Abstract

Information systems tend to last several years. But they have to be updated over time to meet the new business needs, originating new requirements. Maintenance plays an important role in the software's life cycle, incorporating the new needs into the existing product, through a controlled, defined and effective development process. Unfortunately this is not an easy task. Most of the projects slipped, exceeding their initial budgets and schedules. Additionally, maintenance costs are rising, with estimated cost from 60% to 80% of the total software budget. Maintenance is an extended subject. This study is focused only in the perfective maintenance process.

A few techniques (such as IEEE Standard 1219, CMM, Scrum and ITIL) are being used in order to reduce the costs and gain control over the process. Besides that it is necessary an excellent knowledge of the perfective maintenance process, as well a management tool with good control mechanisms, aligned with the process.

With this study we propose to elicit the perfective maintenance process and to create a management tool which reflects the process needs and integrates the old systems used by OutSystems.

Working close to OutSystems' professionals, it was defined and evaluated their main needs during the perfective process. The result was a documented and defined perfective maintenance process.

Moreover it was created a software management tool to support the elicited process.

The resulted process was validated by the users and the new tool used to manage new releases. The created tool increased the control over the working items and reduced the total time spent in the process.

Although, innovation, improvements and optimizations continue to be possible in the development process and management tool.

Keywords: Maintenance, Perfective Maintenance Process, Management Tool, Control, Visibility

1 Introduction

Nowadays, requirements often change during the product development life cycle to meet shifting business demands. Software companies need to become more and more agile to satisfy all the customers’ needs. In order to achieve higher software quality and on-time deliveries, good control mechanisms are essential, as well as a defined software maintenance process.

Traditionally, maintenance is seen as one of the final activities of the software life cycle. In the waterfall model, proposed by Royce, is the final activity. The IEEE 1074-1997 standard, represents the seventh step of eight software development steps [1]. Only in recent years, the software community has started to recognize software maintenance as a fundamental area of software engineering [2]. In 1998 were
developed international standards for software maintenance, the IEEE 1219 and ISO/IEC14764 standards which still in use today [1].

Software maintenance is defined by the IEEE standard 1219 [3] as: *Modification of a software product after delivery to correct faults, to improve performance or other attributes, or to adapt the product to a modified environment.*

It is wildly recognized that software maintenance is one of the most expensive phases during the software life cycle, with estimated costs from 60% to 80% of the total software budget [4].

In recent years studies about software maintenance became more complete, resulting in a clear categorization of software maintenance. The most accepted categorization was done by Lientz and Swanson in a survey study [5]. They categorized maintenance activities into four classes:

- **Adaptive** – Changes made to a system to evolve its functionality to changing business needs or technologies;
- **Perfective** – Changes made to a system to add new features or to improve performance;
- **Corrective** – Changes made to a system to repair flaws in its design, coding, or implementation;
- **Preventive** – Changes made to a system to avoid possible future problems.

The survey also showed that the bigger maintenance effort was on the first two types, around 75%, and error correction consumed only 21%. Perfective maintenance is the activity which consumes the most effort, 50% of the maintenance budget, while corrective adaptive and preventive maintenance consume 20% to 21%, 25%, 4% to 5%, respectively, of the maintenance budget [6][7][8]. Another studies show that the core problem for software evolution and maintenance is the incorporation of new user requirements [5].

Software maintenance is an extended subject. This study is focused only in Perfective Maintenance, since it is where the most maintenance effort is spent.

Over the years, information systems researchers spent an enormous effort in this problem, trying to reduce maintenance costs and increase control over the process. New techniques such as structure design, CASE tools, and object oriented programming have been adopted to solve this problem. But, most of them proved to be inappropriate, since the most important factor for high cost of software maintenance is the lack of a close and effective management of the maintenance process [9].

Most of the time the maintenance process is undocumented and undefined, making it less efficient. All the tasks in the process must be fully formalized by procedures and maintainers should have easy access to this information [10]. Formalization of the process is also essential because its degree of formalization is considered to be the main indicator of how mature organizations are [11].

Outdated and inappropriate management tools are another problem for maintenance. Management tools are important, in order to plan the maintenance activity, to allocate resources, to track maintenance tasks, to keep the history of problems and solutions, and to organize and search the history [10].

Few techniques were used to reduce maintenance costs. The IEEE standard 1219 establishes requirements for process, control, and management of the planning, execution, and documentation of software maintenance activities [3]. The Capability Maturity Model for Software provides guidance on
how to gain control of processes for developing and maintaining software [12]. The Scrum agile methodology gives the values, practices and rules which are frequently used for maintenance of existing systems [13]. The Information Technology Infrastructure Library (ITIL) is a well-know standard for IT management, providing a set of good practices for managing IT services. One of its goals is to develop and maintain IT services that are easily developed and enhanced to meet future business needs [14].

2 State of the Art

This chapter explains in detail the state of the art related with this study.

2.1 IEEE Standard 1219


2.2 CMM

The CMM describes an evolutionary improvement path for software organizations from an ad-hoc, immature process to a mature, disciplined one. This path is encompassed by five levels (Initial Level, Repeatable Level, Defined Level, Managed Level and Optimizing Level) of maturity. Each maturity level comprises a set of process goals that, when satisfied, stabilize an important component of software process [15].

2.3 Scrum

Scrum is an agile methodology for software development, characterized as an iterative and incremental process for developing any product or managing any work [16]. Scrum is known to deliver faster and better software for customers or end users, and can be viewed as a collection of good ideas and best practices [17]. It is possible to identify three roles, three ceremonies, and two artefacts in the Scrum process. The roles are: Product Owner, Scrum Master and Scrum Teams. The ceremonies are: Sprint Planning, Sprint Review Meeting and Daily Scrum Meeting. The two artefacts are Product Backlog and Sprint Backlog [18].

2.4 ITIL

ITIL (the IT Infrastructure Library) is a comprehensive, consistent and coherent set of best practices for IT service management processes, promoting a quality approach to achieving business effectiveness and efficiency in the use of information systems [19].

At the core of the library are two volumes: Service Support and Service Deliver. Together they form the Service Management discipline. The set of books become complete with Planning to Implement Service Management, ICT Infrastructure Management, Applications Management, Security Management and The Business Perspective.

ITIL Service Support includes six disciplines that provide flexibility and stability for delivering IT services to the business. These six subjects are: Service / Help Desk, Change Management, Release Management, Incident Management, Problem Management and Configuration Management [20].

ITIL Service Delivery has five disciples that make available high-quality and cost-effective IT services for the business. The disciplines are: Availability Management, IT Services Continuity Management, Capacity Management, Financial Management and Service Level Management [20].
3 Case Study / Problem

During the past months we studied the perfective maintenance process of OutSystems, a multinational software company. The problems found during that period are explained next.

3.1 Wrong and Ambiguous Terms

The first problem that we found during our project was the ambiguous or wrong definition of some essential terms used in the software industry. This ambiguity of terms created communication problems and misunderstands by the participants of the process. In worst cases these difficulties could delay all the perfective maintenance process and increase its costs.

3.2 Undocumented and Ineffective Perfective Maintenance Process

Another problem faced during our investigation was an undocumented and ineffective perfective maintenance process.

A documented and effective software development and maintenance process is essential in today's businesses. If the maintenance process is not documented communication difficulties may occur. It is hard to explain how the process works without a clearly idea of each step. With a documented process is easier to improve it, eliminating useless and redundant stages. Only with a defined process is possible to achieve the higher levels of the CMM making the process more effective and efficient resulting in lower costs of the total project's budget and shorter development cycles.

3.3 Inappropriate Management Tool

The use of an inappropriate tool to manage the perfective maintenance process was another difficulty found during our case study. The system was obsolete and didn't support the user and the process needs.

The main problems revealed by the tool were:
- Lack of visibility over the tasks;
- Weak hierarchal organization;
- Confusing an unclear concepts;
- Few control mechanisms;

Besides these problems, the application was used by all sectors of the company (maintenance, support, help desk, development). This made modification extremely complicated and difficult.

3.4 Integration Between Multiple Systems

Another challenge faced during our investigation was the integration between multiple systems, such as in house systems and commercial systems.

One of the main requirements for the new system was the integration with other systems, Issue Manager and Time Manager (in house systems) and Google Spreadsheets (commercial system).

This was an obligatory requirement for the new system. With this integration the perfective process become more effective since the total time of the process is reduced.

4 Proposal

Based on the problems described in the previous chapter we divide our proposal in three distinct sub-proposals:

- Creation of a common knowledge base of software engineering terms understood by all the human resources in the engineering department;
- Elicitation of the perfective maintenance process and its formalization;
- Creation of a new management tool that reflects the process and the maintenance needs.

During the development, the proposal is periodically evaluated by OutSystems' members. They provide useful feedback which is incorporated in the proposal.
4.1 Software Engineering Terms

We propose to create a common knowledge base of terms by comparing the terms found in the state-of-the-art study with the terms used by OutSystems. The terms will be used in the formalization of the process and will be incorporated in the new version of the system.

4.2 Perfective Maintenance Process

The elicitation of the perfective maintenance process is one of the critical points in our study. It is extremely important to have a defined process to improve the management visibility into progress, increase the control mechanisms over the process and provide a common language and a shared vision [11].

We propose to interview the most important professionals that participate in the process, asking what they do, what are the main actors in each step of the process and how they see the process. Then we gather all the feedback from the interviews and define the perfective maintenance process.

4.3 Management tool

Besides the elicitation of the perfective maintenance process it is necessary to have a tool that reflects it. It is also important that the tool matches all the needs of the development team, making them more productive and effective.

The proposal for the new system is characterized by the following requirements:

- Compatibility with agile methodologies, in particular SRCUM;
- Adaptability with existent systems;
- Easy to change;
- Good control and report mechanisms;
- Incorporation of the best practices studied in the state-of-the-art chapter.

5 Implementation / Prototype

This chapter explains in more detail the implementation and the prototype of the solution.

5.1 Software Engineering Terms

To have an efficient communication it is necessary that the users share the same knowledge about the same terms and concepts used in the development. Communication plays an important role during the process.

To understand what was the knowledge about the terms used in the software engineering field and to verify if they were being correctly used a few interviews were performed to the engineering professionals. After completing these interviews the terms were aggregated to form a single definition which reflected all opinions. Then the created definitions were compared with literature terms, to understand which ones were being improperly used or not used at all. Finally it was created new definitions based on literature terms and in state of the art studies.

5.2 Perfective Maintenance Process

The knowledge of the processes is essential in today’s companies if they want to become more competitive and perform more efficiently. Several members of the engineering department were interviewed to elicit the perfective maintenance process.

First we started with the interview preparation and meeting schedule and decided what questions were necessary for the interview. We asked two main questions to all the interviewees: how they think how was the process and what were the main actors in each step of the process.

To obtain different points of view and impartial opinions we interviewed different people with
different roles within the organization. The selected roles were: Manager, Team leader and Developer.

Finally, when all the interviews were completed all the documents were gathered and a new document with the perfective maintenance process was created based on common information collected in the previous interactions. After finalizing the document, it was sent to all participants in the interviews to validate the elicited perfective maintenance process.

5.3 Management Tool

It is necessary to have a project management tool that matches the needs of the maintenance process, specially the perfective maintenance process. The management tool must reflect the maintenance process and at the same time be agile enough to incorporate new requirements and changes that the process may suffer over time.

Hierarchical Structure

An organized structure of concepts is essential in a project management tool. It increases the control over the working items (releases, projects or tasks), and also improves the visibility over them.

The structure is formed by the following concepts: Product, Release, Project, Activity and Tasks. With this structure a Product can have multiple Releases. A Release has one or more Projects. A Project must have more than one Task. Tasks can have child tasks. The parent tasks are called Activities and they group related tasks.

Graphical Interface

The graphical interface of the management tool is focused only in essential elements and displaying only the necessary fields. A lot of management improvements were made in the new application to increase the control mechanisms and management visibility over the process. Tree, Priority and Iteration view are a few examples.

All the necessary interfaces (add / edit releases, projects, activities and tasks) to support the process were created. Several feedback interfaces were added after performing an action along with report mechanisms to increase the control over the perfective maintenance process. Other improvements that were created are related with the priority of activities / tasks and resources allocation.

6 Results

This chapter describes the results of the Software Engineering Terms and the elicited Perfective Maintenance Process, using both approaches process mining techniques and Interview method.

6.1 Software Engineering Terms

A few software engineering terms (Release, Project, Task and Sprint / Iteration) were redefined. These redefinitions helped the communication and clarify the hierarchal structure of the terms. Most of the engineering terms obtain with this study were used in the definition of the process and in the new management tool.

6.2 Perfective Maintenance Process

The result of the interviews was a macro process of the perfective maintenance process.

As known, perfective maintenance is characterized to be the changes made to a system to add new features or to improve performance. Because of that the perfective maintenance process covers all the common activities of software development, such as requirement elicitation, architecture and design, implementation, testing and deployment.

The macro process can be divided into three major phases:

- Requirements, High Level Architecture and Design;
- Architecture, Design and Implementation;
• Testing and Deployment.

Requirements, High Level Architecture and Design Phase

Requirements, high level architecture and design are characterized to be a management phase, where the scope and all the features for the next release are selected. This phase is formed by five steps:

• Choose the subject for the release;
• Guide lines for the release;
• Requirements’ elicitation;
• High level specification of features;
• Choose the features for the scope's release.

Architecture, Design and Implementation Phase

This phase of the macro process is formed by two processes. The first one is more focused in the architecture and design of the features, while the second is more focused in the implementation of the features.

The architecture and design process is formed by five steps:

• Resources allocation;
• Architecture and specification;
• Specifications' drilldown;
• Estimations;
• Review.

The architecture, design and implementation phase are focused in the implementation of the features. It is formed by four steps:

• Coding;
• Usability Test;
• QA test / Unit tests;
• Documentation.

Testing and Deployment Phase

Testing and deployment is the final phase of the perfective maintenance meta process. This phase is constituted by seven steps:

• Upgrade tests;
• Scalability tests;
• Knowledge transfer materials;
• Beta version;
• Beta support;
• Release candidate;
• General availability.

7 Evaluation

To evaluate the performance and quality of the management tool a few tests were performed. It was used a sample of items to tests the speed of adding tasks because this action is recurrent during the perfective maintenance process. It was also tested the time spent during the Estimations phase of the process using two methods, Edit Screen and 3-Point.

7.1 Add Tasks

It was created 29 items (4 activities and 25 tasks) in the old systems and the new application, and the time spent to add all the items was recorded.

With the previous system it was necessary 15 minutes and 5 seconds to perform the action. During the tasks’ creations a few errors occurred because of it poorly interface and absence of automation's mechanisms.

Using the new management tool it was necessary only 9 minutes and 10 seconds. The new tool needed less time (around 6 minutes) to complete the same task.

Figure 1 presents the relation between time spent and number of items. This graph is based in the previous result.
7.2 Estimations

Another test that was performed was related to the Estimated phase of the perfective maintenance process. During this phase the duration of tasks is estimated in more detail using the 3-Point method by team members.

In this test it was recorded the time spent using the Add/Edit Tasks Interface and the 3-Point Interface to introduce the 3-point duration of 25 tasks.

Figure 2 shows the results in seconds to perform these actions, using both interfaces.

![Figure 2: Time Spent to introduce the duration in Add/Edit and 3-Point Interfaces](image)

Using the Edit Interface it was spent 4 minutes and 30 seconds, while with the 3-Point Interface it was only spent 1 minute and 32 seconds.

7.3 Testimonials

Here are a few statements provided by the users (developers, managers and team leaders) about the qualities and enhancements presented in the new project management tool.

“In my opinion, the hierarchical vision of tasks is the main improvement. It is very useful to view the tasks grouped by parent and create sub-tasks is much more simple. Editing tasks is easier and has some interesting defaults, for example the project is implicit in the edited task. Drag-and-drop order is another improvement. It is very useful to create a view of the tasks’ priority or an order view for each it must be implemented.”

Hugo Lourenço

“With the new tool the creation of backlog items and its prioritization is easier. The release interface is excellent, in particular the view over all projects. Updating the projects’ budget becomes a much simple task.”

Rodrigo Castelo

“The resources planing and allocation tasks are performed more effectively, due to the integration between multiple systems. The tree view also provides a good visibility and control over the working items.”

Luis Lopes

“The reviewed version of Issue Manager, with its new hierarchical view allow me to have an immediate understanding about the tasks’ dependencies presented in the backlog, this was not possible until now. The extra information that the new structure can have (for instance iterations and priority) provides faster way for re-ordination the tasks.”

Miguel Melo

8 Conclusions

To meet the business demands it is obligatory to have an effective perfective maintenance process. To support the process is necessary to have good communication between team members in way that
information flows without misunderstands. It is also very important to have a management tool that satisfy the process needs and helps the users to perform more efficient.

Communication is essential to have an effective process. The definition of the engineering terms clarified the concepts, making the communication easy and reducing the misunderstands.

The perfective maintenance process used by OutSystems has a few similarities with release management of ITIL. Phases such as release design, building and configuration, testing and release acceptance and communication, preparation and training are also present in the elicited process. Both processes are concerned in the protection of the live environment, ensuring that only tested and correct versions of software are available [19].

The process reflects all the good practices of the SRUM methodology. Sprint Planning meetings, Sprint Review meetings and development in a series of iterations (Sprints) characterized this agile methodology. These qualities are part of the process, especially in the Architecture, Design and Implementation phase.

The final result is also extremely important for CMM. With the knowledge of the perfective maintenance process it is possible to achieve higher levels in the software process maturity framework.

The new management tool also gave important improvements to the process. Common tasks became easier and more efficient as result of a simpler interface focused only in the essential elements.

The knowledge of the process combined with an appropriate management tool and clear engineering terms understand by all members offers great benefits. The control over the process is increased and total time of the process is reduced. The predictions became more correct and total cost of the process is reduced.

8.1 Future Work
A lot of work had been made, but still plenty of room for improvements. Here are presented a few ideas that can be followed for future improvements:

- Improve the perfective maintenance process;
- Improve the graphical interface;
- Add templates, Checklists and Tags;
- Obtain the operations that are more performed often.

References


