Towards Experience Management for Very Small Entities

Vincent Ribaud, Philippe Saliou, Claude Laporte

Towards Experience Management for Very Small Entities

Vincent Ribaud, Philippe Saliou
Département d'Informatique
Université de Bretagne Occidentale, UEB
20 avenue Le Gorgeu, CS 93837
29238 Brest Cedex, France
{Vincent.Ribaud, Philippe.Saliou}@univ-brest.fr

Claude Y. Laporte
Département de génie logiciel et des TI
École de technologie supérieure
1100 rue Notre-Dame Ouest
Montréal (Québec), Canada, H3C 1K3
Claude Y. Laporte@etsmtl.ca

***Abstract*—The ISO/IEC 29110 standard: Lifecycle profiles for Very Small Entities, provides several Process Reference Models applicable to the vast majority of very small entities (defined by the ISO as "an entity (enterprise, organization, department or project) having up to 25 people") which do not develop critical software and share typical situational factors. An ISO/IEC 29110 pilot project has been established between the Software Engineering group at Brest University and a 14-employee company with the aim of establishing an engineering discipline for a new Web-based project. As the project proceeded, it became apparent that setting up the ISO/IEC 29110 standard has to be performed in two steps: 1) provide self-training materials to the VSE employees on this new standard; and 2) support good practices with a simple Experience Management system which is compatible with the ISO/IEC 29110 standard. This paper reports the lessons learned about training from the pilot project, and addresses the research issues associated with the Experience Management system.**

Keywords: software engineering processes, experience management, very small entities, ISO standards.

I. INTRODUCTION

This paper is an extended and enhanced version of a paper presented at the ICSEA 2010 conference [1].

Very Small Entities (VSEs) are recognized as extremely important to the software economy, producing stand-alone or integrated software components for large software systems. While the use of Software Engineering standards may promote recognized and valuable engineering practices, these standards were not designed with the needs and expertise of VSEs in mind, and do not fit the characteristics of VSEs. They are consequently difficult to apply in these settings [2]. The term 'Very Small Entity' (VSE) was defined by ISO/IEC JTC1/SC7 Working Group 24 (WG24) as "*an entity (enterprise, organization, department or project) having up to 25 people*." This definition has subsequently been adopted by the ISO in their response to the specific needs of VSEs: the ISO/IEC 29110 standard, Lifecycle profiles for Very Small Entities [3]. The standard defines a group of Standardized Profiles. Profiles are subsets of appropriate elements of standards which are relevant to the VSE context; for example, processes and products of the main software engineering standards. The ISO/IEC 29110-4-1 Basic Profile [4] applies specifically to a VSE involved in the development of a single software application by a single project team with no special risk involved and no particular situational factors at play.

This paper reports some of the conclusions reached as a result of a pilot project the authors conducted with a 14-person VSE that builds and sells counting systems for tracking visits to public and private sites. Only 3 of the employees are software developers, however, and so the VSE asked for assistance with software processes – mainly managing requirements and establishing a disciplined test process. ISO/IEC 29110 was naturally chosen as the reference framework, and the aim of the pilot project was to set up, within the VSE, the part of the Basic Profile related to requirements.

A VSE claiming compliance with the ISO/IEC 29110-4-1 Basic Profile will implement and use all the profile elements, as identified in Clause 7 of the profile specification [3]. The profile elements concerning requirements are: Project Plan Execution (PM.2), and Project Assessment and Control (PM.3), producing the Change Request work product; and Software Requirements Analysis (SI.2), producing a work product Change Request and Requirement Specification.

These profile elements state what has to be done, but provide very little guidance on how to do it. For the latter, Deployment Packages (DP) are expected to be particularly helpful. A DP is a set of artifacts developed to facilitate the implementation of a set of practices of the ISO/IEC 29110 standard. We introduced the ISO/IEC 29110 materials related to requirements to the VSE, and began to coach a novice engineer on the use of these materials for managing requirements. As the pilot project proceeded, it became apparent that the ISO 29110 set of documents (including DPs) was not up to the task of sustaining this VSE in its engineering activities. We maintain in this paper that implementing standardized software engineering activities in a VSE requires specific operational materials and mechanisms. What we are proposing is to provide VSE employees with a Self-Training Package intended to help the engineer carry out these activities.

Because software engineers in a VSE use SE processes and produce SE products continuously in different projects, we expected that an Experience Management (EM) system tailored for a VSE would provide a way to relate and integrate those project experiences and be a significant help. We also maintain in this paper that an EM system for a VSE should be constructed on a framework suitable for that entity,

but derived from a standardized Process Reference Model (presented in part in Section III.C) taken from the ISO/IEC 29110 Basic Profile [4].

EM solutions to organizing knowledge can be supported by experience factories (EF) [5]. "*EM includes methods, techniques, and tools for identifying, collecting, documenting, packaging, storing, generalizing, reusing, adapting, and evaluating experience; and for the development, improvement, and execution of all knowledge-related processes*" [6]. EF is defined as "*an infrastructure designed to support experience management*" and "*supports the collection, preprocessing, and dissemination of experiences*" [6]. This paper outlines a simple knowledge management system intended to gather, link, and reuse knowledge about SE activities. Requirement Analysis and its associated work products will be used as an example.

Professional competency management focuses on the development of a professional attitude and skills. These components are usually addressed in a 'practicum' or in 'clinical work', and the concept of reflection, inspired by D. Schön's work [7], is central to this competency development. The knowledge management system was designed based on two main guiding principles: the extraction of knowledge of existing SE standards – providing the system with a bootstrap, and the building of new knowledge by the software engineers themselves – a process required to maintain accurate and 'living' knowledge.

The next section provides an overview of the ISO/IEC 29110 standard, EM and EF, and related work. The standard is discussed in section III, and a case study introduced focused on requirement analysis and test activities. In section IV, we present our work on EM for a VSE and discuss some facts related to the case study. We conclude the paper by briefly presenting a few perspectives.

II. Requirements and Related Work

In this section, we present the ISO/IEC 29110 initiative, discuss about the Experience Factory, report on Argyris and Schön's theories of action, and overview D. Schön's reflection-on-action work. Related work is also discussed.

A. SE Standards for Very Small Entities

1) ISO terminology

A Base Standard is an approved International Standard or ITU-T Recommendation [8]. An International Standardized Profile (ISP) is a harmonized document on which there is international agreement, and which describes one or more profiles [8]. A Profile is a base standard or set of base standards and/or ISs, including, where applicable, the selected classes, conforming subsets, options, and parameters of those base standards, or ISPs, required to accomplish a particular function [8]. A Technical Report (TR) is developed like a standard, but its purpose is simply to provide technical information, rather than requirements on implementation. Also, TRs are available free of charge.

2) ISO initiative

SE standards and methods often neglect the needs and problems of small and medium-sized enterprises (SMEs), which constitute a major part of the software industry. In 2005, the ISO recognized the needs and problems of VSEs and established a Working Group (WG24) mandated to develop a set of standards and technical reports suitable for these entities. The resulting ISO/IEC 29110 standard constitutes a set of guidelines for use by VSEs. Those guidelines are based on subsets of appropriate elements of standards, referred to as VSE profiles [3], relevant to the VSE context; for example, processes and outcomes of ISO/IEC 12207 [9], and products of ISO/IEC 15289 [10].

The Generic Profile Group targets VSEs that do not develop critical software and that share typical situational factors. It is composed of 4 profiles: Entry, Basic, Intermediate, and Advanced. As mentioned in the introduction, the Basic Profile [4] applies to VSEs involved in the development of a single software application by a single project team with no special risk involved or situational factors at play. By design, it excludes many of the ISO/IEC 12207 processes.

The standard is composed of five parts. As specified in [3], Part 1 targets VSEs, assessors, standards producers, tool vendors, and methodology vendors. Part 3 targets assessors and VSEs, and Parts 2 and 4 target standards producers, tool vendors, and methodology vendors. Parts 2 and 4 are not intended for VSEs. Part 5 targets VSEs. If a new profile is needed, only Parts 4 and 5 can be developed without impacting existing documents, and they would become Part 4-x and Part 5-x-y respectively, through the ISO/IEC standardization process.

The simplest path for a VSE is to start with Part 5-1-2: Management and engineering guide: Generic profile group: Basic profile. Using the Guide, a VSE can benefit in the following ways [11]:

- An agreed set of project requirements and expected products is delivered to the customer;
- A disciplined management process, which provides project visibility and proposes corrective actions for project problems and deviations, is performed;
- A systematic software implementation process, which satisfies customer needs and ensures quality products, is followed.

3) Deployment Packages

Once ISO/IEC TR 29110-5-1-2 has been downloaded, at no cost, from the ISO website, a VSE may consider that the help provided in it is insufficient to guide the implementation. Deployment Packages (DPs), by contrast, can be expected to provide significant help, a DP being defined as "*a set of artifacts developed to facilitate the implementation of a set of practices, of the selected framework, in a VSE*" [12]. The elements of a typical DP are: process description (activities, inputs, outputs, and roles), a guide, a template, a checklist, an example, presentation material, references and mapping to standards and models, and a list of tools [12]. The mapping is given only as information to show that a DP has explicit links to standards, such as ISO/IEC 12207, or to models, such as the CMMI®. So, by deploying and implementing the package, a VSE can visualize the concrete steps required to achieve or demonstrate coverage. Packages are designed so that a VSE

can implement their content without having to implement the complete framework at the same time.

4) Pilot projects

Pilot projects are an important means for reducing risks and learning more about the organizational and technical issues associated with the deployment of SE practices. A successful pilot project is also an effective means for encouraging the adoption of new practices by members of a VSE [12]. DPs are intended to apply the ISO/IEC 29110 standard in a VSE. Tailoring software processes to a VSE constitutes a kind of process improvement. A pilot project may also be an initial implementation of a DP, which provides WG24 with feedback from the improvement proposals before the DP is adopted as a standard.

B. Experience Management

1) Knowldege and Experience

The main asset of a software company is its intellectual capital, and knowledge management (KM) aims to capitalize on that capital. Schneider has explained how knowledge can be encoded in different representations and stored in ontologies, and that the instances of an ontology make up a knowledge base which can be searched and used for reference purposes [13, p. 135]. Business issues of KM are related to decreasing time and cost while increasing quality, and to making better decisions [14]. But KM implementation requires investment, and is probably out of reach for a VSE. Experiences constitute a subset of knowledge, and the reuse of experiences is a variant of KM. Experience management (EM) is a lightweight KM approach, a possible implementation of which is presented in the next section.

2) Experience Factory

EM is aimed at improving project performance by leveraging experiences from previous projects. In order to achieve experience reuse, Basili et al. [15] have proposed an organizational framework that separates project-specific activities from reuse packaging activities, with process models to support each activity.

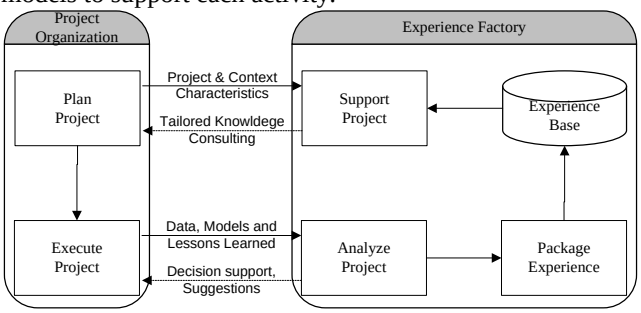

Figure 1. Experience factory (adapted from Ras et al. [6]).

The framework, represented in Figure 1, defines two separate organizations: a project organization, intended to deliver the system required by the customer, and an Experience Factory (EF), the role of which is to monitor and analyze project developments, to develop and package experience for reuse in the form of knowledge, processes, tools, and products, and to supply them to the project organization upon request [15]. "*The EF employs several methods to package the experience, including designing measures of various software process and product characteristics and then build[ing] models of these characteristics that describe their behavior in different contexts*" [16, p. 30]. A dedicated sub organization is required for learning, packaging, and storing experience.

Separating the project from the experience organization, physically or logically, may relieve project teams of the tasks required by EM, but is not, in our opinion, applicable in VSEs, or even in software SMEs (up to 250 employees). We agree with [17] that software SMEs need a more lightweight means of creating these knowledge bases with minimal overhead. Wiki-based repositories are often used as knowledge repositories, because the wiki concept easily integrates users into the knowledge-sharing process in SMEs [18]. Lightweight tools are useful, but knowledge transfer processes have to be built because the goal of EF is to build knowledge by learning from experience. We draft a learning theory in the next section.

C. Argyris and Schön's Theories

1) Theory of Action

According to Argyris and Schön, people design and guide their behavior by using theories of action. They suggest that there are two kinds of theory of action: a theory consistent with what people say, and a theory consistent with what people do. "*Espoused theories of action are the theories that people report are governing their actions. Theories-in-use are the theories of action that actually govern their actions*" [19, p. 7].

Argyris and Schön used three constructions to explain theories-in-use (see Figure 2 for a more comprehensive explanation). *Governing variables* are values that a person is trying to keep within a preferred range (e.g. a manageable workload). *Action strategies* are strategies used to maintain the governing variables within the accepted limits (e.g. refusal to accept extra work). These strategies will have *consequences* which are either intended (e.g. the amount of work does not increase too much) or unintended (e.g. the amount of work is decreasing too drastically).

When there is a mismatch between intended consequences and outcomes, the situation has to be corrected. Argyris states: "*An organization may be said to learn to the extent that it identifies and corrects errors*" [19, p. 4]. They suggest that the first response to this mismatch is to select another action strategy that will still satisfy the governing variables (e.g. accept extra work, but delay providing the result). Such a process of changing the action strategy only, and not the governing variables themselves is called *single-loop learning*. Another possibility is to examine and modify the governing variables (e.g. accept too great a workload in order to reach a new position). In this case, both the governing variables and the action strategy have to be modified, and this is called *double-loop learning*.

Argyris and Schön argue that, although espoused theories vary widely, theories-in-use do not. They labeled the most prevalent theory-in-use 'Model I'. "*Model I theories-in-use are theories of top-down, unilateral control of others for the actors to win, not to lose, and to control the environment in*

which they exist to be effective" [19, p. 7]. They argue that, with such a theory-in-use, problem solving works for issues that do not require that the underlying assumptions of Model I be questioned (*single-loop learning*). Model II theories-in-use make it possible for people "*to have problem-solving skills that question the governing values of their theory-in-use*" [19, p. 7] (*double-loop learning*).

Models of theory-in-use

Model I and Model II theories-in-use consider three elements: (1) *governing variables*, which are values that actors seek to maintain [1], each of which can be thought of as a continuum with a preferred range (e.g. not too concerned, but not too indifferent either); (2) *action strategies*, which are sequences of moves and plans adopted by actors in particular situations to satisfy governing variables [1] to keep those variables within the preferred range (e.g. undertaking physical exercise to reduce stress); and (3) the *consequences* that follow as a result of action, which can be intended – those that the actor believes will satisfy the governing variables (e.g. feeling better after engaging in a sport), or unintended, both types designed to be dependent on the theories-in-use of the recipients, as well as those of the actors.

Single and double-loop learning

When the consequences of an action strategy are what the actor wanted, then that person's theory-in-use is confirmed. If there is a mismatch between intention and outcome, the consequences are unintended. Argyris defines learning as the detection and correction of error. The first response to error is to search for another action strategy. "*Single-loop learning occurs when errors are corrected without altering the underlying governing variables*" [2, p. 206]. An alternative is to question the governing variables themselves, to subject them to critical scrutiny (e.g. to openly investigate the anxiety, rather than trying to suppress it). "*Double-loop learning occurs when errors are corrected by changing the governing variables and then the actions*" [2, p. 206]. Argyris and Schön argue that many people espouse double-loop learning, but are unaware of it, much less able to produce it.

Model I and Model II

Briefly, Model I is composed of four governing variables: (1) achieve the purpose as defined by the actor; (2) win, not lose; (3) suppress negative feelings; and (4) emphasize rationality [1]. The primary behavioral strategies are to control the relevant environment and tasks unilaterally, and to protect oneself and others unilaterally. Thus, the most widely used action strategy is unilateral control over others. Characteristic ways of implementing this strategy are: to make non illustrated attributions and evaluations (e.g. "Your work is poor."); to advocate courses of action in ways that discourage inquiry (e.g. "Surprise me, but don't take risks."); and to treat one's own views as obviously correct, leaving potentially embarrassing facts unstated [1]. The consequences are likely to be defensiveness, misunderstanding, and self-fulfilling processes [2]. Model I leads to low-level learning, and double-loop learning tends not to occur. Argyris and Schön aim to move people from a Model I theory-in-use to a Model II theory-in-use that fosters double-loop learning.

The governing variables of Model II include: (1) valid information; (2) free and informed choice; and (3) commitment: vigilant monitoring of the implementation choice to detect and correct errors [2]. The behavioral strategies involve sharing control with those who have the competence to do so and who participate in designing or implementing the action [1]. As in Model I, prominent behaviors are advocated, evaluated, and attributed. Unlike Model I behaviors, Model II behaviors stem from action strategies where attributions and evaluations are illustrated with observable data, and the surfacing of conflicting views is encouraged so that they can be publicly tested. The consequences include minimally defensive interpersonal and group relationships, great freedom of choice, and a high level of risk-taking. Defensive routines are minimized and genuine learning is facilitated [1, 2].

References

[1] C. Argyris, R. Putnam, and D. McLain Smith, "Action Science: Concepts, Methods, and Skills for Research and Intervention," San Francisco: Jossey-Bass, 1985, pp. 4, 80.

[2] C. Argyris, "Double-Loop Learning, Teaching and Research," Learning & Education, vol. 1 (2), Dec. 2002, pp. 206-219

Figure 2. Theory of Action, according to Chris Argyris and Donald Schön.

2) The reflective practitioner

Schön's "reflective practitioner" perspective [7, p. 20] guides creative professionals to reflect about their creations during (*reflection-in-action*) and after (*reflection-on-action*), thereby completing the creative process. A specialist is a professional practitioner who is used to dealing with certain types of situations again and again. Practitioners build up a collection of ideas, examples, situations, and actions which Schön calls a "repertoire". "*A practitioner's repertoire includes the whole of his experience insofar as it is accessible to him for understanding and action*" [7, p. 138].

A practitioner develops a repertoire of expectations, images, and techniques. As long as her/his practice continues to present the same types of cases, s/he becomes less and less susceptible to surprise [5, p. 60]. But, when a new situation is stimulating enough, the reflective practitioner is surprised. Schön argues that these experienced professionals deal with the 'messiness' of practice not only by consulting the research knowledge base, but by engaging in what he calls "reflection-in-action" [20], which is sometimes described as 'thinking on our feet'.

In many EF implementations, effort is put into analyzing and packaging experiences from raw experiences. But further effort is required to change the way that the whole organization performs its work. "*An organization adopting the EF approach must believe that exploiting prior experience is the best way to solve problems and ensure that the development process incorporates seeking and using this experience*" [16, p. 30]. A parallel can be drawn between an organization using an EF and a reflective practitioner. Raw and packaged experiences play the role of the practitioner's repertoire. As long as his/her practice remains stable, the practitioner relies on her/his tacit knowing-in-action, which is built on previous experiences directly. An organization building new knowledge while analyzing and packaging raw experiences is similar to Schön's reflection-on-action. "*Practitioners do reflect on their knowing-in-practice […], they think back on a project they have undertaken […], and they explore the understandings they have brought to their handling of the case. They may do this in a mood of ideal speculation, or in a deliberate effort to prepare themselves for future cases*" [7, p. 61]. An organization using the EF assumes that the EF and the Project Organization are integrated. "*The activities by which the Experience Factory extracts experience and then provides it to projects are well integrated into the activities by which the Project Organization performs its function*" [16, p. 31]. It assumes that the Project Organization makes no special effort to reuse packaged experiences.

But a practitioner may also reflect on a practice while s/he is performing it (Schön's reflection-in-action). In this case, the possible objects of this reflection are varied. "*He may reflect on the tacit norms and appreciations which underlie a judgment, or on the strategies and theories implicit in a pattern of behavior. He may reflect […] on the way in which he has framed the problem he is trying to solve*" [7, p. 62]. Of course, anyone can encounter a situation where a rule drawn from previous experience cannot be applied, in which case he/she has to be engaged in what

Schön calls a "practitioner's reflective conversation" with the materials related to the situation. Occasions will also arise when none of the packaged experiences will help the Project Organization using EF. Although Schön did not explicitly establish links between the reflection concepts and the nature of organizational learning presented in Figure 2, we consider reflection-in-action as a kind of double-loop learning, and we assume that developing a reflective practice will favor that type of learning.

D. Application to Very Small Projects

The set of ISO/IEC 29110 documents establishes what has to be done in a software project, and will be presented in section III.A. Little help is provided to explain the procedure, however, although pilot projects are carried out expressly intended to put this standard into practice which can be considered as an experience packaging activity. Section III.B and section III.C provide an insightful presentation of a pilot project on requirements.

The main deliverables of a completed SME project are stored by the project manager in an Experience Repository, according to a fixed storage scheme. At best, the Experience Repository of a VSE contains only raw experiences. Data, models, and deliverables collected on previous projects are stored as is, without any structure. The initial benefit for a VSE of using the ISO/IEC 29110 standard is that a common Process Reference Model (PRM) will be shared between projects, as the structure of the PRM may help to structure the Experience Repository. Since the most common knowledge pattern transfer is the copy-paste model, a shared structure will favor reuse of raw experiences. Section IV.A presents an Experience Repository for raw data, and section IV.B.1 presents the copy-paste activities that are using it.

We believe that there can be a considerable gap between the structure of an engineer's repertoire (and hence the project organization that he or she may use) and the structure of the EF. Extracted knowledge facilitates experience reuse and learning. In Figure 1, adapted from [6], arrows from left (Project Organization) to right (Experience Factory) indicate knowledge extraction. Knowledge transfer is a double-loop learning activity. Lessons learned from the pilot project indicate that it is a difficult issue for engineers to cope with, especially novice engineers. Our challenge was to find a way to encourage reflection-in-action and develop double-loop learning. Sections IV.3.2 and IV.3.3 present the practical solution that we provided to the VSE.

Ras et al. [21] address this problem with an approach called 'learning space generation', which enriches experience packages with additional information from specifications provided either by the instructor or by the student. The learning space is presented by means of Wiki pages within a specialized Wiki based on the Software Organization Platform (SOP). Our approach does the opposite. Rather than providing engineers with access to the experience packages, we essentially provide task description exemplars and product exemplars created in small projects.

III. A Standardized Process Profile for VSEs

In this section, we present the context of the pilot project, the expectations of the VSE, and the application of the ISO/IEC 29110 standard to this project.

A. Implementation of Standardized Processes

At the core of the ISO/IEC 29110 standard is a Management and engineering guide (ISO/IEC 29110-5) [11], which focuses on Project Management and Software Implementation, and an Assessment Guide (ISO/IEC TR 29110-3) [22]. ISO/IEC 29110-5 provides a practical guide to the ISO/IEC 29110-4-1 standard [4], identified as a Basic Profile of the Generic profile group. For instance, the starting point for ISO/IEC 29110 use for requirements is the SI.2 Software Requirements Analysis activity, its list of tasks (SI.2.1 to SI.2.7), and the associated roles.

Deployment Packages (DP) provide VSEs with assistance in adopting standards through a DP Repository http://profs.logti.etsmtl.ca/claporte/English/VSE/index.html. For instance, DP Software Requirement Analysis [23] simplifies task decomposition and provides a step-by-step method for each task.

B. Pilot Project

1) Requirements

Fenton et al. state in [24]: "*For 25 years, software researchers have proposed improving software development and maintenance with new practices whose effectiveness is rarely, if ever, backed up by hard evidence.*" They suggest several ways to address this problem, in particular careful design and measurement experiments, such as pilot projects.

2) Context of the VSE

A VSE with a staff of 14 (3 of them software engineers) requested the help of our SE group in the spring of 2009. The VSE designs, builds, develops, and sells counting systems designed to collect and analyze data on visits to public or private sites. Initially intended for counting pedestrians, this VSE's products now cover bikes, horses, and cars. Counting systems are based on stand-alone counter boxes (including sensors, a power supply, data storage, and data exchange) and a software chain capable of collecting, analyzing, presenting, and reporting counting data. In the previous software chain, the set of data was downloaded from counters by infrared link or GSM, stored on personal computers, and then transmitted via a file transfer utility.

3) The new software project

Because of its clients' requirements and the products supplied by the competition, the VSE began a complete reconstruction of its software chain in order to transform it into a Web-based system, called Eco-Visio, intended to host the data of fleets of counting systems for each client, and capable of processing statistics and generating analytical reports on counting. At the end of June 2009, the VSE hired a graduate of Brest University, who had done his final internship at the VSE. At the same time, we visited the VSE and initiated a pilot project with the intention of transferring a part of the ISO/IEC 29110 standard to the specific context of the VSE. Project stakeholders decided to focus on two SE

activities: 1) the establishment of a practical technique for gathering and managing requirements; and 2) improvement of the system's reliability with a disciplined test process.

The new software project, completed at the end of March 2010, was released as the first version of the new Eco-Visio Web-based system.

C. Basic Profile

1) Basic Profile processes

The Generic Profile Group [4] is a collection of four profiles (Entry, Basic Intermediate, Advanced), providing a progressive approach to satisfying the needs of a vast majority of VSEs that do not develop critical software and share characteristic situational features. The Basic Profile applies to a VSE that is involved in software development of a single application by a single project team involving no special risk or situational factors. The objective of the project is to fulfill an external or internal contract. The internal contract between the project team and their client need not be explicit.

The Basic Profile is made up of two processes: Project Management (PM) and Software Implementation (SI). A process is defined as "*a set of interrelated or interacting activities which transforms inputs into outputs*" [9]. Table I provides the process/activity breakdown, and presents tasks related to requirements and tests (which are the focus of the pilot project cited above).

TABLE I. BASIC PROFILE PROCESS BREAKDOWN

Process	Activities	Pilot project-related tasks
PM Project Management	PM.1 Project Planning	PM.1.1
	PM.2 Project Plan Execution	PM.1.13, PM.1.14
	PM.3 Project Assessment and Control	PM.2.2 and PM.2.4
	PM.4 Project Closure	PM.3.5
SI Software Implementation	SI.1 SW Implementation Initiation	-
	SI.2 SW Requirements Analysis	SI.2.2, SI.2.3, SI.2.4
	SI.3 SW Architectural and Detailed Design	SI.3.5, SI.3.6
	SI.4 SW Construction	SI.4.4
	SI.5 SW Integration and Tests	SI.5.4
	SI.6 Product Delivery	-

ISO/IEC TR 29110-5-1-2 [11] is intended to guide the Basic Profile implementation of PM and SI processes described in ISO/IEC 29110-4-1 [4]. These processes integrate practices based on the selection of ISO/IEC 12207 SW life cycle processes and ISO/IEC 15289 information product (documentation) standards elements. DPs will facilitate the implementation of these processes.

2) Basic Profile products

Clause 9 of ISO/IEC 29110-4-1 [4] establishes the normative list of Basic Profile work product and deliverable specifications. There are 23 work products, which can be the input, output, or internal products of processes, activities, or tasks.

3) Process assessment

ISO TR 29110-3 [22] is an Assessment Guide applicable to all VSE profiles. It is compatible with ISO/IEC 15504-2 and ISO/IEC 15504-3. The assessment has two purposes: 1) to evaluate process capability based on a two-dimensional assessment model (from the ISO/IEC 15504:2006 standard [25]); and 2) to determine whether or not an organization achieves the targeted VSE Profile based on the process capabilities evaluated [22]. A VSE-specific Process Assessment Model (PAM) can be derived by selecting only a set of assessment indicators from ISO/IEC 15504-5: "*an Exemplar PAM"* We selected the assessment indicators relevant to the corresponding process outcomes, as defined in ISO/IEC 29110-4-1.

4) Performing the ISO/IEC 29110 Requirements Analysis

ISO/IEC 29110-4-1 provides a set of cohesive tasks for each activity. Also established here are the VSE needs and suggested competencies. For instance, it sets out the SI.O2 objective: "*Software requirements are defined, analyzed for correctness and testability, approved by the Customer, baselined and communicated. Changes to them are evaluated for cost, schedule and technical impact previously to be processed*" [4, p. 8].

ISO/IEC TR 29110-5-1-2 details the tasks to be performed for each PM and SI process activity: role, description of the task, and input and output products. For instance, it defines tasks SI.2.1 to SI.2.7, detailed in Table II, and their associated output products: *Requirements Specification*, *Verification Results*, *Change Request*, *Validation Results*, and *Software User Documentation*.

The Software Requirements Analysis DP [23] simplifies task decomposition: requirement identification, requirement refinement and analysis, requirement verification and validation, and requirement change management. A step-by-step method is described for each of these four tasks. The DP also provides a Software Requirement Specification template. Training materials and an Excel-based traceability tool can be downloaded from the publicly accessible WG24 website.

The pilot project was intended to provide coaching for the implementation of the Software Requirements Analysis DP. One VSE novice engineer studied the DP and was given a short training course, using the training material associated with this DP. Despite all this helpful material, the VSE engineer was not able to start the Software Requirements Analysis activity, suffering from 'blank page' syndrome. The authors could not provide strong support to the VSE, and we have had to reorientate the pilot project.

5) Problem analysis

Based on feedback that the novice engineer was not able to perform the SI.2 Software Requirements Analysis activity, the authors set about to analyze the problem.

As we stated in section III.B, action theory studies what an actor does in a given situation in order to achieve objectives. Argyris and Schön [26] made a distinction between *espoused* theories, which are those that an individual claims to follow, and *theories-in-use*, which are those that can be inferred from action. Espoused theory and

theory-in-use may be contradictory, and the agent may or may not be aware of any inconsistency. By definition, however, the agent is aware of espoused theory, and theories-in-use can be made explicit by reflection-on-action [27].

Software companies use SE and software quality standards as the foundation of their quality assurance process and quality management system. Since these companies claim to follow and respect standards, we may think that these standards constitute a part of the espoused theories of software engineers, especially Process Assessment and Process Reference Models. In the software field, we observe that a software engineer may have a work behavior – her/his theories-in-use – which often runs counter to the organizations' processes, practices, and procedures that she/he is supposed to follow and talk about, i.e. espoused theories.

What happened to that young engineer? Through the standard documentation and DPs, he received a great deal of information on espoused theory. However, as his repertoire of experience (and VSE Experience Repository) was all but empty, he could not act in accordance with any theory-in-use.

An Experience Repository may act as a product and project memory. It records footprints of the organization's theories-in-use and provides support for learning from past experience. Thus, managing experience in a repository may provide VSE engineers with a simple form of knowledge management. But, as we will see in the next section, an Experience Repository requires additional processes in order to support knowledge transfer.

IV. Experience Management for the VSE

Chan and Chao present a research survey conducted among 68 SMEs which have implemented Knowledge Management (KM) initiatives [28]. SMEs are significantly bigger than the targeted VSEs, but the lessons learned in this survey also apply to VSEs. Effective KM is influenced by two types of KM capability: infrastructure and process, which have to be deployed. This section presents a simple Content Management System-based infrastructure to manage experience and some activities that may be part of EM processes.

A. An Experience Repository

1) Related work

A significant part of EM in a software company should be about software documentation reuse (code reuse is outside the scope of this paper). So, the primary inputs of our system are documentation deliverables: plans, requirements, design specifications, data schemas, test cases, and so forth. Publishing and content management systems (CMS) are generally used as the basis for a documentation management infrastructure. But several authors have criticized the rigidity of the editorial control required by a CMS [29] and the need to balance structure/constraint and flexibility [30]. Some promote the use of Wikis and RDFs (Resource Description Frameworks) to resolve these issues [31].

Wikis are probably a suitable tool for facilitating collaborative design and development, and may be viewed as part of the project repository (see Figure 1), but requirements for an EM infrastructure are different. Rech et al. identified several challenges related to knowledge transfer and management processes for SMEs in the software sector: recording, reusing, locating, and sharing information [18]. The authors evaluated a small software enterprise and a micro software enterprise with reuse policies in place. They point out that the engineers have little confidence in knowledge transfer, because only a few reusable documents have been created. They also note that the workflow for reusing knowledge is slow and typically demotivating, because multiple sources have to be searched manually and documents belonging together weren't grouped together or linked [18]. As we will see in the next section, a CMS-based system with a simple and fixed structure may resolve most of these issues.

2) Experience Repository infrastructure

According to Peter Senge [32], what he calls "personal mastery models" and "mental models" are two of the five disciplines that distinguish a learning organization from more traditional organizations. The questions to be answered are: how do experts learn compared to novice practitioners, and how do their mental models differ? "*People with a high level of personal mastery live in a continual learning mode*" [32, p. 142]. Mental models are "*deeply ingrained assumptions, generalizations, or even pictures and images that influence how we understand the world and how we take action*" [32, p. 8]. They are similar to Schön's professional's repertoire. "*From a cognitive point of view, there is a quantitative difference between expert and novice knowledge bases and also a qualitative difference, e.g. the way in which knowledge is organized. Novices lack background knowledge and are not able to connect their experience to their knowledge base. The organization of knowledge at the experience provider's and at the experience consumer's makes the transfer of knowledge between different levels of expertise extremely difficult*" [33]. Part of the problem can be avoided if experts and novices share a common repertoire structure. We use the ISO/IEC 29110 Basic Profile Process Breakdown (see Table I) as the shared structure of the Experience Repository. We discuss later how learning processes should be developed in order to support knowledge transfer.

Managing an Experience Repository for a small project can be greatly facilitated if the structure is kept as simple as possible, which means we should also avoid amassing too many artifacts. Our proposal is that, whenever a project is completed, the project closure activity create its own space in the CMS and use the Process/Activity decomposition of ISO/IEC 29110-4-1 [4, Clause 7] as the structure for that space. Then, only the main deliverables of the project, as defined in ISO/IEC TR 29110-5-1-2 [11, Clause 4], will be stored, and in the right place in the structure.

Table II shows the structure and content of the Experience Repository for some representative activities of each process. To further illustrate our work, we added the activity-related tasks. The left-hand column provides links to

the ISO/IEC 12207 activities profiled in the parts of ISO/IEC 29110 mentioned. We added (in italics) a proposal (published in [34]) for the management of deliverables related to support activities.

The infrastructure is not intended to be a project repository (the left part of Figure 1) hosting project deliverables in different versions. The infrastructure forms part of an Experience Repository intended to record the final state of the project and to provide further projects with exemplars of deliverables. The Alfresco content platform (http://www.alfresco.com/products/wcm) is used as a Web

TABLE II. STRUCTURE AND CONTENT OF THE EXPERIENCE REPOSITORY

12207	Activity	Tasks	Output products
Project Management Process			
6.3.1.3.3, 6.3.2.3.1, 6.3.2.3.2	PM.2 Project Plan Execution	• PM.2.1 Review Project Plan • PM.2.2 Change request analysis • PM.2.3 External revision meeting • PM.2.4 Internal revision meeting	Project Plan Change Request Meeting Record
…			
Software Implementation Process			
6.4.1.3.1, 6.4.1.3.2, 6.4.1.3.3, 6.4.1.3.4, 6.4.1.3.5, 7.1.2.3.1	SI.2 SW Require-ments Analysis	• SI.2.2 Document requirements • SI.2.3 and 2.4 V & V requirements	Requirements Specifications V&V Results
7.1.3.3.1, 7.1.4.3.1	SI.3 SW architectural and detailed design	• SI.3.3 Document software design • SI.3.4 Software design verification	Software Design Traceability Record Verification Results
…			
Management and Implementation Support Process			
6.2.1.3.1, 6.2.1.3.3	*Method and tool support*	• *Process establishment* • *Process improvement* • *Tool support*	*Process implementation recommendations* *Tool usage guide*

content management suite, mainly for providing an upload-download system organized into a hierarchy of space, with the possibility of a fine-grained control of users' rights over spaces. As mentioned above, the space hierarchy structure is, for each project, the Process/Activity decomposition of ISO/IEC 29110-4-1 [4]. Each space hosts a variety of work products of ISO/IEC TR 29110-5-1-2 [11]. Examples of these products are given in Table II, column 4.

B. KM Support Processes

According to [35], knowledge can be created through dedicated acquisition, conversion, application, and protection of knowledge assets. In the survey of 68 SMEs by Chan and Chao [28], most of the respondents stated that they encounter knowledge capture problems related to time, place, and people. But Conradi maintains that the hard part is not the "upward externalizing direction", but the "downward, internalizing flow" [36]. The problems that arise are very similar to those encountered in software engineering reuse. The literature agrees that understanding is a high cost factor for reuse. Dusink and Van Katwijk state: "*Essential for a higher degree of reuse is the reusing engineer's understanding of the reusable artifacts, the process, and the actions to be taken*" [37]. Ras states that general reuse education and technology training are the two principal solutions proposed to enhance reuse with respect to understanding [33].

The standard method for transferring knowledge from experts to novices is the copy-paste model, and the VSE asked for a similar pattern, which is for the Experience Repository to work based on this model, and for us to seed the Experience Repository by providing them with suitable examples, such as a Software Requirements Specification, that they can reproduce as closely as possible. The VSE asked for immediate working solutions and did not want to invest in understanding the experiences stored. However, we decided to provide the VSE with two levels of Experience Management Process: a copy-paste level, which is presented in this section, and a continuous understanding level, as discussed in section IV.C.

1) Copy-paste activities

TABLE III. SI.2 SOFTWARE REQUIREMENTS ANALYSIS -- TASKS AND ROLES. THE ASTERISK MEANS 'IF APPROPRIATE'.

Task List	Role
SI.2.1 Assign tasks to the Work Team members in accordance with their role, based on the current *Project Plan*.	Technical Leader, Work Team
SI.2.2 Document or update the *Requirements Specification*.	ANalyst, CUStomer
SI.2.3 Verify the *Requirements Specification*.	AN
SI.2.4 Validate the *Requirements Specification*	CUS, AN
SI.2.5 Document the preliminary version of the *Software User Documentation* or update the present manual.*	AN
SI.2.6 Verify the *Software User Documentation*	AN
SI.2.7 Incorporate the *Requirements Specification*, and *Software User Documentation** to the *Software Configuration* in the baseline.	TL

The copy-paste process is designed to be as simple as possible. Clauses 4.2.8 and 4.3.8 of ISO/IEC TR 29110-5-1-2 [11] propose task decomposition of the PM and SI processes for each activity (Table III presents the decomposition for SI.2 Software Requirements Analysis), together with inputs and outputs of each task. So, we can establish the workflow for each of the 23 work products (cf. §III.C.2). For instance, Figure 2 presents the workflow of Work Product 11, *Requirements Specification*.

Figure 3. WP11 *Requirements Specification* workflow

It may happen that a work product workflow spans several activities of the same process, such as WP17 *Software User Documentation*, which covers SI.2 to SI.5,

and even PM and SI processes, such as WP8 *Project Plan*, which covers PM.1 to SI.6.

The VSE needs a simple model to locate, store, and retrieve work products, according to the Process Reference Model used. Our proposal is to locate a work product inside the CMS space associated with the last activity that outputs the final version of this work product. So, WP11 *Requirements Specification* will be located in the 'SI.2 SW Requirements Analysis' space, WP17 *Software User Documentation* will be located in the 'SI.5 SW Integration and Tests' space, and WP8 *Project Plan* will be located in the 'SI.6 Product Delivery' space.

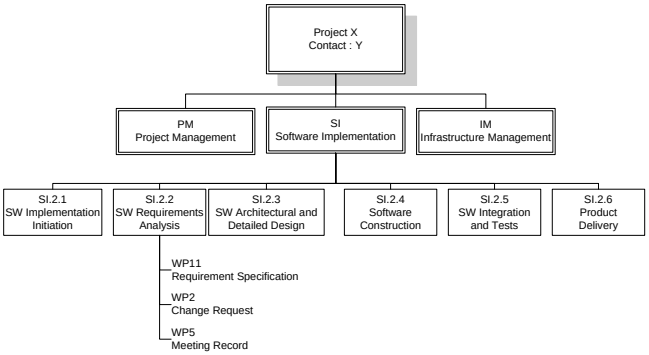

Figure 4. Structure and content of a project in the Experience Repository.

The task of storing work products in an Experience Repository space associated with the project will be allocated to the PM.4 Project Closure activity, which is the responsibility of the Project Manager (PM) role. With this simple copy-paste EM process, the PM role stores artifacts, and (ideally) every VSE employee can access and copy the artifacts of his/her choice. An extract from the structure and content of a project space is given in Figure 4.

2) Learning software engineering processes

A common assumption in software process improvement is that "*the quality of a software product is largely governed by the quality of the process used to develop and maintain it*" [38, p. 8]. Based on this assumption, Conradi states: "*This often means that relevant work practices (processes) must be systematically documented as formal routines, often as standard process models. These routines must then be communicated to the developers, customized and adopted by them and later revised based on experience and overall strategies*" [39, p. 268].

The ISO/IEC 29110 standard provides a Process Reference Model and DPs aimed at guiding the implementation of this model. We were not confident in the ability of novice engineers to understand the work practices and associated DPs documented in the ISO/IEC 29110 standard. So, we scheduled a training week on ISO/IEC 29110 Software Requirements Analysis in December 2009. Ten novice engineers (including the VSE engineer) attended the session, which comprised a course on requirements and a case study using the Software Requirement Analysis DP [23]. This DP is summarized in Figure 5.

Task 1. Requirements identification. The objective is to clearly define the scope of the project and identify key requirements of the system. Steps are: (i) Collect information about the application domain; (ii) Identify project scope; (iii) Identify and capture requirements; (iv) Structure and prioritize requirements.

Task 2. Requirements refinement and analysis. The objective is to detail and analyze all the requirements identified. Steps are: (i) Detail requirements; (ii) Produce a prototype.

Task 3. Requirements verification & validation. The objective is to verify requirements and obtain validation from the customer or his representative. Steps are: (i) Clarify fuzzy requirements (verification); (ii) Review SRS (Software Requirements Specification); and (iii) Validate requirements.

Task 4. Requirements change management. The objective is to manage requirements change in line with a process agreed upon with the customer. Steps are: (i) Track changes to requirements; (ii) Analyze the impact of changes; (iii) Identify changes that are beyond the project scope; (iv) and Prioritize changes.

Figure 5. Step-by-step path proposed by the DP Requirement Analysis.

The session began with an introductory lecture on requirements, but trainees were quickly plunged into action with the preparation of a peer review on a requirements analysis guide. This guide was issued by a major ISO 9001 software company (at which both authors had been employed for about 10 years). The SW Requirements Specification (SRS) Document was issued by the DOD-STD-2167A software development standards [40]. This guide is intended to facilitate the writing of the SRS. Peer review of this guide provided trainees with their initial exposure to standardized requirements management.

During the second phase of the session, trainees had to contribute to writing a similar guide, based only on the ISO/IEC 29110 standard. The authors provided trainees with a preliminary version of the guide, written in a top-down manner, starting with the ISO/IEC 12207 standard processes devoted to requirements (6.4.1 Stakeholder Requirements Definition, 7.1.2 SW Requirements Analysis) and finishing with the ISO/IEC 29110 Basic Profile SI.2 Software Requirements Analysis activity. Trainees had to incorporate both the Software Requirement Analysis DP and its step-by-step approach into the guide.

Finally, trainees had to apply the enhanced guide to a 'real' SRS and update this SRS to comply with the guide requirements. That SRS is for eCompas – an existing system developed by the second author and former graduate students.

C. Understanding Experience

1) Learning from experience

Despite the path traced in the standard during the training session), some young engineers (and this is true of the VSE engineer in particular) reported being unable to find their way through managing the requirements.

As mentioned above, the ISO/IEC 29110 standard attempts to document the best practices as formal routines. Several authors have studied the gap between the rationalistic, linear model of software engineering and the reality for most small software organizations. Conradi and Dyba carried out a study in the context of a national software process improvement program in Norway for SMEs to assess the attitude to formalized knowledge and experience sources. They found that "*developers are rather skeptical at using written routines, while quality and technical managers are taking this for granted*" [39]. Dyba [41] states that "*a specific challenge involves balancing the refinement of the existing skill base with the experimentation of new ideas to find alternatives that improve on old ideas.*" But our hypothesis is more straightforward: for many novice engineers, the copy-paste model is not sufficient as a knowledge transfer pattern, because they have no previous experience to help them understand the formalized knowledge. The training session provided novice engineers with products resulting from past experience and with the assistance of teachers. We built the content session using a normative curriculum of a professional school, as attributed to Edgar Schein in [42]: "*First teach them the relevant basic science, then teach them the relevant applied science, then give them a practicum in which to practice applying that science to the problems of everyday life.*" The normative curriculum reflects an objectivist view of professional education, often portrayed as the opposite of a constructivist view. "*Objectivist conceptions of learning assume that knowledge can be transferred from teachers or transmitted by technologies and acquired by learners. […] Constructivist conceptions of learning, on the other hand, assume that knowledge is individually constructed and socially co constructed by learners based on their interpretations of experiences in the world*" [43, p. 217]. We do not oppose the notions of objectivism and constructivism. Rather, we believe that they offer different points of view which may be combined to favor learning. An important step in the learning cycle is the activity of reflection. If we provide learners with details of past experience acquired by other people, we have to find a way to help learners reflect on that experience. By reflecting on the experience acquired (by her/himself or others), learners integrate the lessons learned from that experience into their own knowledge structures.

2) Reflection-on-action and reflection-in-action

To meet the challenges of their work, practitioners rely on their repertoire of experience, along with a certain ingenuity acquired during that practice, rather than on knowledge-oriented curricula or formulae learned during their basic education. D. Schön describes this repertoire as follows: "*The practitioner has built up a repertoire of ideas, examples, situations and actions. […] When a practitioner makes sense of a situation he perceives to be unique, he sees it as something already present in his repertoire. To see this site as that one is not to subsume the first under a familiar category or rule. It is […] to see the unfamiliar, unique situation as both similar to and different from the familiar one, without at first being able to say similar or different with respect to what. The familiar situation functions as a precedent, or a metaphor, or […] an exemplar for the unfamiliar one*" [7, p. 138].

In order to help VSE employees understand the VSE Experience Repository, and consequently add to their own repertoire, we have designed practices that may help software engineers become 'reflective' practitioners. These practices are generally borrowed from two streams: industrial – process improvement and product assessment – and Schön's theory of reflection-on-action and reflection-in-action. For instance, bootstrapping an engineer's repertoire for a given activity in SE (e.g. requirements analysis or design) may require an approach based on tailoring an activity before the activity itself is performed. This approach has been presented through the specific case of the design in [44].

Such an approach is generally implemented in two steps: 1) tailoring the activity to acquire a minimal structure of the repertoire through a deductive approach (by writing a guide, for instance); and 2) initializing the repertoire through an inductive approach, with the use of retroengineering, for instance. This approach is a pragmatic answer to the lack of support and training that may be experienced in small projects, where the main effort is concentrated on project management and software development tasks.

3) Self-Training Packages

As reflective practices are performed by the learners themselves, very few interactions with a coach are required. The next step is related to organizing the self-learning process. Our proposal is to organize the engineer's training path through small units of work, called 'self-training tasks'. The description of the task is designed as a theater scene: the scene is the reference context where action takes place; it aims to maintain unity of place, time, and action, and is a site where a situation can occur and where people perform actions (and learn). It also serves as a location for action scenarios, for role distribution, and for mobilizing resources and means. The various components of a scene, along with their linkages, are depicted on a self-training report card. The card structure is standardized:

- Related ISO/IEC 29110 Process/Activity

This reference (for instance, SI/SI.2 SW Requirements Analysis) provides a smooth link to ISO/IEC 29110, and through the profile to ISO/IEC 12207 and ISO/IEC 15504.

- Role

The role (for instance, Analyst) is a brief reference to ISO/IEC 29110.

- Task title and objectives

These are similar to Process Title, Process Purpose, and Process Outcomes, as defined in ISO/IEC 12207.

- Step-by-step guide

This is a comprehensive description of the work to be done, intended to be a practical guide to completing the task.

- Resources

This is the set of required resources. It may include the hosting of technical support (such as Oracle Metalink) that a technology transfer center is able to afford when the cost is out of reach for a VSE.

- Output products

These are generally a methodological survey, a tool usage guide, or an installation manual.

The set of self-training activities that a VSE engineer should perform is incorporated into a Training Package (TP) (analogous to the ISO/IEC 29110 Deployment Package, or DP). Developing the concept of the TP is outside the scope of this paper, but suffice to say that a TP is primarily intended to provide self-training on SE activities, with the supplementary goal of initiating and developing a strategy of capitalizing on this knowledge and transferring it to VSE employees.

4) Empirical evaluation

The VSE engineer in question was provided with two TPs on Requirements at the end of 2009. The first was intended to provide the engineer with a basic maturity level on ISO/IEC 29110 Requirements Management (through the study of an SI.2 activity and a review of a 'real' WP11 *Requirements Specification*), and the second involved performing a Requirements Analysis on a 'real' case. The first package was made up of 3 training scenes and the second of a single one. Each TP was calibrated to a week of self-training. The VSE engineer worked through both packages in January 2010.

Favoring reflection-in-action through TPs is, in our opinion, a kind of software improvement. Although no measurements can be easily defined and performed to confirm this, there is empirical evidence of it in the form of 'customer' satisfaction.

The VSE engineer reported that he was now ready to apply the SI.2 SW Requirements Analysis to the Eco-Visio project. As the specifications were established by a subcontractor, he merely reviewed the existing Requirements Specification and rewrote parts of it in order to verify conformity with the template provided in the DP, Software Requirement Analysis [23]. Once updated, the WP11 *Requirement Specification [Validated]* served as an input to SI.5, SW Integration and Tests. The system has been deployed since the end of March 2010, and load testing and application optimization should soon be completed. Defects will then have to be corrected through a short cycle of SI activities.

As an empirical measure of satisfaction with the approach, the VSE asked for a similar approach for SI.5 SW Integration and Tests. In particular, the VSE wanted assistance in establishing a disciplined Change Request Process. This TP is under construction, and we plan to begin with the Software Testing DP [45] as a basis for the whole TP. Probably because tests occur in many SE activities, this DP is organized in a manner that spans PM and SI tasks, which raises many new questions.

D. Towards a sustainable model for a VSE

1) Packaging experiences

For a VSE, the investment in Knowledge Management may appear to take too much time before benefits appear. Obviously, it will take time before a critical mass of experiences will be available in the Experience Repository. Schneider and Schwinn report several problems they faced to in order to achieve a suitable repository, the Experience Base, at DaimlerChrysler. They pointed out the importance of "*thinking [of] an Experience Base as something that needs to be seeded in order to grow*" [46].

Experience management assumes that all relevant experience can be collected and packaged for reuse. Rus and Lindwall have established that there is a difference between explicit and tacit knowledge. "*Explicit knowledge corresponds to the information and skills that employees can easily communicate and document, such as processes, templates, and data*" [14]. Packaging raw experiences in the Experience Factory produces explicit knowledge. "*Tacit knowledge is personal knowledge that employees gain through experience; this can be hard to express and is largely influenced by their beliefs, perspectives, and values*" [14]. Relying on tacit rather explicit knowledge is the prevailing model in a VSE because it does not require the documentation of knowledge or the packaging of experiences. Komi-Sirviö et al. analyzed the case of a company that failed in several attempts to improve knowledge reuse. The company was looking for a new solution that should have a minimal impact on the software development organization. "*This new approach consisted of a knowledge-capturing project and customer projects. The former gathered knowledge from relevant sources and packaged and provided it to a customer project for reuse on demand*" [47]. The knowledge-capturing project is similar to the analysis organization in the Experience Factory framework, but it does this for the customers' project needs.

2) Packaging experiences in a VSE

The previous section suggests that packaging experiences should be performed outside the software development organization. As reported in section IV.C, we used the pilot project to solve an immediate need of the VSE: a disciplined management of requirements. Because the VSE was aware of their weakness in requirements management, they agreed to invest enough time and effort to change their working process for this point. But packaging the required materials was performed by the authors rather than the VSE.

Our proposal for an EM system for a VSE is a simplified approach of the EF infrastructure presented in Figure 1.

The simplified EF is made up of two separate parts: an Experience Repository, and a Training Package Repository. The Experience Repository contains raw experiences; as stated in sections IV.A.2 and IV.B.1, the project manager has to store the main deliverables of the completed project according to a fixed storage scheme. The use of the Experience Repository relies only on the copy-paste knowledge transfer pattern. No help in understanding the raw experiences is provided. When a project is experiencing difficulties in completing a software engineering activity and no useful materials can be found in the Experience Repository, an external task force has to build a Training Package on the given activity and store it in the Training Package Repository. Then, VSE employees may perform self-training using this Training Package. VSE employees may store feedback in the repository in order to improve the process. Self-training tasks are designed to develop reflective thinking. They are based on past experiences, either from the

VSE or from elsewhere. Self-training packages are not intended to explain the packaged experiences for reuse; the main goal is rather to initialize the engineer's repertoire regarding the problematic software engineering activity. Once the self-training package has been completed, the hypothesis is made that the engineer will be able to return to his/her practice and interact with the problematic situation in such a way that it will lead to his/her success. Decisions, support, and suggestions are built up by the engineer her/himself rather than provided by the packaged experiences. Figure 6 shows all the Infrastructure and Process issues that we have addressed in this section.

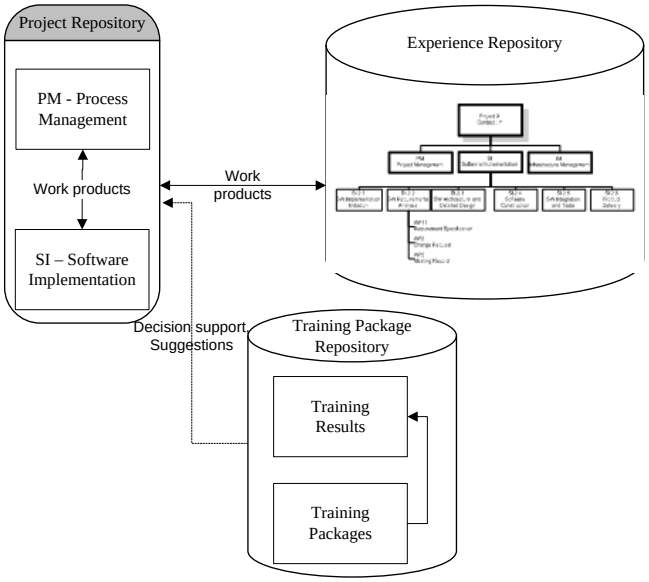

Figure 6. Overview of the EM Infrastructure and Process.

V. CONCLUSION AND FUTURE WORK

We have proposed a simple Experience Management system for a VSE that is compatible with the emerging ISO/IEC 29110 standard. Two hypotheses are posed: (1) the EM infrastructure is kept as simple as possible with the use of a CMS structured with the decomposition of the PM and SI processes; and (2) EM requires dedicated processes that can be taken from D. Schön's reflection-on-action work. The needs of a VSE and the solutions that we have provided are reported as a case study.

Further work is required to consider how the concept of the Training Package could complement that of the Deployment Package.

REFERENCES

[1] V. Ribaud, P. Saliou, and C. Y. Laporte, "Experience Management for Very Small Entities: Improving the Copy-paste Model," in Proc. Fifth International Conference on Software Engineering Advances, New York :IEEE Press, 2010, pp. 311-318.

[2] C. Y. Laporte, "The Development of International Standards for Very Small Entities: Historical Perspectives, Achievements and Way Forward," Joint International Council on Systems Engineering (INCOSE) / Concordia Institute for Information Systems Engineering (CIISE) Distinguished Seminar, 2010.

[3] ISO/IEC TR 29110-1:2011, Software Engineering -- Lifecycle profiles for Very Small Entities (VSEs) -- Part 1: Overview, Geneva: International Organization for Standardization (ISO), 2011.

[4] ISO/IEC 29110-4-1:2011, Software engineering -- Lifecycle profiles for Very Small Entities (VSEs) - Part 4-1: Profile specifications: Generic Profile group, Geneva: International Organization for Standardization (ISO), 2011, available at: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=51154 (last accessed June 1)

[5] V. Basili, G. Caldiera, and D. Rombach, "Experience Factory," in Encyclopedia of SE, vol. 1, Hoboken:Wiley, 2002, pp. 476-96.

[6] E. Ras, R. Carbon., D. Decker, and J. Rech. "Experience Management Wikis for Reflective Practice in Software Capstone Projects," IEEE Transactions on Education, vol. 50 (4), Nov. 2007, pp. 312-320.

[7] D. Schön, The Reflective Practitioner, New York: Basic Books, 1983.

[8] ISO/IEC TR 10000-1:1998, Information technology -- Framework and taxonomy of International Standardized Profiles -- Part 1: General principles and documentation framework, Geneva: International Organization for Standardization (ISO), 1998.

[9] ISO/IEC 12207:2008, Information technology -- Software life cycle processes, Geneva: International Organization for Standardization (ISO), 2008.

[10] ISO/IEC 15289:2006, Systems and Software Engineering -- Content of systems and software life cycle process information products (Documentation), Geneva: International Organization for Standardization (ISO), 2006.

[11] ISO/IEC TR 29110-5-1-2:2011, Software Engineering -- Lifecycle profiles for Very Small Entities (VSEs) -- Part 5-1-2: Management and engineering guide: Generic profile group: Basic profile, Geneva: International Organization for Standardization (ISO), 2011, available at: http://standards.iso.org/ittf/PubliclyAvailableStandards/c051153_ISO_IEC_29110-5-1-2_2011.zip (last accessed June 1)

[12] C. Y. Laporte, Contributions to SE and the Development and Deployment of International SE Standards for Very Small Entities, PhD thesis, Université de Bretagne Occidentale, Brest, 2009, available at: http://tel.archives-ouvertes.fr/tel-00483255/fr/ (last accessed May 25).

[13] K. Schneider, Experiences and Knowledge Management in Software Engineering, Berlin Heidelberg:Springer-Verlag, 2009.

[14] I. Rus and M. Lindvall, "Knowledge Management in Software Engineering," IEEE Software, vol. 19 (3) , May-June, 2002, pp. 26-38.

[15] V. Basili, G. Caldiera, and G. Cantone, "A Reference Architecture for the Component Factory," ACM Transactions on SE and Methodology, vol. 1 (1), January 1992, pp. 53-80.

[16] V. Basili and C. Seaman, "The Experience Factory Organization," IEEE Software, vol. 19 (3) , May-June, 2002, pp. 30-31.

[17] T. Chau and F. Maurer, "A Case Study of a Wiki-based Experience Repository at a Medium-sized Software Company," Proc. ACM K-CAP'05, ACM Press, 2005, pp. 185-186.

[18] J. Rech, C. Bogner, and V. Haas, "Using Wikis to Tackle Reuse in Software Projects," IEEE Software, vol. 24 (6), November-December, 2007, pp. 99-104.

[19] C. Argyris, "Organizational learning and management information systems," ACM SIGMIS Database, vol. 13 (2-3), Winter-Spring 1982, pp. 3-11, ISSN:0095-0033.

[20] D. Schön, Educating the Reflective Practitioner: Toward a New Design for Teaching and Learning in the Professions, San Francisco: Jossey-Bass, 1987.

[21] E. Ras and J. Rech, "Using Wikis to support the Net Generation in improving knowledge acquisition in capstone projects," Journal of Systems and Software, vol. 82, April 2009, pp. 553-562.

[22] ISO/IEC TR 29110-3:2011, Software Engineering -- Lifecycle profiles for Very Small Entities (VSEs) -- Part 3: Assessment guide, Geneva: International Organization for Standardization (ISO), 2011.

[23] S. Alexandre and C. Y. Laporte, "Software Requirement Analysis," http://profs.logti.etsmtl.ca/claporte/VSE/Publications/DP-Software%20Requirements%20Analysis-V1_2.doc, Montréal, 2011 (last accessed May 25).

[24] N. Fenton, S.L. Pfleeger, and R.L.Glass, "Science and substance: A challenge to software engineers," IEEE Software, vol. 11 (4), July 1994, pp. 86-95.

[25] ISO/IEC 15504:2004, Information technology -- Process assessment, Geneva: International Organization for Standardization (ISO), 2004.

[26] C. Argyris and D. Schön, Organizational learning: A theory of action perspective, Reading: Addison Wesley, 1978.

[27] C. Argyris, R. Putnam, and D. McLain Smith, Action Science, Concepts, methods, and skills for research and intervention, San Francisco: Jossey-Bass, 1985.

[28] I. Chan and C. Chao, "Knowledge management in small and medium-sized enterprises," Communications of the ACM, vol. 51 (4), April 2008, pp. 83-88.

[29] J. M. García Alonso, J. J. Berrocal Olmeda, and J. M. Murillo Rodríguez, "Documentation Center – Simplifying the Documentation of Software Projects," Proc. Wiki4SE Workshop – 4th International Symposium on Wikis, Porto, 2008.

[30] J. W. Maxwell, "Using Wiki as a Multi-Mode Publishing Platform," Proc. 25th annual ACM international conference on Design of Communication, ACM, New York, 2001, pp. 196-200.

[31] A. Rauschmayer, "Next-Generation Wikis: What Users Expect; How RDF Helps," Third Semantic Wiki Workshop. at ESWC, Redaktion Sun SITE, Aachen, 2009, poster.

[32] P. M. Senge, The Fifth Discipline. The art and practice of the learning organization, London: Random House, 1990.

[33] E. Ras, "Learning Spaces: Automatic Context-Aware Enrichment of Software Engineeering Experience," PhD Thesis in Experimental Software Engineering no. 29, Stuttgart:Fraunhofer Verlag, 2009.

[34] V. Ribaud, P. Saliou, R. V. O'Connor, and C. Y. Laporte "Software Engineering Support Activities for Very Small Entities," Proc. 17th International Conference on European Systems & Software Process Improvement and Innovation (EuroSPI 2010), Springer-Verlag, September 2010.

[35] A. H. Gold, A. Malhotra, and A. H. Segars, "Knowledge Management: An Organizational Capabilities Perspective," Journal of Management of Information Systems, vol. 18 (1), May 2001, pp. 185-214.

[36] R. Conradi, "From software experience databases to learning organizations," Proc. 11th Int'l Conf. on Software Engineering and Knowledge Engineering (SEKE'99), 1999, Knowledge System Institute, pp. 204-206.

[37] L. Dusink and J. van Katwij, "Reuse dimensions," Proc. Symposium on Software reusability (SSR '95), ACM Press, 1995, pp. 137-149.

[38] M. C. Paulk, C. V. Weber, B. Curtis, and M. B. Chrissis, The Capability Maturity Model for Software: Guidelines for Improving the Software Process, SEI Series in Software Engineering, Addison-Wesley, 1995, 640 p.

[39] R. Conradi and T. Dyba, "An empirical study on the utility of formal routines to transfer knowledge and experience," SIGSOFT Softw. Eng. Notes 26(5), 2001, pp. 268-276.

[40] Department of Defense, Defense System Software Development, DOD-STD-2167A, 29 February, 1998.

[41] T. Dyba, "Improvisation in small software organizations," IEEE Software, 17(5), 2000, pp. 82-87.

[42] D. Schön, "Educating the Reflective Practioner" in Meeting of the American Educational Research Association, http://resources.educ.queensu.ca/ar/schon87.htm, 1987, (last accessed January 25).

[43] D. Jonassen, "Designing Constructivist Learning Environments," in Instructional Design Theories and Models: A New Paradigm of Instructional Theory, Mahwah (New Jersey):Lawrence Eribaum Associates, 1999, pp. 215-240.

[44] P. Saliou and V. Ribaud, "Bootstrapping an empty repertoire of experience: The design case," Proc. 1st Workshop on Human Aspects of SE (OOPSLA 2009), ACM, October 2009.

[45] L. Gómez Arenas, "Deployment Package – Software Testing," http://profs.logti.etsmtl.ca/claporte/VSE/Publications/DP-Software Basic_Profile_Testing-CL00.doc, Montréal, 2010 (last accessed May 25).

[46] K. Schneider and T. Schwinn, "Maturing Experience Base Concepts at DaimlerChrysler," Software Process Improvement and Practice, vol. 6, 2001, pp. 85-96.

[47] S. Komi-Sirvio, A. Mantyniemi, and V. Seppanen, "Toward a practical solution for capturing knowledge for software projects," IEEE Software, 19(3), May/June 2002, pp.60-62.