View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Sapientia

Computer Science and Information Systems 13(3):901–925 DOI: 10.2298/CSIS160802035M

Process and Project Alignment Methodology: A Case-Study based Analysis

Paula Ventura Martins[1] and Alberto Rodrigues da Silva[2]

[1] Research Centre of Spatial and Organizational Dynamics,
Universidade do Algarve, Campus de Gambelas,
8005-139 Faro, Portugal
pventura@ualg.pt
[2] INESC-ID, Instituto Superior Técnico,
Universidade de Lisboa,
1049-001 Lisboa, Portugal
alberto.silva@tecnico.ulisboa.pt

Abstract. Process descriptions represent high-level plans and do not contain information necessary for concrete software development projects. Processes that are unrelated to daily practices or that are hardly mapped to project practices, cause misalignments between processes and projects. We argue that software processes should emerge and evolve collaboratively within an organization. With this propose, this article describes the ProPAM methodology and explores the details of its static view. We also present a case study to validate effectiveness of the proposed methodology. The aim of the case study was to analyse the effects of using ProPAM in a IT organization.

Keywords: Software Process Improvement, Process Management, Project Management, ProPAM.

1. Introduction

Software process improvement (SPI) is a challenge to organizations that have tried to continually improve their software quality and productivity and to keep up their competitiveness [1]. Organizations tend to react to changes in the environment that they operate, changes at a corporate level, unplanned situations not considered in the model, or improve the quality of their final products. Such changes may be caused, for example, by poor team performance, by new tools acquired by the company to support its software development teams, changes in the marketing strategy or in customers' expectations and requirements. Thus, an existing process model must be modified or extended to reflect the evolution of the environment and/or internal changes. However, existing process models – that mostly take into account descriptive aspects, such as work related activities and technical work products – couldn't address such features. Several surveys and studies [2-4] have emphasized that the majority of small and very small IT organisations are not adopting SPI standard models such as CMMI [5] or ISO/IEC 15504 [6]. Another case is observed in Brazil where software industry and universities are working cooperatively in implementing a successful SPI strategy that

take into account software engineering best practices and aligned to Brazilian software organizations context [7].

We argue that the emphasis in SPI should be stressed on communication, coordination, and collaboration within and among project teams in daily project activities, and consequently the effort in process improvement should be minimized and performed as natural as possible. Little attention had been paid to the effective implementation of SPI models, which has resulted in limited success for many SPI programs. SPI managers want guidance on how to implement SPI activities, rather than what SPI activities do actually implement. Limited research has been carried out in exploring new approaches to implement effectively SPI programs. However, to bridge this gap some initiatives have emerged, such as MIGME-RRC methodology [8], on this basis, we propose a new methodology to describe and improve software processes for IT organizations with intensive projects experience. Table 1 presents key terms in the domain of SPI.

Table 1. List of key terms

Key term	Description
SPI model	Basic philosophy for a disciplined, cyclical approach to software process improvement
SPI standard model	SPI model proposed by an international (or national) organizational like ISO, CMMI or OMG
SPI initiative	Work performed by investigators (or research group)
SPI program	Action plan to be taken within the organization that intends to improve their software process
SPI manager	Professional with a wide range of knowledge topics on improving software processes. Responsible for leading SPI programs
SPI activity	Practice in the context of a SPI program

In this paper we propose a SPI methodology called "Process and Project Alignment Methodology" (ProPAM). The main goal is to develop a SPI methodology that can evolve with project's knowledge and consequent improvement at software development process level. ProPAM takes into consideration projects and organization's views and intends to integrate the best practices used in real projects to derive a customized and organization-specific SPI. ProPAM is independent of the technologies, tools and concrete software development processes that could be adopted by different organizations or even in multiple projects of the same organization. For instance, organizations should decide which industry standard shall ultimately prevail for a concrete process; or stakeholders shall decide which modelling language to use, e.g. Unified Modelling Language (UML) [9], Business Process Modelling Notation (BPMN) [10], Archimate [11] or other modelling language.

ProPAM is grounded from personal experience and observations in real organizations, and based on a comparative study of relevant SPI models, identified in the literature review. ProPAM is focused on three main objectives: (1) further understand how modelling and implementation of software processes can contribute to successful SPI programs; (2) to provide a SPI methodology for SPI practitioners to ensure a successful process implementation; and (3) to contribute to the body of

knowledge of SPI with a focus on implementation of software processes based on project experience.

In 2006, the authors presented a case study [12] on an early version of the ProPAM methodology, which they did not specify the two complementary views and the levels of the process improvement methodology. That case study illustrated the application of ProPAM considering only the temporal perspective (dynamic view) structured in phases and iterations. The purpose of the methodology application was to obtain a detailed description of the organization's software development process, following a top-down approach, in order to validate the proposed approach. According to the applied research methodology (research in action), that case study was intended to empirically collect data to specify the static view of ProPAM.

This paper presents the latest version of ProPAM, introducing in detail the disciplines that compose its static view. The strategy of this case study follows a bottom-up philosophy, different from the first case study, the start point will be the project. The focus of this paper is on project monitoring, problems identification, new practices proposed and data analysis related to software process improvement performed by stakeholders involved at both levels (project level and process level) and, if successful, could lead an improved software process.

The remainder of this paper is organized as follows: Section 2 presents related work and initiatives. Section 3 overviews ProPAM and describes the details of its static viewas core and supporting disciplines in terms of activities, work products and roles. The main results regarding adoption of the ProPAM in a case study are presented in section 4. Finally, Section 5 presents the conclusions and discusses our perception that this proposal has innovative contributions for the community.

2. Related Work

There are some SPI models, such as CMMI [5], ISO/IEC 15504 [6], ISO/IEC 29110 [13] and MPS.BR [14] which are well known among practitioners and researchers. However, the implementation and adoption of SPI at software development organizations is frequently unsuccessful [15, 16]. These SPI models are often prescriptive and attuned to those relative areas for which they are intended and therefore do not take into account other aspects like project and organization specific features. Rather than just repair and adjust the process to specific areas imposed by these SPI models. We claim that practitioners should refocus SPI to analyse the current organization practices and introduce practices adapted to the organizations' needs.

Table 2 compares characteristics of the most important SPI models. Due to time and budget constraints, small and very small organizations have been unable to apply standard approaches such as CMMI and ISO/IEC 15504. These standards target large organizations and are too long to implement. Based on the new trends (see by approaches like ISO/IEC 29110) and considering that organizations must be able to adapt to new situations, we propose that SPI should be based on project's experience and learn from project team member. Software development is not a rigid or a controlled industry. It has a strong creative and social interaction that cannot be totally re-planned in a standardized and detailed process model elaborated by specific groups and without active participation of all team members. SPI models, like CMMI, ISO/IEC

15504 and MPS-Br, identify **what** to improve but do not give information about **how** to do it. Indeed, the ISO/IEC 15504 also pretends to mitigate this issue. So, given the reported problems with existing SPI models, there is a need for a more comprehensive model to SPI.

Table 2. Comparative study of SPI models

Category	Characteristic	CMMI v1.3	ISO/IEC 15504	ISO/IEC 29110	MPS-Br
General	Geographic Origin/Spread	USA/World	World/World	World/World	Brazil/Brazil
	Scientific Origin	SW-CMM, ISO/IEC 15504	ISO 9000:2000, ISO 9001:2000, ISO 12207, CMMI	ISO/IEC 12207, ISO/IEC 15289	CMMI, ISO/IEC 15504, ISO/IEC 12207
	Development	2010	2004	2011	2003
	Popularity	Top (USA)	Moderate	Low	Top(Brazil)
	Software Specific	No	Yes	No	No
Organization	Actors/Roles	Management, Engineering Process Group, Partner	Management (senior manager, process owner)	Customer, Project Manager	Customer, Organization, Assessor
	Organization Size	All	All	Very Small	Small and medium enterprises
	Coherence	Internal and external	Internal and external	Internal and external	Internal and external
Process	Prescriptive/Descriptive	Both	Both	Descriptive	Both
	Adaptability	Limited	Yes	No	Yes
	Assessment	Organization/ Process Maturity	Process Maturity	Process	Process Maturity
	Philosophy	Goal-Oriented	Goal-Oriented	Purpose and objectives	Purpose and results
	Comparative	Yes, maturity and capability level	Yes, capability level	Yes, profiles	Yes, maturity level
	Certification	Yes	Yes	Yes	Yes
	Appraisal method	SCAMPI	Spice Doc. Part 7	Process Assessment Model (PAM)	MA-MPS
	Analysis Techniques	CMMI appraisal/ Questionnaire	interviews or questionnaires	Self-evaluation	Interviews
	Assessor	Internal and external	Internal and external	internal or external	internal and external
Improvement	Perspective	Organizational	Process	Process	Organizational
	Improvement Initiation	Top-down	Process Instance	Project	Top-down
	Maturity/Capability	5	6	-	7
	Improvement Focus	Management Processes	Management Processes	Project Management	Management Processes
	Process	25 process areas	48 processes	2 processes	23 processes
	Progression	Stages and Continuous	Continuous	-	Stages
Empirical Evidence	Goal	Process improvement, supplier capability determination	Process assessment	Process Assessment and Improvement	Process Assessment and Improvement
	Process Artefacts	Process documentation, assessment result	Process profile, assessment record	Deployment Package	Process documentation, assessment result
	Empirical Validation	Survey and projects	Surveys and projects	Case studies	Case studies

We argue that SPI initiatives requires further researches on SPI models based on real-world projects experience. Following the trend of agile processes [17, 18], SPI initiatives require that the organizational knowledge should be constructed through strong collaboration of all team members. Therefore, we should include guidelines in a SPI model that allow to incorporate project team knowledge in the software process (without constraints imposed by a standard which limits embed tacit knowledge) and that can address features not focused on existing SPI models.

3. Process and Project Alignment Methodology – The Static View

ProPAM is a SPI-based methodology with the purpose to capture process and project representations and to allow project teams to imbibe and use knowledge, improving their work [19]. ProPAM is different from existing models in which SPI is seen as starting for the implementation of best practices according to a predetermined scheme. ProPAM proposes to solve identified problems in software development projects carried within the organizations.

A critical feature of ProPAM is the integration of SPI activities with software development activities. In that way, ProPAM considers projects and project teams as the baseline for improvement. Project managers and project teams, under the supervision of the process manager, are the foremost responsible for keeping the organization's processes on the leading edge (table 3).

Table 3. List of main roles considered in ProPAM

Role	Description	Level
Process manager	Concentrated in process definition and implementation	Process
Project manager	Plans and manages the project, coordinates interactions with the stakeholders, and keeps the project team focused	Project
Project team member	Execute project activities. He avoids repeating mistakes by studying lessons learned	Project

As Figure 1 illustrates, ProPAM includes SPI activities to monitoring and tracking software projects (project level) besides the SPI activities that intend to develop and implement the software process (process level). The scope of these levels is defined considering that process and projects actors collaborate on SPI programs. However, to manage the inherent complexity of these levels, namely ProPAM represented at process level, it is common practice to include views on each level. In general, a view is defined as a projection of a process model that focuses on selected features of the process [20]. ProPAM is organized in two correlated and complementary views: the static view and the dynamic view. The static view describes aspects of the methodology as core and supporting disciplines in terms of activities, work products and roles. On the other hand, dynamic view shows the lifecycle aspects of ProPAM expressed in terms of stages and milestones.

The remaining of the section is dedicated to details of the static view. Previous work already described the dynamic view [19]. The ProPAM static view describes disciplines

involved in SPI and relations between them. Static view is expressed as workflow diagrams, which show structural elements (roles, work products and activities) involved in each ProPAM discipline. Swim lanes in the workflow diagram make obvious the roles responsible to perform specific activities and also identifies involved input and output work products. For each role, control flow transitions between activities are omitted since activities are neither performed in sequence, nor done all at once. Nevertheless, such representation does not describe SPI program changes with time passing. A time-based perspective of the process is left to the dynamic view.

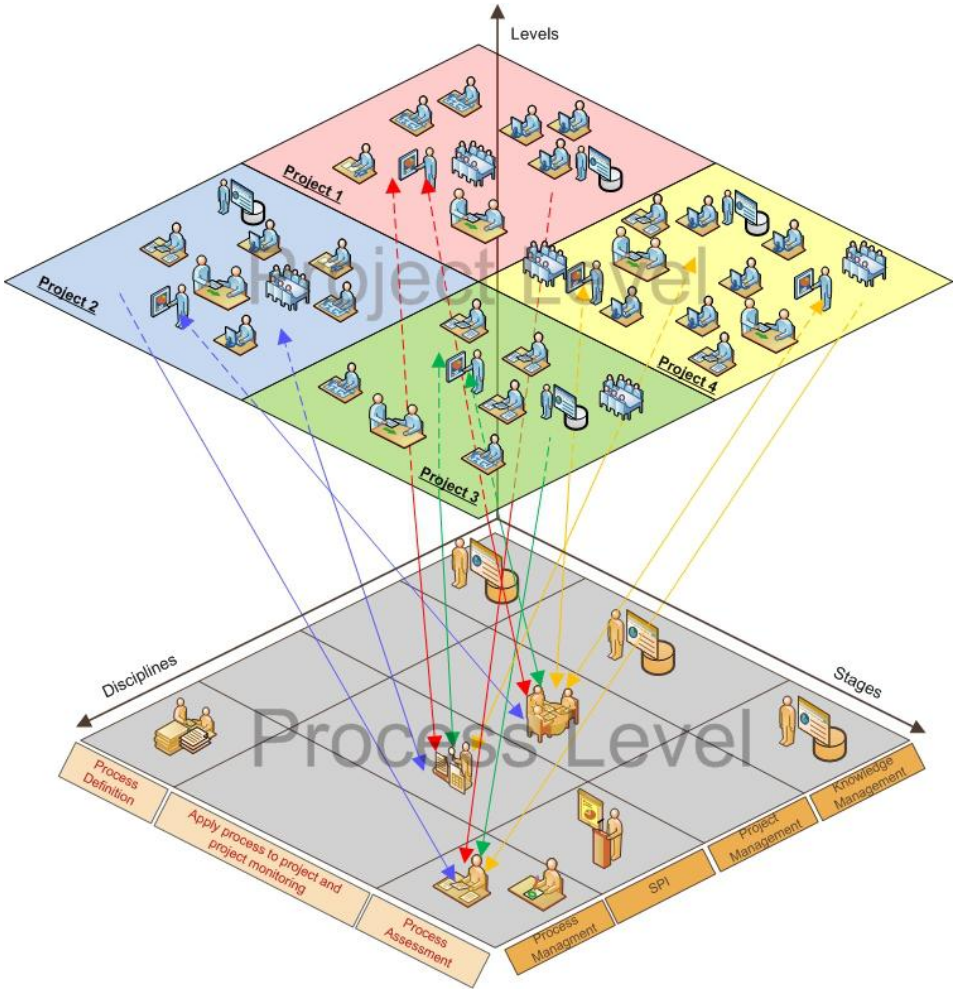

Fig. 1. ProPAM Levels (Process and Project)

At **project level** ProPAM helps organizations in their efforts to assess and manage problematic situations of specific projects, and to develop and implement solutions to manage these problems. The project level encompasses project(s) information needed to systematically support or reject many of the decisions about the process. At project level**,** team members work together to develop work products. This focus on project

team members and their collaborative process is important because no one embodies the breadth and depth of knowledge necessary to comprehend large and complex software systems. Project teams are concerned with concrete situations as experienced in all their complexity during software development. Projects context is constantly being created and recreated and it can't be based on a static process model. Participating in a project team is consequently not only a matter of developing software, but also to change organization's knowledge about software development.

On the other hand, at **process level**, project's feedbacks conduct to process reviews and iterative process improvement. The dynamic interplay between these two levels shows the synergy between the activities performed by project roles (project manager and team member) and the activities performed by the process roles (process manager) involved in SPI. At process level**,** actors involved in SPI programs take time to express its shared practices in a form that can meaningfully be understood and exploited by other organizational actors. This includes not only the definition of concepts, models and guidelines, but also the evaluation of success of the improvements.

The approach can benefit more from an integrated environment that allows to describe process based in project information. We considered that ProPAM is tool-agnostic since can be applied independent of the tools to support different software development disciplines, for example: project management and software process management. In order to validate proposed ideas and contributions of ProPAM, we decide to develop a tool, called ProjectIT-Enterprise. This tool provides collaborative features for process definition, project management as well as process and project alignment. ProjectIT-Enterprise currently supports the two most relevant stages of ProPAM methodology: (1) process definition and (2) apply process to projects. A detailed description of this tool is out of scope of this paper and is given in [21].

ProPAM static view integrates project management, process management, SPI and Knowledge Management (KM) disciplines. These disciplines assure alignment of projects with organization vision and goals, and the adopted and improved software process. Other disciplines of concern were omitted, like business modelling, analyse and design, environment, requirements management or configuration management, because those concerns are considered too specific for SPI programs.

Project Management. Project managers are usually interested in being informed about how the project follows its base process and how to handle changes introduced in the project that are not compliant with the respective process. It is important to detect deviations from schedules (project control and project tracking activities) as soon as possible in order to take corrective actions. Deviations allow identifying elements that do not appear or are incorrectly described in the software process. Therefore, project managers have to be informed about process states in a way that satisfies management needs. This bridges the gap between process management and project management, since project plans should reflect the exact set of activities defined for a given process. To avoid creating detailed plans, project managers may create the plan incrementally, and using only higher-level activities, leveraging lower level tasks only as a guide for how to do the work. The most important goal is to address conflicts and align projects and processes. Figure 2 illustrates the main roles, activities and work products involved in the Project Management discipline.

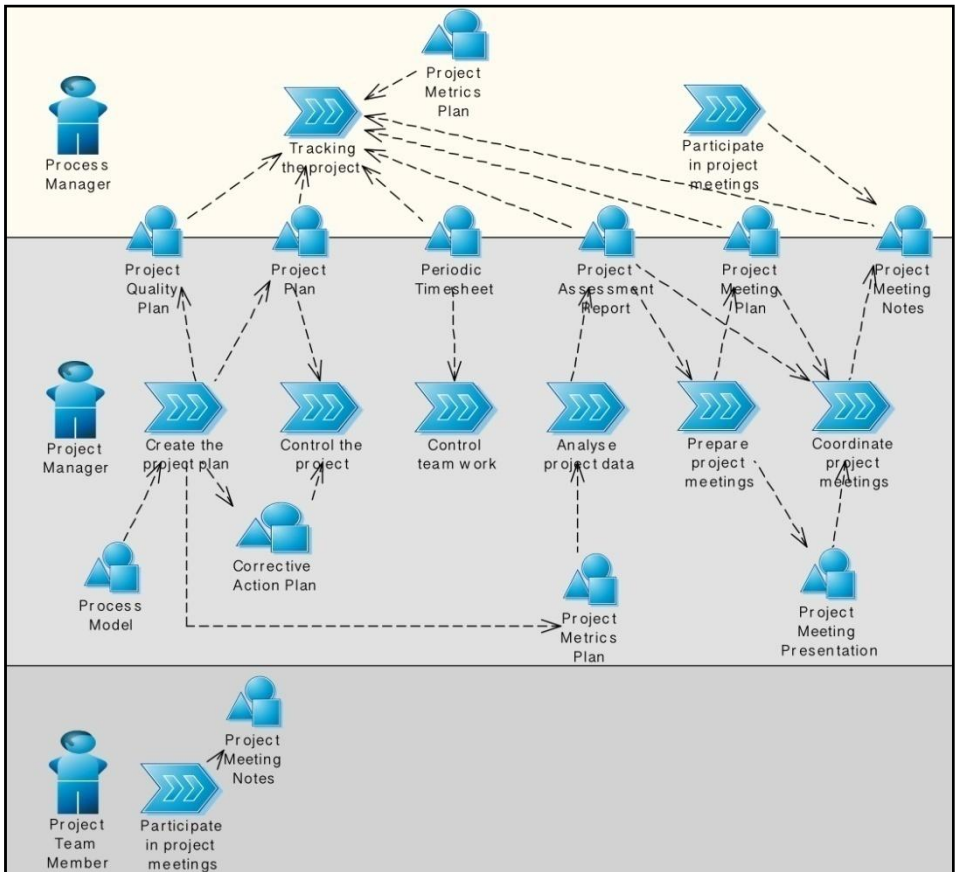

Fig. 2. Project Management View

Software Process Management. Software process management discipline involves actions performed to coordinate knowledge acquisition about software processes, to model and to analyse the way teams develop software and, finally, to ensure that future software processes are carried out on the basis of findings obtained in process analysis [22]. Software process management is a collective work involving project managers, senior engineers and the process manager. Nevertheless, at process level the process manager must be concentrated in process definition and implementation. While at project level the process manager coordinates the interaction with projects team members with respect to process assessment. Software process roles should develop the following activities with direct impact on SPI: (1) collect relevant material; (2) organize interviews and questionnaires; (3) make interviews; (4) understand project experiences; (5) define and implements the process model; (6) establish engineering practices; (7) identify the technical infrastructure; and (8) participate in interviews/answer to questionnaires.

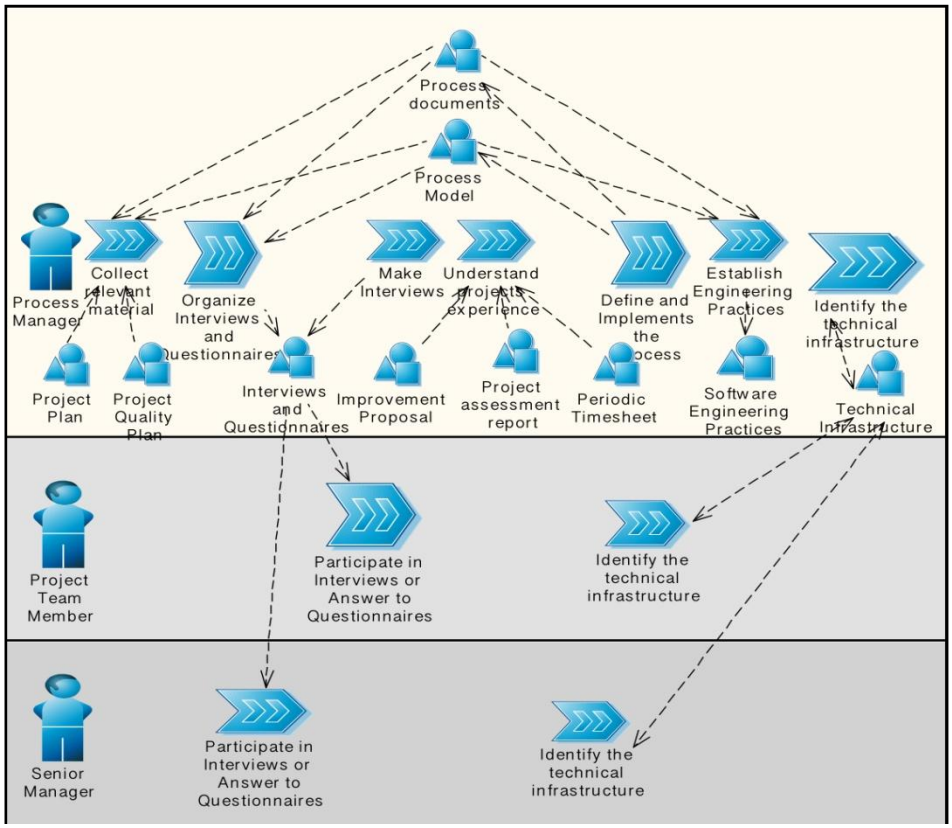

Fig. 3. Software Process Management View

Figure 3 presents the main roles, activities and work products involved in the Software Process Management discipline. Some details of the main activities allow understanding the importance of this discipline. In this case, the project manager has the same responsibilities of the other team members, so he isn't seen as a specific role. The most important goal is to design a set of solutions for the software process based on performed projects. To help the viewer understanding the diagram in Figure 3, a restriction on some flows from and to work products were omitted, since these work products are inputs or outputs of almost all the activities of the discipline.

Software Process Improvement. The effort of supporting software processes is encompassed by the SPI discipline of the ProPAM methodology. This discipline extends the process management discipline, where the main difference is the scope: the process management discipline is concerned with the process configuration for the organization, while the SPI discipline addresses improvements in the process itself based on assessment results. SPI is the discipline of characterizing, defining, measuring and improving software management and development processes, leading to software business success, and successful software development management. Success is defined

in terms of greater design innovation, faster cycle times, lower development costs, and higher product quality, simultaneously [23]. SPI focus is related to establishing a set of responsible roles and associated competences concerned to the software development process with the aim of improving the organization's software process. The main activity of this discipline is the maintenance of software process knowledge and the improvement of coordination and monitoring activities.

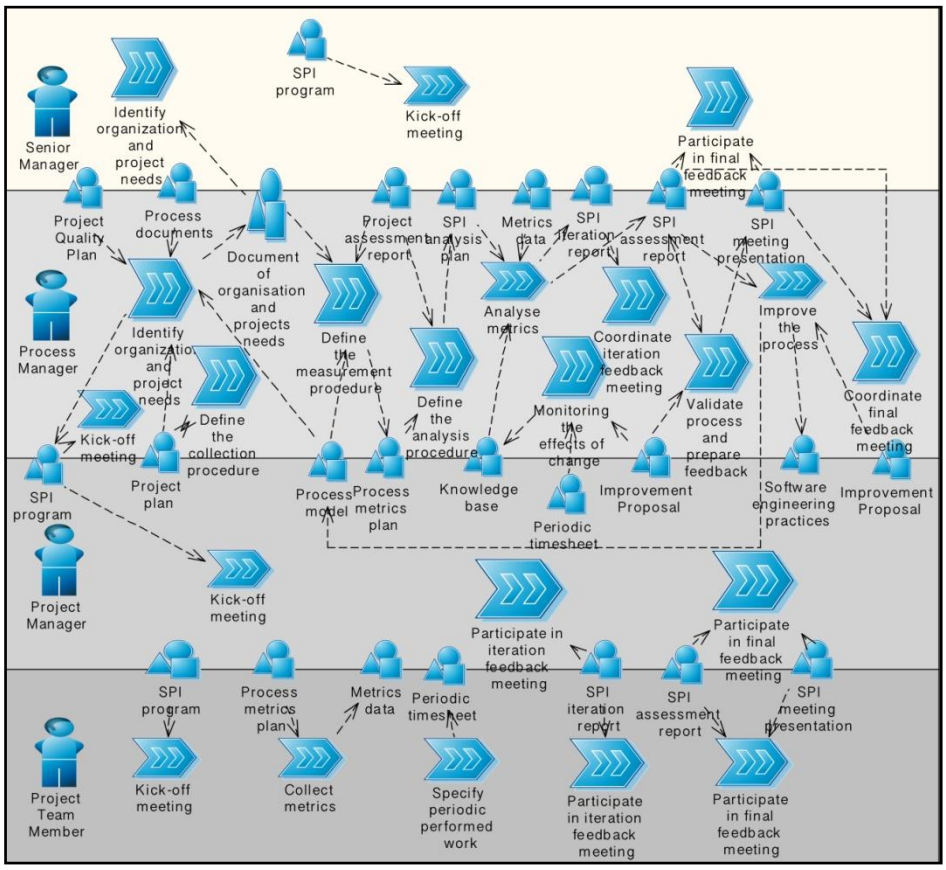

Fig. 4. Software Process Improvement View

The organization must plan to create a stable environment and monitor these activities in order to have clear commitments for current and future projects. The most important goals to be achieved are: (1) software development process and improvement activities are coordinated throughout the organization; (2) the strengths and weakness of the used software process are identified relative to a base process, if it was previously defined; and (3) improvement activities are always planned. ProPAM also suggests that organizations should identify a group of software managers composed by skilled persons and (internal or external to the organization) advisors, who contribute to identify the process strengths and to improve it when weakness are identified. Figure 4

presents the workflow diagram that illustrates the main roles, activities and work products involved in the SPI discipline.

Knowledge Management (KM). Data is organized into information by combining with prior knowledge and the person's self-system to create a knowledge representation. This is normally done to solve a problem or make sense of a phenomenon. This knowledge representation is consistently changing as we receive new inputs, such as learning, feelings, and experiences. Knowledge is dynamic, that is, our various knowledge representations change and grow with each new experience and learning. Due to the complexity of knowledge representations, most are not captured by documents; rather they only reside within the creator of the representation. In many cases, the knowledge representation stays within the creator, in which case the "flow of knowledge" stops.

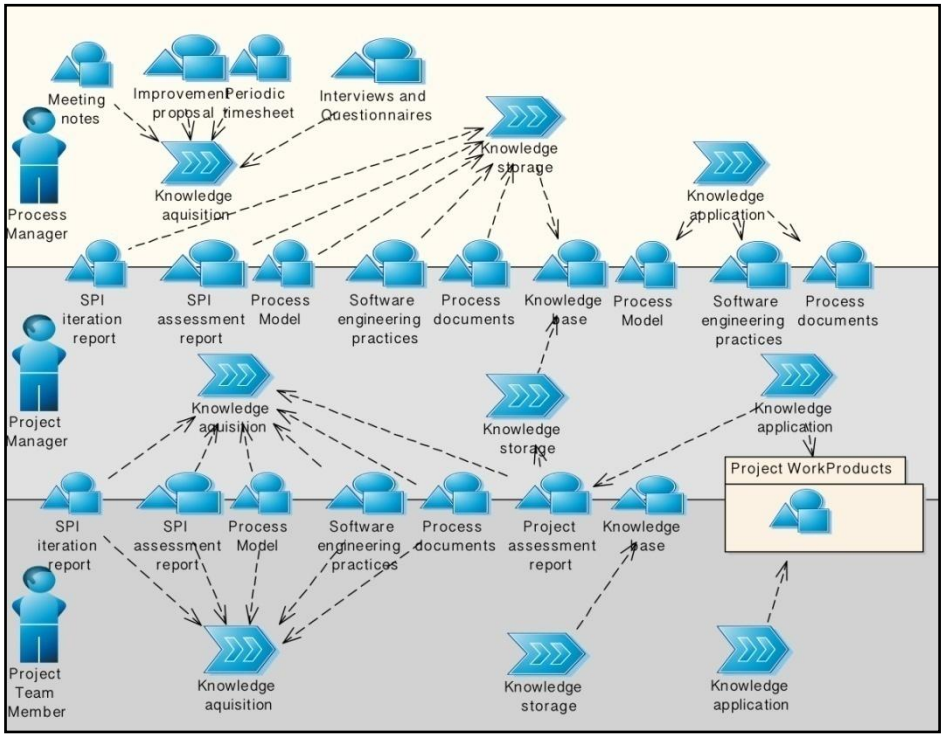

Fig. 5. Knowledge Management View

A KM system, which may be as simple as a story or as complex as an expensive computer program, captures a snapshot of the person's knowledge representation. Others may make use of the knowledge representation "snapshot" by using the story or tapping into the KM system and then combining it with their prior knowledge. This in turn forms a new or modified knowledge representation. This knowledge representation is then applied to solve a personal or business need, or explain a phenomenon. The main goal is to connect knowledge providers with seekers concerning software processes.

Figure 5 presents the main roles, activities and work products involved in the Knowledge Management discipline.

4. Case Study

This section introduces a case study conducted by the authors in the context of a small-size IT organization. This case study allowed us to evaluate pros and cons of ProPAM as a suitable methodology for SME organizations. We collaborate with a software house that had demonstrated interest to define and improve their software development process. The case study included observation of three different projects and application of the proposed methodology to define and improve their software development process. A SPI program was conducted in order to monitor, control and analyse projects developed by this organization (the data in this article only refers to one of these projects due to page limit).

The case study was restricted to Portuguese software organizations which significantly increased the prospects of obtaining the historical informational required to understand process foundation and evolution which would not be the case of multinationals operating in the country, as their processes would likely been initially developed and used within the parent company prior to being disseminated to the Portuguese subsidiary. Because the organization required to remain anonymous, we will refer to as "NISO (Not Identified Software Organization)".

NISO was established in 1996 and currently employs 35 people of whom 25 are directly involved in software development activities (services sector), the others 10 belong to the commercial sector. Actually, NISO provides enterprise and mobile solutions for information management, development and integration. NISO enables clients of all sizes to unwire their enterprises and make information available from the data center to the point of action, and back, anytime, anywhere.

Recently, this organization concentrates on the quality aspect of software development. As a first step, the organization recognized the need to introduce a formalised process. The overall goal was to successfully implement a knowledge management system for the software process in order to assess its effectiveness and, if possible, to still improve it. Having recognized the need to improve its process, the organization sought guidance from our SPI research project through ProPAM methodology.

Prior to this SPI program, this organization had never applied CMMI or other SPI model to diagnose their current maturity level or even improving their software development process. The main problem was high costs incurred for standard certification process and full-time resources allocated to SPI programs. NISO has no financial or resource conditions to accomplish a maturity assessment using CMMI based assessment method.

The aim of the case study was to follow project teams and refine their working practices applying ProPAM. Initially, this meant carrying out a study of current practices employed within the company. Following this, a set of software engineering practices were established which formed the basis of the adopted process. Project Management was one of the areas which showed obvious weakness and, therefore, was chosen as the most important area for the SPI program.

The following sub-section describes the main facts of this case study, namely general data about the three project analysed. The others sub-sections present data of only one of these projects (as justified before). Sub-section 4.2 details the first stage (process definition) of this SPI program. Sub-section 4.3 introduces "the apply process to project(s) and monitoring stage" of this SPI program. Final feedback about process assessment and refinement stage is discussed in sub-section 4.4.

4.1. Case Study Overview

Three projects were conducted and analysed within NISO. However, the customer organizations were different. The first and second project share the same organization was the same entity. While the third project has a different customer. NISO could not justify the support of a full-time process improvement due to cost constraints and its reduced number of collaborators. At the beginning of this SPI program, the organization assigned small project teams due to these reasons.

This SPI program was organized throughout three stages. The first stage was dedicated to an initial process specification based on previous projects information. In the second stage, several activities had been realized at process and project level. At project level, three projects had been under inspection to detect, introduce and validate new software development practices. Then, these practices had been analysed at process level as candidates for future improvements in the base process. Final stage was dedicated to specify the improved process and also included a final feedback meeting to discuss introduced practices.

SPI roles planned and performed improvement activities over a period of ten months, which resulted in the definition of the process (a process model, process documentation guidelines) and a knowledge base (documents, guidelines, projects data, template library). At the end, the changed process had been presented to senior manager and project teams and further improved based on their feedback. Table 4 presents a brief description of the three projects followed.

Table 4. Brief description of the three projects analysed within NISO

Project Name	**NGRID**	**PIS**	**FTF**
Application	Web-based development	Web-based development	Portal (front-end and back-office)
Weeks	7 weeks (planned) 10 weeks	6 weeks (planned) 9 weeks	18 weeks (planned) 25 weeks
Iterations	5 iterations	4 iterations	12 iterations
Project team size	5	4	4

Critical work of a SPI program was developed during the second stage of this case study. At project level, the third project had been monitored during 12 (twelve) iterations, two or three weeks' time each. The first and second project were monitored during fewer iterations, respectively 5 (five).

At process level, only one iteration took place during the second stage. As we can see, at process level, iterations act in a different time scale expressed in months. In this case study, this iteration lasted six months. The nature of the project and process level iterations won't necessarily change much, so we recommend at least one SPI program each year. Figure 6 illustrates the difference between the time scale of the iterations at process and project level. It also identifies main activities and demonstrates the interaction between these two levels.

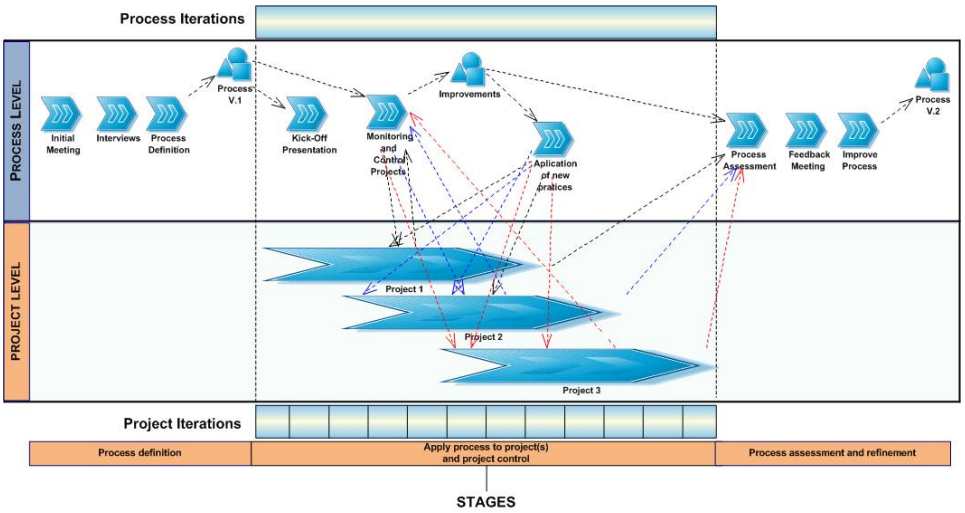

Fig. 6. SPI program at NISO

In the project that we describe here, the software product development focused on implementing a portal supporting several user groups: tennis front-end customer services and back-office management services. The timeframe as well as the cost of the project were supposed to be fixed, based on a commercial contract. Originally, the schedule of this project was set at eighteen weeks. Although, this project had no critical problems considering requirements elicitation, others cases happened pointing towards a substantial project delay. As a result, the planned project of eighteen weeks evolved into a total of twenty-five weeks. In all, twelve software development iterations were conducted in the project. The first, second and third iterations lasted for two weeks and the fourth iteration lasted for one week. Subsequent iterations took three weeks each. The last iteration was concerned with system-testing and final fixing the defects found in the product. The project team was not dedicated in full time to this project, project members were also acting in other projects, so the project duration was set considering the maximal development effort per day.

4.2. Stage 1: Process Definition

In the beginning, an informal meeting with senior managers at NISO showed that their organization needed process improvement. At that time, senior managers had not detailed knowledge of the depth of the problem and how to define and improve the software process. However, they were aware that their administrative capability to solve problems was diminishing. Our goal was to initiate a SPI program to analyse and understand the problems of their software project practices and to contribute to, if possible, to improve their software development process.

The organization did not know when and where to start improvement efforts. After two initial meetings with senior managers to present ProPAM methodology, the SPI program finally starts. The first step was to establish the composition of the project team and the process manager. Project groups were compound by a project manager (responsible for planning, monitoring and controlling projects) and developers (responsible for performing technical activities). The SPI group also included the process manager (responsible for documenting the process and the SPI program) and two project members (responsible for executing the SPI program).

ProPAM proposes four initial activities in this first stage: (1) Initial Meeting; (2) Interviews and Questionnaires; (3) Process Definition; and (4) Kick-off Meeting. Although, the methodology advices these activities (not all of them are mandatory). This time, considering the constraints imposed by the organization, Interviews and Questionnaires were not followed. The initial software development process was defined based on information of previous projects of NISO.

4.3. Stage 2: Apply Process to Project(s) and Monitoring

The second stage (apply process to project(s) and monitoring) identifies and defines the problem with existing procedures, proposes new practices to address these problems and observes the application of the new proposed practices. A comprehensive description of the project monitored will be presented before identifying problems and propose new practices.

Project Monitoring. Considering the reduced number of collaborators, multiple roles were played by the same person in these inspected projects. This condition also was presented for the SPI program. Some collaborators performed several of the following activities: requirements gathering, requirements analysis, project planning, project monitoring and controlling, design, programming and testing. Some compromises may be forced to ignore or diminish some of the activities mentioned above due to the problem of biased judgment. The objectivity of performing reviews, testing, and quality assurance activities may be compromised in this situation. In all projects, the customer was in a central role by iteratively evaluating the quality of the system.

Within this SPI program, several individuals and groups were involved and they were organized as process manager, internal support team, project managers, software development teams and senior manager. Process manager was an external researcher, not a member of NISO, considered as an important element of independent thought. The project manager of the project was permanently associated with the SPI-effort.

Improvements were validated with one or more of the projects and subsequently implemented in the software process. An internal support team helped the process manager in several initiatives to implement the improved process. In the future, senior managers may play process manager roles, while he also has to meet management responsibilities or business goals with strict deadlines. When the top manager is the leader of the process manager, priorities and guidelines to provide status effort must be established.

The project adopted NISO process model, described in the first stage of the SPI program. The software development process was incrementally built during and between the projects and evolved from a simplified version of NISO process to a new improved process version. These process models had been specified through the PIT-ProcessM metamodel [24]. The ProPAM methodology was incrementally validated and improved during these projects.

PIT-ProjectM metamodel [19] had been important as a visual language to facilitate communication with project team members. Through several projects iterations, project models identified the work (activities and work products) assigned to team members. These early models often served as documentation of progress and allow to identify changes introduced in their daily work that had not been reported till this moment. These kinds of models were very important to the project manager in order to track and control the project. At personal view, project team members maintained an overview of their individual work. These allowed them to manage their work, elaborate SPI change proposals and keep a perspective of the current developed work to produce periodic reports. These were the main advantages of PIT-ProjectM metamodel, not only to control projects but also to improve the process based on the new practices introduced in these projects.

This sub-section describes the phases of project PTF: commercial proposal phase and software development phase.

Commercial Proposal phase. This project emphasized the need for a documented and well understood architecture for the developed system. Although the commercial proposal of this project had been written before this SPI program begin, activities performed during this phase followed a pattern common to other proposals that we had opportunity to formally observe and analyse.

This phase has two goals: (1) specify user requirements which will guide commercial proposal terms and (2) define the commercial proposal. Initial effort was oriented to capturing the most important and stable user requirements. Typically, project manager writes the project proposal that describe everything that the project encompasses. This document embodies at a higher level: (1) Project and Organization Structure; (2) Commercial Specifications; (3) Technical Specifications (system architecture, system requirements); (4) Project Schedule and (5) Financial Aspects.

In the Commercial Proposal phase, considering reports submitted by team members and information from the iteration workshop, the process manager identified several problems: (1) customer's representation (sponsor) from different areas were not really motivated to participate (some of them change the meetings date several times); (2) the organization didn't preserve their knowledge about different architectures used in previous project (knowledge repository). So, system architect had to do some research in order to identify the best architecture that fit this solution. Previous experience from others members from the organization could be considered if they had a common

reposition; (3) the project manager spent a lot of time defining the project plan. He also spent additional time confirming project schedule with other team members.

During this phase relevant business requirements were gathered, costs and benefits are defined and quantified. Commercial proposal outlines project plan, associated costs, system architecture and the business solution. Final documents delivered at the end of commercial phase were the commercial proposal, requirements document and analysis and design document.

Software development phase. A summary of qualitative observations carried out during this project development phase is presented below and organized according to process disciplines. Activities of different disciplines had specific problems encountered during this project life cycle.

This phase started with a detailed requirements analysis to help ensure consistent and sound decision-making throughout the system development. However, the phase consisted in two different development sub-phases. In the first sub-phase, front-end system was analysed and a detailed requirements description was done. In the second sub-phase, the same approach was applied to the back-office system. A two sub-phases approach was taken considering the volatility of requirements and the costs of adapting the developed product based upon latter discoveries when interacting with the customer.

At the beginning of each sub-phase, system analyst meets with the customer to identify and negotiate the requirements to be implemented in this sub-phase. During these meetings, customers suggested additional requirements and provided more data about requirements identified in the commercial proposal phase. The approach followed in this project facilitated customer involvement by increasing the frequency of meetings with the customer. Frequent meetings allowed the project team to have continuous feedback from the customer and adjust the activities as the project progressed.

Project planning started with a global project plan view. Across the project, project manager detailed plan only on the features and requirements to be implemented in a specific iteration that enabled project team to incorporate changes in requirements in a later time with less impact to the project. Regular project meetings allowed project team to be adaptable and re-evaluate the requirements addressed in development activities of each iteration.

Although, project team members produced requirements spreadsheets, they didn't control how often software requirements evolved. Software developers should be the first ones to adopt these newer practices. The main problem was that requirements control was performed manually and continuous changes in requirements introduced inconsistencies after some time. So, requirements management and tracking continued a problem throw this project. Everyone knew how important it was requirements management, but no one was committed to check consistency of multiply requirements documents from different team members.

No project's software quality plan was produced, the subsequent lack of control on products quality leads to higher defects in work products and less customer satisfaction. The main reason was the complicated procedure of supervising several projects and support developers in their activities at same time. In the end of the project, project assessment report focused on implementation issues and physical and financial achievements, and less on lessons learned and impact. An example of a supporting activity performed by the project manager is reported here and the respective solution described. Programmers sometimes get stuck or frustrated and needed help to found a

solution. The project manager or a more experience programmer stopped his work and gave some guidelines about how to solve the problem. Pair programming is an alternative approach and proposed solution. Nevertheless, each actor should switch roles frequently, changing from the driver (code writer) to the partner and so on. This approach also involves design decisions, less chance of both actors neglected test, spreads knowledge throughout the team and frequent code reviews.

In the final part of the project, team members delivered period report through the new reporting tool, however periodic meetings were important to inform project team about project progress and problems. Requirements changes were discussed at these meetings and the course of action (project plan) decided by the team but under project manager supervision. Members of the project team were assigned to implement changes in their respective areas of responsibility. Project meetings included risk identification and evaluation. However, no risk mitigation plan was produced. In this project, the team demonstrated a higher level of awareness of risks than at the other two projects. The team (and especially project manager) should always regard risk identification in a positive way to ensure contribution of as much information as possible about the risks it faces. A negative perception of risk causes team members to feel reluctant to communicate risks. Risk identification, analysis, planning, tracking, control and learning are logical activities and that project teams do not need to be followed in strict chronologic order for any given risk. Teams will often cycle iteratively through the identification-analysis-planning activities as they develop experience on the project for a class of risks and only periodically visit the learning step for capturing knowledge for the organization.

Project meetings and SPI iteration meetings provided immediate feedback to process manager considering data provided by project team members, and experience on whether and how the SPI mechanisms needed to be modified. The main idea of SPI meetings was to base process improvement on the obstacles and problems that were identified by the project team.

Without training opportunities at proposed testing approaches, testers are not equipped to meet the rigors of testing, especially in technically difficult situations. Test cases were proposed as an approach to reduce defects reported by final users and maximize customer satisfaction. Since, programmers (at same time testers) were not always in direct contact with the customer, customer acceptance tests were the effective validation technique to ensure the developed system meets their requirements. Despite, the process manager provided support on employing Test Driven Development (TDD) methods, developers reacted negatively considering the absence of adequate tools and lack of training.

New practices introduced in previous projects, evolved within this project. Nevertheless, new methods were proposed, such as pair programming, peer code review, risk management, customer acceptance tests and TDD method.

Improvements were made to data collection practices, especially to collect quality data (such as the number of development defects). Within all the projects, complaints concerning defects collection mechanisms declined toward the end. Consequently, project team's response to improvements in defects detection can be seen as a positive finding.

In this phase several final work products were produced: prototypes, requirements documents, models, code, project plan, project presentation, meeting notes, bug report, test cases and traceability matrix with test cases and interviews. Nevertheless, new work

products were introduced to improve the process, namely: an analysis and design document, a software configuration management (SCM) plan, a quality plan and a risk mitigation plan.

Problems with existing procedures. Figure 7 presents a simplified schema of some most important problems identified through the SPI program.

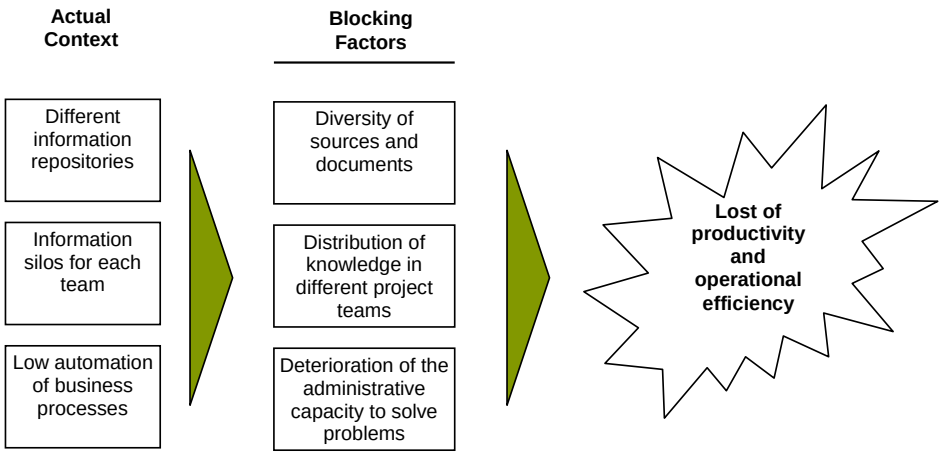

Fig. 7. Problems identified

The following table describes the main problems faced during the second stage of the SPI program.

Table 5. List of problems with existing procedures

Problem	Problem Description
P1	SCM requires extra activities in order to have an operational SCM system
P2	Team members were not motivated because of the time spend in a manual activity with constant updates
P3	Lack of consistency in different requirement documents
P4	No tool support to control requirements.
P5	Team members were not confident about the benefits of test cases (lack of knowledge how to act)
P6	Team members reacted negatively when asked to write test cases
P7	No tool was available to support requirements traceability. Team members has to produce spreadsheets with the traceability matrix
P8	Since they fail in writing test cases, Cross-reference between requirements and test cases was not done
P9	Data from the first project was not available in a knowledge base
P10	Estimation and planning support was weak. No data available in a knowledge base to estimate and create feasible plans
P11	Integrated project management tool not available (NISO intends to produce a supporting tool adapted to their organizational culture)
P12	Initially, spreadsheets are used as templates to periodic reports (reporting tool not available)

Problem	Problem Description
P13	Control bugs were manually implemented (customers do not differentiate new requirements from defects in software products)
P14	Courses were not administrated in this period
P15	Team were not motivated for pair programming because of the time spend in a common activity and the small amount of human resources available to the project
P16	Pair programming not applied (peer code review was proposed to be performed by an extra person)
P17	Risk management not performed, or not effective or results ignored
P18	Client acceptance tests was not an organized and structured process
P19	Team members were not confident about Test Driven Development (TDD)

Concerning the initial process, two disciplines revealed the most problematic cases: project management (planning and estimation, software configuration management and metrics collection) and tests (unit tests and customer tests).

Proposal of New Practices. A new discipline was identified through the SPI program which is knowledge management. Knowledge management revealed as an essential discipline focused on learning of the team members and preserve this knowledge to future projects (knowledge transfer). Organizational practices and guidelines to support project teams in concrete improvements should be managed for future projects and other project teams. These observations reflected the collaborative work of process manager, project manager and other project team members. Altogether, the findings of the projects were group in a total of 17 different improvement practices. Table 6 summarizes all the new practices identified through this project.

Table 6. New practices proposed

Proposed Practice	Proposed Work product	Related Problem
Software Configuration Management (SCM)	SCM plan SCM repository	P1
Specify and control requirements	Requirements spreadsheet	P2, P3, P4
Write test cases	Test case	P5
Customer participation on test cases	Test case	P6
Requirements traceability through design	Traceability matrix with design	P7
Cross-reference between requirements and test cases	Traceability matrix with test cases	P8
Historical data	Knowledge base	P9
Estimation and planning	Project plan	P10
Formal procedures for project planning and tracking	Project management environment	P11
Automate periodic reports	Periodic timesheet	P12
Control bugs reported	Bugs spreadsheet	P13
Project teams training		P14
Pair programming		P15
Peer code review	Bugs spreadsheet	P16

Proposed Practice	Proposed Work product	Related Problem
Risk management	Risk mitigation plan	P17
Client acceptance tests	Client acceptance document	P18
TDD (test-driven development)	Test cases	P19

4.4. Stage 3: Project and Process Assessment

During the period of the pilot case study, we collect data from the project already described. All the data presented in this section were obtained through analysis of project work products and SPI documents. Proposals were written to improve the software development process based on the analysis of the qualitative data collected, the software process improvement literature, and other quality improvement findings from developed projects.

Quantitative research methods are used to establish general laws and principals and its approach can provide answers which have a grounded base. Therefore, the study of software processes lends itself to the application of qualitative methods, as they are oriented towards how project teams view and understand their world and get knowledge from their experiences. As the goals of these projects relate to define and improve the software process of this organization, we also applied qualitative methods as an appropriate technique to take decisions and improve the process.

Three distinct problematic areas were determined through the SPI program, concerning project management (planning and estimation, software configuration management, metrics collection and technical environment), knowledge management (technical environment) and testing (unit testing and technical environment). In the following, these areas are examined to evaluate project practices and improve the process. Figure 8 shows suggested practices (unused, adopted and proposed) organized by disciplines.

Concerning project management, planning/re-planning, estimation and data collection (historical data) were the most problematic areas in all three projects. The lack of method concerned effort estimation, inaccurate definition and re-planning of activities, project tracking and risk management during iterations of project were among the initially most reported problems. However, planning/re-planning of iterations, project tracking and risk management were successfully included in project PTF. Metrics collection was carried out extensively and manually through data collections spreadsheets in the project, for research proposes, however it consumed a lot of time and effort from project manager and process manager. For project proposes, technical infrastructure was not available to support data collection and further estimation.

Unit testing problems were most related to the approach followed by project teams. Figure 9 shows defect rates (bugs reported and changes in requirements requested by the customer) and provide a particularly good view of the state of customer's tests for the project. As illustrated in Figure 9, defect trends follow a fairly predictable pattern in a customer testing cycle. The trend reflected in this analysis shows that new defects are discovered and opened quickly at the beginning of the project, and that they decrease over time. The trend for open defects is similar to that for new defects, but lags slightly behind. The trend for closing defects increases over time as open defects are fixed and verified. These trends depict a successful effort. Since in this project, trends deviated

slightly from these, it indicated a problem and identified when additional resources are needed in specific areas of development or testing.

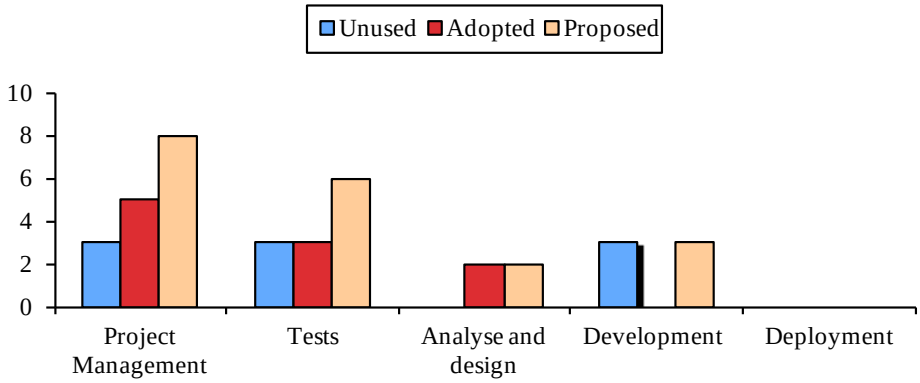

Fig. 8. Quantity of improvement practices by disciplines

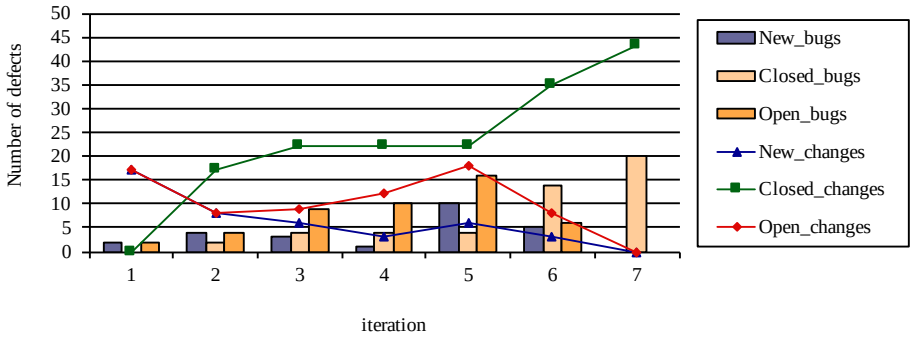

Fig. 9. Defect rates (project PTF)

The trend reflected in this analysis shows that new defects are discovered and opened quickly at the beginning of the project, and that they decrease over time. The trend for open defects is similar to that for new defects, but lags slightly behind. The trend for closing defects increases over time as open defects are fixed and verified. These trends depict a successful effort. Since in this project, trends deviated slightly from these, it indicated a problem and identified when additional resources are needed in specific areas of development or testing.

Most frequent problems derived from the fact that testers reviewed the code they wrote and do not have enough time to do required tests. Alternative techniques must be used, such as peer code review process and test cases. Despite, these improvements were not enough to solve defects or even requirements problems. Senior managers need to take concrete actions and namely, decide to acquire a new tool for TDD activities and invest in improving the TDD training of project teams. Project teams should allocate

more time to planning test cases, elaborate test cases before actual development and, as a consequence, ensure rapid feedback after any change.

This case study reported on the beneficial effects of adequately structuring the development process to improve project management (planning and tracking activities) and get at least an indication of the influences on defect occurrence and defect detection. The main goal was to assess defects that can be prevented by adequate application of defect measures in specific cycles of a project. On the basis of the results of experiences, five improvement proposals were produced, with a specific focus on: (1) process definition and documentation; (2) project management; (3) knowledge management; (4) quality management; and (5) requirements engineering. However, technical environment issue, common to all three areas, was not addressed in this SPI assessment as a priority. Its problems were largely dependent on the internal capacity to develop this kind of supporting tools rather than on the learning of the project team or the state of the process itself. Devices and tools available on the market are not an option since NISO intends to develop their own tools considering the lack of adaptability and costs of existing tools.

After presenting these results to senior managers, SPI group gave priority to knowledge management through:

- Creation a knowledge base that would support all areas of interest identified by the process manager;
- Creation of a document management system to support documentation and sharing of projects results;
- Definition of the software process.

5. Conclusion

In the state of the art of SPI, several problems are identified in what concerns the cost and difficulty of implement effective SPI programs based on the most popular SPI standard models. ProPAM is proposed as a complementary approach to SPI focused on gaps and problems identified on such existing SPI standard models.

A case study was conducted in a small IT organization (e.g. without conditions to accomplish a maturity assessment using CMMI). The main goal of this case study was to give us real world and effective insights into how SPI programs can best suit organizational goals and also showed us the impact on the organization and the strengths and weaknesses of a methodology such as ProPAM. Nevertheless, the adoption of this methodology requires that the involved practitioners be aware on the following limitations of ProPAM: First, ProPAM is based on projects experience, so it is highly context sensitive. There are many factors affecting final results, such as people, facilities and culture. It is important to separate individual practices and process practices and take decisions considering the benefits to the organization. Second, ProPAM is an iterative SPI methodology. People involved in iterative process improvement must be aware about how to performed SPI programs and keep this complex process under control. It is important to explicitly plan and show that a SPI program should have final goals and identifies milestones. Third, mixing process manager roles with project roles makes objective analysis difficult. Because a process manager should validate the changes proposed by project team members, an individual with these two roles probably faces difficulties to make an objective

analysis. In addition, it is unlikely that anyone in the organization will be able to repeat the study to validate process manager observations.

As final conclusion, the prescriptive nature of traditional SPI models (such as CMMI) and costs necessary to implement SPI programs are the main reasons for further research on SPI based on project's experience. Namely, SPI models must address the importance of using the experience of software teams as an important source to defining a SPI. Another gap observed was the deficient alignment between the process and projects. Nevertheless, the contribution of this work was not just an approach to align process and project specifications; we also discussed a mechanism to analyse such evolution based on the changing needs of the organization in consideration.

References

1. Salo, O.: Improving Software Development Practices in an Agile Fashion. Agile Newsletter 2, pp. 8 (2005)
2. Staples, M., Niazia, M., Jefferya, R., Abrahamsd, A., Byatte, P., Murphyf, R.: An exploratory study of why organizations do not adopt CMMI. Journal of Systems and Software 80,883-895 (2007)
3. Basri, S., O'Connor, R.: Organizational Commitment Towards Software Process Improvement An Irish Software VSEs Case Study. 4th International Symposium on Information Technology 2010 (ITSim 2010), Malaysia (2010)
4. Sánchez-Gordón, M.-L., O'Connor, R.V., Colomo-Palacios, R., Sanchez-Gordon, S.: A Learning Tool for the ISO/IEC 29110 Standard: Understanding the Project Management of Basic Profile. In: Clarke, M. P., O'Connor, V.R., Rout, T., Dorling, A. (eds.) Software Process Improvement and Capability Determination: 16th International Conference, SPICE 2016, Dublin, Ireland, June 9-10, 2016, Proceedings, pp. 270-283. Springer International Publishing, Cham (2016)
5. CMMI Product Team: CMMI ® for Development, Version 1.3, Improving processes for developing better products and services. (2010)
6. ISO/IEC 15504-4:2004: Information technology -- Process assessment -- Part 4: Guidance on use for process improvement and process capability determination. pp. 33 (2004)
7. Santos, G., Kalinowski, M., Rocha, A. R., Travassos, G. H., Weber, R. C., Antonioni, J. A.: MPS.BR Program and MPS Model: Main Results, Benefits and Beneficiaries of Software Process Improvement in Brazil. Eighth International Conference on the Quality of Information and Communications Technology (QUATIC), 2012, vol. -, pp. 137-142, Lisboa (2012)
8. Mirna, M., Jezreel, M., Calvo-Manzano, J. A., Cuevas, G., San Feliu, T.: The results analysis of using MIGME-RRC methodology for software process improvement. 6th Iberian Conference on Information Systems and Technologies (CISTI), Chaves (2011)
9. OMG: UML 2.1.1: superstructure and infrastructure. OMG (2007)
10. OMG: Business Process Model and Notation (BPMN). Object Management Group (2013)
11. The Open Group: Archimate® 2.1 Specification. (2013)
12. Martins, P. V., Silva, A. R.: A case study applying Process and Project Alignment Methodology. JBCS- Issue on Experimentation in Software Engineering 12,65-82 (2006)
13. ISO/IEC TR 29110-5-2-1:2016: Systems and software engineering -- Lifecycle profiles for Very Small Entities (VSEs) -- Part 5-2-1: Organizational management guidelines. (2016)
14. Weber, K., Araujo, E., Scalet, D., Andrade, E., Rocha, A., Montoni, M.: MPS Model-Based Software Acquisition Process Improvement in Brazil. In: 6th Quality of Information and Communications Technology (QUATIC 2007), pp. 110-122. IEE Computer Society, (2007)

15. Baddoo, N., Hall, T.: De-motivators for software process improvement: an analysis of practitioners' views. Journal of Systems and Software 66,23-33 (2003)
16. Khaled, E., Emma, D., Goldenson, J., Mccurley, J. H., Fraunhofer, I.: Success or Failure? Modeling the Likelihood of Software Process Improvement. International Software Engineering Research Network (1998)
17. Beck, K.: Extreme Programming Explained: Embrace Change. Addison Wesley (2004)
18. Rising, L., Janoff, N.: The Scrum software development process for small teams. IEEE Software 17,26-32 (2000)
19. Martins, P. V., Silva, A. R.: ProPAM: SPI based on Process and Project Alignment. In: 2007 IRMA Internacional Conference. IGI Publishing, (2007)
20. Verlage, M.: Multi–view modeling of software processes. In: Proceedings of the Third European Workshop on Software Process Technology, pp. 123–126. Springer–Verlag, (1994)
21. Martins, P. V., Silva, A. R.: ProjectIT-Enterprise: a Software Process Improvement Framework. 17th European System & Software Process Improvement and Innovation Conference (EuroSPI2 2010), pp. 2.57-52.66, Grenoble, France (2010)
22. S. Dissmann, V. Gruhn, Ohrndorf, D.: Integration of Software Process Management and Development History Recording. Second Asia-Pacific Software Engineering Conference (APSEC'95), pp. 468-477 (1995)
23. Rico, D.: Using Cost Benefit Analyses to Develop Software Process Improvement (SPI) Strategies. Defense Technical Information Center (DTIC)/ AI (2000)
24. Martins, P. V., Silva, A. R.: PIT-ProcessM: A Software Process Improvement Meta-model. Seventh International Conference on the Quality of Information and Communications Technology (QUATIC), 2010, pp. 453 - 458. IEEE, Porto (2010)

Paula Ventura Martins is an assistant professor of the Electronics and Computers Engineering Department of Faculty of Sciences and Technology at Universidade do Algarve, and a member of Research Centre for Spatial and Organizational Dynamics (CIEO). She has a PhD degree in Computer Science and Engineering from the Instituto Superior Técnico (Universidade de Lisboa). In 2000, she got the Master's degree in Computer Science and Engineering from the Faculdade de Ciências e Tecnologia (Universidade Nova de Lisboa). She concluded her undergraduate course in Computer Science and Engineering on 1992, in the same institution. Her personal interests are Software Process Improvement, Software Development Processes, Domain Specific Languages, Modelling Languages and Business Process Modelling.

Alberto Rodrigues da Silva is Associate Professor with Habilitation at Instituto Superior Técnico (Universidade de Lisboa), he is also a senior researcher at INESC-ID Lisboa, and a partner at the SIQuant company. His main academic and research interests are in the areas of information systems, software engineering, model-driven engineering, requirement engineering, social computing and project management, with multidisciplinary application domains. He is the author or co-author of 5 technical books and more than 200 peer-reviewed scientific communications. He has served on program committees of several international conferences and workshops. He is member of the ACM, PMI and the Portuguese Engineers Association (Ordem dos Engenheiros).

Received: August 2, 2016; Accepted: November 14, 2016.