16. ISO/IEC 12207:2008, “Information technology -- Software life cycle processes”. International Organization for Standardization (ISO), Geneva (2008)
17. ISO/IEC 15504:2004, “Information technology -- Process assessment”. International Organization for Standardization (ISO), Geneva (2004)
18 ISO/IEC FCD 24765, “Systems and software engineering – Vocabulary”. International Organization for Standardization (ISO), Geneva (2009)
19. Roques, P., Vallée, F. : UML en action. Eyrolles, Paris (2002)
20. Meirieu, P. : Si la compétence n’existait pas, il faudrait l’inventer In IUFM de Paris Collège des CPE, 2005, http://cpe.paris.iufm.fr/spip.php?article1150 (2007)
21. Donin, N., Theureau, J.: Music composition in the wild: from the horizon of creative cognition to the time & situation of inquiry. In: EACE 05, Crète, 57-64 (2005)
22. Argyris, C., Schön, D.: Organizational learning: A theory of action perspective. Addison Wesley, Reading, Mass. (1978)