Providing software maintenance and evolution as a service in a small organization: an approach based on CMMI-DEV and CMMI-SVC

[1]Renata Moreira, [2]Maurício Souza, [3]Yguaratã Cavalcanti, [4]Ana Cristina Rouiller, [1]Alexandre Vasconcelos

[1]CIn – Centro de Informática, Universidade Federal de Pernambuco (UFPE)
[2]Departamento de Ciência da Computação da Universidade Federal de Minas Gerais (DCC/UFMG) – Belo Horizonte – MG – Brasil
[3]SERPRO - Serviço Federal de Processamento de Dados
[4]DEINFO – Departamento de Estatística e Informática da Universidade Federal Rural de Pernambuco (UFRPE) – Recife – PE – Brasil
{rtm, amlv}@cin.ufpe.br, mrasouza@dcc.ufmg.br, yguarata@gmail.com, anarouiller@ufrpe.br

Abstract. This paper evaluates the adoption of CMMI-DEV and CMMI-SVC in small organization for the improvement of software maintenance and evolution process. A Software process improvement (SPI) initiative was performed in a Brazilian small sized software product maintenance organization. We used the Action-Research methodology to evaluate the viability, benefits and lessons learned from the simultaneous adoption of these models. As a result we observed that a set of Process Areas from CMMI-SVC were relevant for supporting the management software maintenance and evolution activities, while Process Areas from CMMI-DEV were relevant for supporting its technical aspects.

Keywords. Software Process Improvement, Software Maintenance and Evolution, Maturity Models, Process Capability Profile

1 Introduction

Software maintenance and evolution is a critical activity during software life cycle. It is commonly described as the process of modifying a software system or component after delivery to correct faults, improve performance or other attributes, or adapt to a changed environment [1]. This concept has been evolved in order to consider the totality of activities required to support the software at the lowest cost, in which some activities start during its initial development but most activities are those following its delivery [2]. Its costs ranges from 50% to 90% of total software life cycle costs [3][4].

Although software maintenance and software development share similarities, some research suggest that a fundamental difference between them is the approach adopted to handle their activities [5][6][7]. Development activities is mainly handled as a software development project, planned to deliver results within an approved budget and

time frame, considering resources, costs/benefits and fixed deliverable objectives. In contrast, to handle software maintenance activities, organizations must adapt to specific characteristics as [3]:

- Maintenance requests (MRs) come in on an irregular basis, and cannot be accounted for individually in the annual budget planning process;
- MRs are reviewed and prioritized, often at the operational level. Most do not require senior management involvement;
- The maintenance workload is not managed using project management techniques but, rather, queue management techniques;
- The size and complexity of each small MR are such that it can usually be handled by one or two maintenance resources;
- The maintenance workload is user-services oriented and application-responsibility oriented;
- Priorities can be shifted around at any time, and requests for corrections of application software errors can take priority over other work in progress;
- Software maintenance is also labor intensive, with the majority of costs arising from programmers' salaries.

Consequently, Software Process Improvement (SPI) efforts based on reference models focused in the software development phase may not “fully” adapt to the characteristics of software maintenance and evolution authors [5][6][7][8].

In this paper we describe the experience of adopting CMMI-DEV [9] and CMMI-SVC [10] for a Software Process Improvement (SPI) project in a small organization focused in the evolution and maintenance of software products. We believe that a service-based approach for managing software maintenance and evolution activities can be complemented with best practices for software process improvement for better results in the given scenario. As a result, we provide describe how the Process Areas from these models contributed for the process improvement in the organization and the improvements observed. The expected contribution of this paper is providing insights for SPI initiatives in similar industry scenarios.

The remainder of this paper is organized as follows: Section 2 describes the related work; Section 3 describes the adopted methodology; Section 4 describes the investigated organization; Section 5 describes the SPI initiatives in relation to the Process Areas introduced; Section 6 discusses the evaluation of the study, discuss the research questions results and raises theories; and, finally, Section 7 discuss some concluding remarks.

2 Related Work

The research carried in [6] was the pioneer in envisioning a software maintenance management strategy through a service perspective, adapting CMM with practices from ITIL. In the meantime, many authors [7][8][5] proposed specific models to manage maintenance. But we believe that the adoption of internationally recognized and up-to-date maturity models are more easily adopted by the software industry.

Research about the adoption of maturity models for software maintenance is diverse. Experience reports as [11] explore the application of maturity models oriented to software development projects. Although the benefits of the adoption of software development maturity models are recognized, we believe that those models do not cover the specific characteristics of the maintenance and evolution activities, and can be complemented with service oriented practices.

Araujo et al. [12] describes the adoption of MR-MPS-SV, a Brazilian reference model for SPI within service oriented organizations, in a Brazilian organization called ECO Sistemas. The organization had already been subject of a SPI program based in MR-MPS-SW (for software development), and the authors point that the models can be harmonized to enrich and organize different teams inside the organization. However their experience focuses in the helpdesk organizational unit, and does not apply a service

Jordão and Kalinowski [13] applied a survey to investigate the applicability of MR-MPS-SV in organizations that have already adopted MR-MPS-SW practices, and thrive for the improvements of its services of software development and maintenance. Results show that MR-MPS-SV can provide benefits in productivity and quality. The results also provide insights showing that the process improvement culture introduced with MR-MPS-SW can make easier the introduction of MS-MPS-SV.

Kalinowski and Reinehr [14] present the definition of a process for software development and maintenance as an IT service. The management of Service Requests was introduced in order to meet Service Level Agreements (SLA). The process was structured based in good practices from Maturity Level G from MR-MPS-SV and good practices related to IT service provision from ITIL. The process adopted agile practices using Kanban. The authors state that a service perspective for software maintenance can provide benefits in productivity, time, costs and quality.

In this paper we share our findings related to the adoption of a service approach to software maintenance and evolution based in CMMI-DEV and CMMI-SVC practices.

3 Methodology

This research followed the "Industry-as-Laboratory" approach proposed by Potts [15], instead of the usual "Research-then-transfer". In this approach, researches are initiated from a practical problem and then refined in continuous and incremental case studies.

During the planning of a SPI in a Brazilian software company (referred as Organization A, for privacy agreements) conducted by SWQuality[1], we identified an opportunity for introducing the best practices from CMMI-DEV and CMMI-SVC simultaneously to improve the software maintenance and evolution activities and related services of Organization A. Based on previous experiences as SPI consultants, we observed that the adoption of Maturity Models related to software development could help improving the process of the Organizational Units related to the software modification activities,

[1] SWQuality Consultoria e Sistemas Ltda (*www.swquality.com.br*)

but SPI initiatives also had to consider other organizational units related to maintenance (such as support units) in order to achieve better results. In order to understand the benefits related to the adoption of these models, we investigated the following two research questions:

RQ1. Is the adoption of CMMI-DEV and CMMI-SVC ML2 viable and appropriated for the investigated organization? – This question evaluates the approach viability considering risks related to the large scope of the project, the size of the organizations and the high impact in all productive units of the organization. We also evaluate if the ML2 Process Areas are suitable for the investigated organization and fit for theirs business needs.

Q2. What are the benefits of the adoption of CMMI-DEV and CMMI-SVC in the investigated organization? – To answer this question, we evaluated how the approach contribute to solve the problems identified in RQ1 and discuss any other benefit observed. Surveys were performed to identify the benefits observed from the perspective of the professional.

We applied an Action-Research approach [16] to iteratively identify solutions and document challenges and lessons learned about the simultaneous adoption of CMMI-DEV and CMMI-SVC in the given scenario. The method was adopted for its emphasis on research and problem solving in a practical and direct way to create knowledge. The Action-Research cycles were adapted from [17] and is composed by the activities Diagnosis, Plan, Intervention, Evaluation, and Knowledge specification.

The SPI project was executed by two researches, SPI consultants from SWQuality. In Organization A, a Software Process Engineering Group (SEPG) was established to act as an interface between the researchers and the organization. The SEPG was responsible for introducing changes to the processes of the organization and providing feedback to researchers about the project execution. The project execution was performed from March/2012 to February/2013 and resulted in the first simultaneous CMMI-SVC and CMMI-DEV SCAMPI appraisal in Brazil.

4 Characterization of Organization A

Organization A has 25 years of experience in the software industry. It provides a software solution and related services (support, deployment, migration, training and development of customized reports) to the market segment of laboratory clinical analysis. Thus, the maintenance and evolution of its software product is a key activity to sustain the organization business model. The company is structured in three main units, related to the organization key activities:

Development (6 collaborators): Responsible for software maintenance and evolution;
Support II (7 collaborators): Responsible for system deployment and migration, customized reports, testing, user training and specialized customer support;
Support I (6 collaborators): Responsible for direct customer support. This unit is responsible for providing users and clients with technical support and guidance in order to maintain the appropriated product usage. This unit is also responsible for registering and directing specific customer needs to other organization units.

Since organization A provides its products to over 500 customers nationwide, requests for maintenance, new functionalities and general support arrived in irregular basis at the "Support I" unit, from diverse communication channels. Then each support analyst was responsible for providing the required support or directing the request to the appropriate organizational unit. In the "Development" unit, the maintenance requests were executed as they came, and requests for new functionalities were analyzed by the organization executive officer, in order evaluate their relevance to the business. In addition, software evolution proposals were planned by the executive officer to ensure business competitiveness and differentiation.

This led to problems regarding the difficulty in managing requests (registering, tracking their status) and, consequently, in the risk of not attending to customer expectation appropriately. This problem was aggravated by the lack of process management and documentation, the lack of clear roles and responsibilities, communication problems. The constant change of priorities, given the urgency of some requests and business opportunities, was a challenge to plan releases and to estimate work efforts, divided between maintenance and evolution requests. The organization size implicated in the development unit being responsible for doing all activities related to implementation of changes in the software.

In the technical aspect, the capability of maintaining the software was hampered by the low understanding of the maintained system due to the lack of software documentation and the necessity of training the maintainers both in the software technologies and in its business rules.

The lack of proper documentation and training in the software usage, also led into a increased number of support request from the users.

The problems observed in the organization during initial diagnosis phase of the SPI project (Figure 1), required that the SPI efforts were tailored to the organization needs.

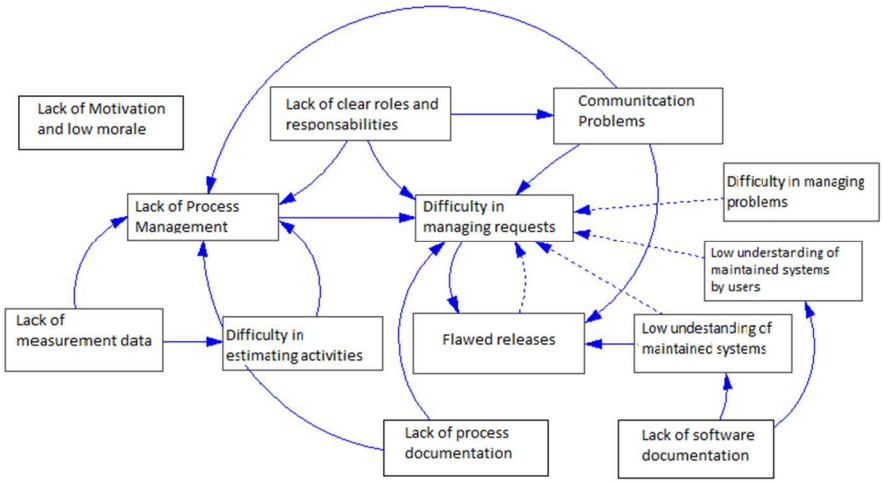

Fig. 1. Problems Observed in the Organization A.

The SPI efforts should not focus only in the “Development” unit, but comprehend the “Support I” and “Support II” units the dependencies amongst these units. The SPI effort should consider the specific characteristic of software maintenance and evolution and the limitations imposed by the company size.

5 Process Improvement in Organization A

The SPI program in organization A was performed in 11 months. In this Section, we discuss how the Process Areas (PA) from CMMI-DEV and CMMI-SVC were introduced to Support the process improvement in Organization A. In addition to PA from Maturity Level 2, we introduced some PA from ML3 considered important to attend to the problems identified during the diagnosis.

5.1 Service Delivery (SD)

In order to improve the request management, a “Service Portfolio” was established, mapping all activities performed in the organization into services. Thereafter, the organization improved their communication with customers, describing Service Level Agreements (SLA) for each service provided. The SLA helped the organization in understanding and defining clear responsibilities for each unit in relation to the provided services. The SLA described clear procedures for accepting, monitoring and closing service requests and also described the responsible organizational units, obligations, request life-cycle, and objective parameters to evaluate the quality of the services provided. The specified services for Organization A were: “system deployment”, “migration”, “training”, “support”, “Product Evolution”, “Product Maintenance” “custom report development".

The communication between units and the coordination of efforts to provide specific services was internally managed based in Operational Level Agreements (OLA). These OLA provided procedures and parameters to evaluate and track services provided internally amongst organizational units. For instance, the service “Product Maintenance” involves the units “Support I”, “Support II” and “Development”, described as follows:

1. A Support I Analyst assess the problem described by the customer. If it’s related to system usage or configuration, the Support I Analyst provide the necessary guidance (“Support” Service) to the user for the problem solution. The solution is registered and the request is closed;
2. If the problem is not solved or if it’s clearly related to a software failure, a “Product Maintenance” request is registered, it is assigned to the Support II unit. Support II analysts evaluate and try to apply a definitive solution. If the problem is not resolved, a palliative solution is applied and the problem is assigned to the Development unit with updated details and urgency evaluation;
3. Based in the priority of the problem and the palliative solution provided (or not) by the Support units, the Development unit evaluates the problem urgency. Non-urgent corrective maintenance are placed in the product backlog as an external bug and are planned for the next development iteration (or in the current, if there are available

resources). Urgent corrective maintenance are immediately placed in the current development iteration and flagged as a “non-planned” activity.

4. Support I unit is notified about the progress of the requests. When the problem is resolved, a detailed description is communicated in the closed ticket and the customer is notified.

The SLA provide clear and objectives parameter to monitor and evaluate the execution time of each request. In the example provided above, the expected time limits for “Product Maintenance” service is governed by the urgency of problems.

- Urgent problems that prevent the customer from performing their activities: 3 days.
- Urgent problems that do not prevent the customer from performing their activities: 6 days.
- Non-urgent problems: 15 days.

To support the definition of a Service System, an internal Helpdesk tool was used to support the management of the Support Teams activities and to register all customer service requests. The tool “Redmine” was customized to support the management of activities for the Development Team. To improve the reception, registering and prioritization of requests, formal communication channels were institutionalized, and a Support I analyst was responsible for the initial customer contact and placing the incoming requests in queues.

5.2 Work Planning (WP) and Work Monitoring and Control (WMC)

In the “Development” unit, a six people workforce was divided into developing new functionalities from a constantly growing product evolution backlog, and the unpredictable load of maintenance requests. Planning activities was difficult because of priorities changing and the difficulty in estimating the activities. We introduced the concept of Work Plans, as defined in the CMMI-SVC, instead of organizing activities in traditional Projects. This way, the planning of activities was concerned about stablishing goals, based in the provision of services during a fixed time. Monthly work cycles were planned in relation to goals, allocation and availability of resources (mostly based in men-hour available), risks, schedules for process related activities and estimates of workload based in history data of previous cycles. Each cycle was composed by a planning phase, two development iterations (2-week long), based in the sprints from the Scrum framework, a review milestone between the iterations and a conclusion phase.

As stated in the literature, software maintenance is labor intensive, with the majority of costs arising from salaries [3], hence the planning of the work cycles was strongly based on the people availability. The first planning action was the accounting of available working hours for each Team Member. The total men-hour was the key resource for the Manager to plan activities. From the total men-hour available, the effort planned for process activities (scrum ceremonies, audits, analysis meetings, management activities) were subtracted and the remaining men-hours represented the available resources for development activities.

The scope of development activities was composed of known set of prioritized software evolution requirements (from a product roadmap) and an unknown amount of maintenance requests that arrives in an irregular basis. For this reason, the amount of unplanned (maintenance requests) work from the previous cycles became a factor to decide how much effort would be planned for known activities (development and process related), and the remaining would be saved for unplanned activities. We perceived that this action resulted in a better control over the changing priorities, and allowed the team to provide a "Product Maintenance" service in accordance to expected service levels. An example of effort planning is shown in Table 1.

Table 1. Effort Distribution example.

Process and Management related activities		Available time for Development Activities		Time Buffer for unplanned Activities (15%)		Total
Sprint 152	Sprint 153	Sprint 152	Sprint 153	Sprint 152	Sprint 153	
87 h	80 h	203 h	182 h	30 h	27 h	552 h

Similar work cycles were introduced in the "Support I" and "Support II" units, with proper adaptations to their activities. Table 2 shows effort planning for the "Support I" unit in a given work cycle.

Table 2. Effort estimates for Support.

Activity	% Hour Planned	Total Hours
Filtering and Training	32%	183
Customer Support	65%	375
Internal Meetings	3%	16
Total	100%	574

By the end of each iteration, retrospectives meetings were held where data from measurement (described in the subsection "Measurement and Analysis") reports are used to evaluate the performance of teams and perceived quality of the services. During this activity, the work plans are revisited and the work progress is evaluated in relation to the plans. Actions are planned to prevent or to mitigate deviations from plans and expected goals, performance and quality levels.

5.3 Measurement and Analysis

To support management processes and ensure greater visibility and understanding of the organization's activities, a measurement base was structured. Changes were introduced in the organizational culture in order to promote measurement analysis. The management tools (Redmine and the internal helpdesk tool) allowed us to store and extract quantitative data about activities such as: spent time, number of activities, types of activities, categories of tickets, and others. Historic data about the team productivity was used to support estimates.

The introduction of the service perspective brought the need to assess data about deviations from the SLA and the quality of services. Once a measurement baseline had

been set, the Goal-Question-Metric (GQM) approach was applied to establish improvements objectives, information needs and the appropriate metrics [18]. A sample of measures established includes: effort in Hours, size in story points, occupation rate, customer satisfaction rate, compliance rate with the process, user history acceptance rate, rework rate and instability scope.

For each measure, thresholds were defined for “good”, “alert” and “critical” levels. The data was collected and evaluated during Sprint Retrospective meetings.. In case of Alert and Critical levels for any indicators, the causes are investigated and actions are planned for mitigate impacts.

The introduction of Scrum framework allowed us to collect information from sprints such as planned activities, story points, sprint velocity, accepted and rejected points. The main productivity data used was the “Story Point Cost”, given by the ratio of work time to Story Points done in a work cycle. Using the mean value of the past four work cycles multiplied by the available men-time for development activities, we determine the expected size of the project scope in Story Points.

5.4 Configuration Management

Configuration Management played a key role in the organization, for both controlling the changes of products and services.

To ensure control of the product evolution, a formal procedure for change control was established in the organization. Any change request (for evolution or maintenance of the product) had to be registered in Redmine as a backlog item. These requests were evaluated for feasibility and impact, and prioritized for development. All the modification made to the software and submitted to the organization's versions repository (SVN) had to be associated to a change request (through log messages during check-in operations). It assured bidirectional traceability between source code and requirements.

A product roadmap was established to plan the release of new features and bug fixes in new versions of the product. Each six months a new version of the product was planned. Additional releases were planned in accordance to business objectives and customers’ needs. Monthly, internal versions of the product were released for testing and for training purpose.

Change procedures were also established for the services provided. Changes related to SLA needed to be formally evaluated and approved, ensuring that impacts were analyzed and minimized.

5.5 Verification and Validation

In the Development sector, testing activities (peer review) started to be executed on all development activities undertaken during the sprints. In addition, the tool “Hudson” was used to perform automated integration tests on the system, and at the end of the sprint, the features were validated in Sprint Review meetings.

Each month, a member of the Support II performed tests on internal releases. And, after the release of version (every six months) a load of general system tests involving all team members was held, lasting 2-3 days. The identified failures now recorded as

"bugs" in Redmine tool, favoring the monitoring of their resolution and allowing to generate measures on rework.

Formal testing procedures were introduced to increase the external quality of the product (lowering the number of external bugs found). As a procedure of the “product evolution” service, Support analysts were responsible for testing internal releases of new versions of the product. As stated before, Support analyst started participating in Development Ceremonies for anticipating

5.6 Requirement Management (REQM)

The REQM practices (from CMMI-DEV) were important to ensure the understanding of the CR (for product evolution and for maintenance). A critical step was the definition of a requirement provider, represented by the Product Owner, acting as a product manager, balancing the user needs, market opportunities, legal obligations and other sources of changes. His role was to maintain a persistent and prioritized product backlog.

Changes in the product backlog were mainly related to priorities changes. The Product Owner responsibility was to ensure the priorities reflected the organizational business goals. In the context of the development iterations, he was responsible for ensuring the team understood the changes they were supposed to do, and to assess the iteration results.

5.7 Requirement Development (RD) and Technical Solution (TS)

The practices from RD and TS were helpful for supporting better software documentation, in relation to requirements, user manuals, release notes, and technical decision over the software changes. Given the high turnover in the organization and the complexity of the software maintained, documentation of business rules, technical solutions and software usage is crucial to sustain the software maintainability.

5.8 Incident Resolution and Prevention

Organization A established a monthly meeting for the manager of each organizational unit with the Chief Executive Office to discuss the situation of each unit. During this meeting, one of the topics discussed was the evaluation of the recurrent problems observed during the month (bugs, user problems and management problems). The top 10 problems were analyzed for identification of causes and to speculate solutions such as enhancive and preventive maintenance requirements for the product.

5.9 Organizational Training

Training schedules were established for support teams and for users to improve the practical knowledge over the software usage. The training sessions comprised pre-delivery or transition activities, ensuring that the organization staff was fully capable of supporting users in the usage of software, and increasing the autonomy of support teams

(especially Support I). Additionally, in every sprint a support professional was selected to participate in a practical training in another organizational unit.

5.10 Product and Process Quality Assurance

A quality assurance process was set in order to verify that the established procedures were being followed in each unit. Deviations from the standard were recorded as "non-compliance" and assigned those responsible to fix in a pre-agreed time. Quality assurance procedures were important to preserve the proper quality of processes, products and services.

6 Evaluation

We observed that the simultaneous use CMMI-DEV and CMMI-SVC reference models allowed us to tailor the SPI efforts to better fit the organization business model. The extent of the SPI project comprised the three main units in the organization, as shown in Figure 2.

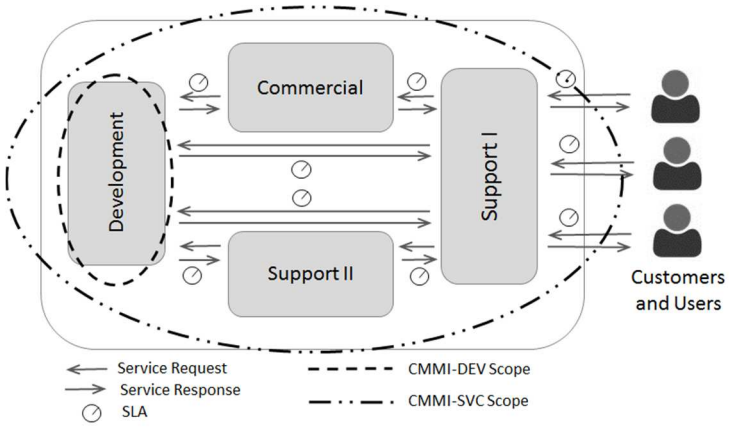

Fig. 2. CMMI-SVC and CMMI-DEV Scope.

Although the SPI project was limited to CMMI-DEV and CMMI-SVC ML2, we observed the necessity of introducing some practices from ML3 Process Areas (VER, VAL, OT, IRP, RD, TS). The experience points that, considering the focus of the organization, the management aspect of maintenance and evolution was benefited from the practices related to improvement of service providing (CMMI-SVC), while the technical aspect (analysis, coding, testing, configuration management, validation, documentation) was better supported by CMMI-DEV practices. In this organization, we observed that the management of activities as a service was more appropriated than the project structure. However, we perceived a lack of product management practices from the reference models to help managing the product evolution backlog.

The SPI project culminated in a successful appraisal of Organization A process in CMMI-DEV and CMMI-SVC ML2. Despite the size of the organization, the amount of Process Areas introduced was not an impediment. The common PA between CMMI-DEV and CMMI-SVC could be introduced in a uniform strategy.

During the Evaluation activity of each Action-Research cycle the SEPG members provided feedback about how the organizational problems were addressed by the SPI actions.

The introduction of monthly work cycles structure, scrum sprints and clear procedures for each service provided was stated as a solid approach for addressing the problems related to communication and lack of process management, process documentation and lack of clear roles and responsibilities.

The introduction of measurement policies improved the overall quality of process management, and allowed better estimates based in quantitative historical data instead of relying only in human predictions. It provided detailed information about the organization performance, the quality of services and overall management visibility.

Figure 3 (a) shows the measure of productivity adopted by Organization A called "Cost of Story Points". It shows the amount of time (in hours) related to the development of a Story Point (SP) in a given sprint. Lower values shows that the team could focus more in the development of new features. In 17 sprints (from September/2012 to May/2013) we observed the decreasing tendency of this measurement.

Figure 3 (c) shows the comparison between the number of planned SP, developed SP and accepted SP (velocity). It shows the precision of estimates and the effectiveness of the team in each sprint.

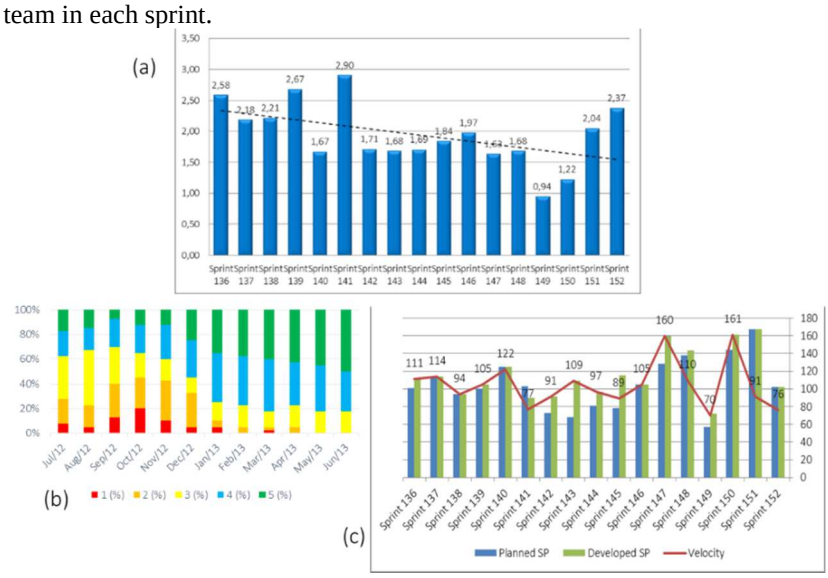

Fig. 3. (a) Cost of Story Points; (b) User Satisfaction; (c) Planning Effectiveness.

The management of corrective maintenances through a service perspective allowed more effective reaction to bug fixing. SEPG members stated that the amount of bugs detected in each sprint and the amount of requests resolved in conformity to SLA, and

that the estimate of a time buffer for unplanned activities improved the efficiency of their plans and the attendance of requests without compromising the commitment to planned tasks.

Formal verification and validation procedures mitigated the release of versions with failures. Testing procedures executed by support analysts both provided better identification of failures before the release of new features to users, and stablished pre-delivery activities, where support teams were able to get familiarized, test and document new features before the release of new versions. Scheduled trainings for users and maintainers were also introduced to the organizational process, increasing the usage quality of the software product.

The introduction of procedures for each service provided, streamlined the management of requests, the accountability of the service quality and allowed each request to be monitored with SLA constraints. Customer satisfaction metrics were introduced, by randomly contacting 10 customers per week whose requests have been resolved in the period. Figure 3 (b) shows and increasing tendency in customer satisfaction from June/2012 to June/2013.

Finally, the introduction of meetings to discuss recurrent problems improved the elicitation of improvement requirements as perfective and preventive maintenance opportunities. The communication between support teams, development team and the executive management was stated as a solid improvement for the product evolution plan.

7 Conclusion

This paper presented insights about the adoption of a SPI initiative based in CMMI-DEV and CMMI-SVC to support the software maintenance and evolution in a small organization. The results show that significant improvements were perceived in the organization. We observed that the managerial aspect of software maintenance and evolution was better supported from a service perspective, while the technical aspect was benefited from the engineering and support practices from CMMI-DEV (e.g. CM, VER, VAL, TS, RD). Thus, we noticed the reference models can be used in complement to each other. Although the SPI project was limited to CMMI-DEV and CMMI-SVC ML2, we observed the necessity of introducing some practices from ML3 Process Areas. Thus, we believe that SPI initiatives in similar environments could beneficiate from CMMI continuous representation [9][10], promoting the adoption of Process Capabilities Profiles [19].

While the achieved results may not be generalized, we believe that this is not a threat to the study, since SPI practitioners could benefit from the reported experience for planning SPI initiatives in similar contexts.

8 References

1. Institute of Electrical and Electronics Engineers: “IEEE Std. 1219-1998 - IEEE Standard for Software Maintenance”, New York, 1998.

2. Bourque, P. e Fairley, R. E. “Guide to the Software Engineering Body of Knowledge – SWEBOK”, v3.0 version, The Institute of Electrical and Electronics Engineers, Piscataway, NJ, 2014.
3. April, A., Abran, A. “Software Maintenance Management: Evaluation and Continuous Improvement”. ISBN: 978-0-470-14707-8. Wiley-IEEE Computer Society Press, 2008.
4. Grubb, P. and Takang, A. “Software Maintenance: Concepts and Practice”, 2nd ed., World Scientific Publishing, 2003.
5. Polo, M., Piattini, M., Ruiz, F. “Using a qualitative research method for building a software maintenance methodology”. Journal Software: Practice and Experience", volume 32, issue 13, pages 1239-1260, November 2002.
6. Niessink, F. e van Vliet, H. “Software Maintenance from a Service Perspective”, Journal of Software Maintenance: Research and Practice 12(2), 103-120, March 2000.
7. April, A., Huffman Hayes, J., Abran, A., Dumke, R. “Software Maintenance Maturity Model (SMmm): The Software Maintenance Process Model”, Journal of Software Maintenance & Evolution: Research & Practice, 17(3), 2005, pp. 197-223, 2005.
8. Kajko-Mattsson, M. “Maturity status within front-end support organisations”, in in 29th International Conference on Software Engineering, ICSE 2007, May 2007, pp. 652–663, 2007.
9. Software Engineering Institute. “CMMI for Development, Version 1.3”, CMMI-DEV v1.3, CMU/SEI 2010-TR-033, Technical Report, Software Engineering Institute, Nov, 2010.
10. Software Engineering Institute. “CMMI for Services”, Version 1.3 CMMI-SVC, V1.3, CMU/SEI 2010-TR-034, Technical Report, Software Engineering Institute, Nov 2010.
11. Pino, F. J., Baldassarre, M. T., Piattini, M., Visaggio, G. “Harmonizing maturity levels from CMMI-DEV and ISO/IEC 15504”. Journal of Software Maintenance and Evolution: Research and Practice 22.4, 2010.
12. Araújo, L. L., Mocny, E. C., Rocha, A. R., Gonçalves, T., Santos, G. “Experiência de Implantação do MR-MPS-SV no Service Desk da ECO Sistemas”. XIII Simpósio Brasileiro de Qualidade de Software. Blumenau, agosto, 2014.
13. Jordão, L., Kalinowski, M. “Investigando a Aplicabilidade do MPS-SV na Melhoria de Serviços de Desenvolvimento e Manutenção de Software”. IX Workshop Anual do MPS, 2013.
14. Kalinowski, M., Reinehr, S. “Estruturando Desenvolvimento de Software como um Serviço de TI: Uma Experiência Prática”. XII Simpósio Brasileiro de Qualidade de Software, julho, 2013.
15. Potts, C. “Software-Engineering Research Revised”, IEEE Software, 10 (5), 1993.
16. P. Coughlan and C. Coughlan. “Action research for operations management”. International Journal of Operations & Production Management, Vol. 22 Iss: 2, pp.220 – 240, 2002.
17. N. Kock. “The three threats of action research: a discussion of methodological anti-dotes in the context of an information systems study”. Decision Support Systems. 37(2) 2004.
18. Basili, V. R., Caldiera, G. e Rombach, H. D. “The Goal Question Metric Paradigm”, Encyclopedia of Software Engineering. New York: John Riley and Sons. 1994.
19. Salviano, C. F. “A Modeling View of Process Improvement”, in SPICE 2011, Dublin. Proceedings of 11th International Software Process Improvement and Capability Determination Conference. Berlin: Springer-Verlag, CCIS 155, 2011.