2008 International Conference on Computer Science and Software Engineering

From Knowing Nothing to Knowing a Little

Experiences Gained from Process Improvement in a Start-up Company

Mira Kajko-Mattsson and Natalja Nikitina
Dept. of Computer and Systems Sciences
Stockholm University/Royal Institute of Technology
Stockholm, Sweden
mira@dsv.su.se, nata-nik@dsv.su.se

***Abstract*— Some software organizations still practice ad hoc development, evolution and maintenance of their systems. Many of them are typical start-up companies. They wish to become immediately profitable without paying heed to rudimentary success factors such as for instance software processes. As a result, they have little insight into and control over their activities. This paper reports on our experiences gained from our process improvement effort at Mobile Navigation, a Swedish start-up company. This effort has aided Mobile Navigation to progress from the stage of knowing almost nothing to knowing at least a little about their processes. (Abstract)**

Keywords-software evolution and maintenance; release management; oral and written communication (key words)

I. Introduction

To step into a highly competitive software market, start-up software companies meet many challenges today [2]. They must make profit from the very beginning, quickly establish their customer base, and make decisions with limited information. Often, they practice ad hoc management of their software systems without having any processes in place [10]. As a result, they have little insight into and control over their activities. Hence, they know almost nothing about them. This was the case of *Mobile Navigation.*

Mobile Navigation is a relatively young company. It was founded in the autumn of 2006 in Stockholm in Sweden. In the first year of its operation, it did not manage to develop any organizational culture, it had little accumulated experience, and it did not have any established technological and methodological base.

About a year ago, it did not have any working method defined. All its development, evolution, and maintenance tasks were performed in an ad hoc manner. As a remedy, we defined a release management process model and conducted a series of process improvement actions.

In this paper, we give an account on our process improvement effort at *Mobile Navigation*. We do it by specifying the improvement actions and by reporting on how they influenced the status within the company. This effort has aided the company to progress from the stage of knowing almost nothing to knowing at least something about the status of their processes.

The remainder of this paper is as follows. Section 2 presents *Mobile Navigation*. Section 3 reports on our research method. Section 4 describes the pre-change status by listing the problems encountered before introducing the release management process. Section 5 briefly describes our process improvement effort. Section 6 presents the post-change status by reporting on how the pre-change problems got remedied. Finally, Section 7 reports on problems related to the process improvement, makes conclusions, and suggestions for future work.

II. Organization Studied

We studied one small Swedish organization. Due to the sensitivity of the results presented herein, we disguise its name. When referring to it, we use a fictive name *Mobile Navigation*.

Mobile Navigation is a small Swedish start up company. It has 16 employees, and manages a software product for mobile phones. Its software product is a freeware available via Internet for unlimited amount of users.

III. Research Steps

During the first step, we studied the status at *Mobile Navigation.* Because the second author of this paper got employed in this company just before commencing this research, we had full access and insight into the whole organization and its activities.

The results achieved in the first step provided strong evidence that *Mobile Navigation* did not have any process in place and that the organization was troubled with many problems related to the management of their software releases. For this reason, we established its status (pre-change status) and listed most of its problems. These problems constituted a starting point for making organizational improvements.

As a next step, we defined a preliminary, however, a complete and fully fledged release management process model. This model was based on the existing theoretical and practical models [1][3][4][6][8][9].

In October 2007, we presented our release management process model to all the employees at *Mobile Navigation*. To be able to enact it, we then defined and performed incremental process improvement actions. These actions were conducted in the course of the next eight months. They went relatively smoothly, although some problems related to process improvement inertia were encountered.

After the period of eight months, we evaluated the status at *Mobile Navigation*. We did it by evaluating the extent to which the problems got resolved. The results achieved in this study provide feedback for future process improvement actions.

IV. Pre-Change Problem Status

When studying the pre-change status at *Mobile Navigation,* we identified the following problems:

Problem 1 – ***Lack of requirements gathering process***: *Mobile Navigation* did not follow any requirements gathering process. The requirements were communicated orally and their specifications were very coarse-grained. They mainly included a general description and graphical sketch of user interface. As a result, developers misunderstood initial requirements and had to redo many of the implemented features. Unstructured requirements gathering process delayed the new feature development for over 40% of the planned time.

Problem 2 – ***Release cycles and their length were not defined***: *Mobile Navigation* did not practice the concept of release management. As soon as only one change request got implemented, a new software version was deployed. As a result, one had no knowledge of how many releases were deployed and when. Usually, a release cycle was no longer than a week. Because of this, developers spent a lot of time on the deployment activities instead of using their valuable time for developing new features.

Problem 3 – ***The releases and their scope were not planned***: The release scope was not defined in advance. Neither did one plan its realization and deployment. There were no deployment dates set and there was no schedule. Usually, management requested features directly from the developers, and changed their assignments frequently on the fly.

The requests were not documented in the change management tool or discussed with the technical management team in advance. Their choice was mainly done in an ad hoc manner. One mainly prioritized new features possessing high business value and neglected the features being critical to the lifecycle assurance. This left many quality, performance or security problems unattended.

Problem 4 – ***Lack of control over the change requests:*** Due to the problems listed above, business and support teams had very little knowledge of the requirements being implemented. If they had some insight into the project, they still did not have full control over its status and progress. Only after a new release was delivered, the business and support teams got aware of its contents. This was due to the fact that there was no priority management of the change requests in place, and thereby, no clear overview of the development status and progress.

Problem 5 – ***Lack of process control*:** Despite the fact that the process was not defined, *Mobile Navigation* lacked control over the release. This is because all the change requests were poorly organized and documented in the change management tool. Additionally, developers were not motivated to use the tool. They did not update the status of the process and of the change requests they worked on. Therefore, the tool was always outdated. As a result, it was impossible to track the status and progress of the process.

Problem 6 – ***Defective releases***: No testing of any kind was performed at *Mobile Navigation*. The only testing that was made concerned a quick check of the implemented change requests, first by the developers and then by the business team. Only limited functionality was checked. As a result, newly released versions consisted of many problems.

Problem 7 – ***Poor communication:*** A lot of oral and written communication is required within a project. Communication aids in understanding the system to be developed and the process used for developing it. *Mobile Navigation* did not document at all. Neither did they employ any oral communication practice. The only communication media were weekly meetings. This was not enough to convey all the information needed to overcome the undocumented system and process.

V. Process Improvement Effort

In this section, we describe our process improvement work. In Section 5.1, we first describe the process improvement method. We then briefly present our release management process model.

A. Process Improvement Method

When designing our process improvement method, we tried to follow the internationally established models such as [2][7][10]. However, we did not slavishly follow them. In contrast to what they advocate, we attended several problems simultaneously and introduced several actions in one sweep. We motivate this with the fact that *Mobile Navigation* was in a hurry to make improvements.

As shown in Fig. 1, our process improvement method includes three main phases: *Evaluate, Plan* and *Change*. At the moment of writing this paper, as indicated by the "**" symbol, we managed to complete the first iteration and step into the first phase of the second iteration. So far, we have accomplished the following:

- *Evaluate*: In this phase, we tried to establish the status within the company. Due to the chaos reigning within it, we could only evaluate it qualitatively. We did not have enough data for evaluating it quantitatively. We then defined a model of the release management process as used within the company and identified problems within it.
- *Plan*: The results of the former phase indicated an urgent need of a defined release management process. For this reason, we decided to develop such a model and define actions for remedying some of the problems. To enable a smooth transition in this chaotic situation, we planned and

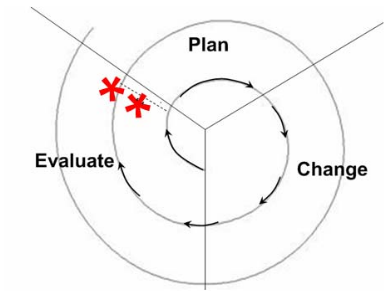

Figure 1. Process improvement method. Symbols ** indicate our present position

prioritized their implementation. The actions are presented in Fig. 2, and listed in the priority order.

- *Change:* Following the plans as defined in the former phase, we first introduced the process and then started to iteratively implement our improvements actions.
- *Evaluate*: After having performed all the planned actions, we established the status by evaluating how the problems got solved. Our results enabled us now to evaluate the process both qualitatively and quantitatively.

B. *Release Management Process Model*

The release management process model that we recommended for *Mobile Navigation* consists of six phases. As illustrated in Fig. 3, they are 1) *Release Scope Preparation,* 2) *Release Planning*, 3) *Release Development,* 4) *System Testing,* 5) *Acceptance Testing,* and 6) *Release Deployment*.

Some of its phases are not pure release management phases. This is because *Mobile Navigation* was in a desperate need of a model integrating development with release management. Also, we would like to point out that the process model is adapted to the current needs of the organization. Hence, it may not be an optimal model. Below, we briefly describe the phases.

Action 1: Motivate developers to use change management tool
Action 2: Introduce a general template for describing requirements
Action 3: Determine goals for each release cycle
Action 4: Determine length for each cycle
Action 5: Determine release scope
Action 6: Determine release deadline
Action 7: Determine functionality testing of the system components
Action 8: Motivate developers to write unit test
Action 9: Motivate business team to use change management tool
Action 10: Introduce daily meetings
Action 11: Enforce a requirements gathering process
Action 12: Define types of release cycle (major and minor)
Action 13: Include engineering team when determining goals and scope for the next release
Action 14: Refine the template for describing requirements
Action 15: Introduce a prioritization process
Action 16: Define a third type of a release: emergency type
Action 17: Improve the dissemination process concerning changes to the release scope
Action 18: Establish performance testing
Action 19: Install automated debugging tools
Action 20: Introduce code review
Action 21: Communicate information about the release to users
Action 22: Introduce the deployment documentation and check-list
Action 23: Introduce deployment testing

Figure 2. List of our process improvement actions

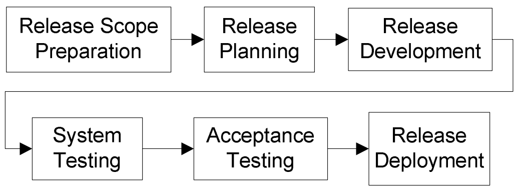

Figure 3. Phases of the release management process

1) *Release Scope Preparation.*

Here, one first identifies general user requirements for the release. Using them, one then defines new functionality requirements and estimates the effort required for developing them. Based on this, one analyzes the requirements, prioritizes them and makes decisions on the release in which they should be included. Afterwards, all the chosen requirements are analyzed in detail, and the user-stories are created.

2) *Release Planning.*

One starts this phase by creating and scoping a release baseline. Afterwards, the approved requirements are prioritized and a set of major activities to develop them is identified.

All the requirements are documented as change requests (CRs) in the change management tool. They are then analyzed, the effort for developing them is estimated, a schedule gets created, and the release plan is produced. This plan is presented to the *Change Advisory Board (CAB)* for approval.

3) *Release Development.*

During this phase, developers evaluate the CRs assigned to them and create designs for these CRs to be then approved by the *System Architect*. After the solution gets implemented, developers perform unit tests. Finally, when release development is completed, the release package is built.

4) *System Testing.*

When the release package is developed, the testing of all the system components is performed. When the release passes the system tests, one suggests deploy it to the testing repository, for acceptance testing.

5) *Acceptance Testing.*

The acceptance testing aims to verify the system functionality. If no problems are identified during this phase, one then decides to deploy the system to the production repository.

6) *Release Deployment.*

Before a new code is transferred to the production repository, one verifies that the system is ready for deployment. The deployment has to proceed according to the deployment checklist. Afterwards, the deployed system is tested and the information regarding the new release is provided to the users.

VI. Post-Change Problem Status

Today, the process is defined, and to some part documented. However, the recommended process model is not fully followed yet. After the eight months of using its improved version, the company has greatly benefited . Below, we report on how our process improvement effort has attended to the problems as listed in Section IV.

Problem 1, *Lack of requirements gathering process,* has been dramatically remedied. Today, all the requirements are structured and documented on a coarse granularity level. The attributes used for documenting requirements are presented in Fig. 4. The delay in feature development has decreased from 40% to 10%. Often, the delays are due to many complementary additions to the requirements as requested by the management.

Problem 2, *Release cycles and their length are not defined,* is no longer a problem. Today, *Mobile Navigation* has defined the following release cycles:

- *Planned major releases* encompassing new features and some scheduled major adaptive maintenance changes. Their length ranges from four to six months.
- *Planned minor releases* encompassing scheduled minor improvements and corrective changes. Their length ranges from two to four weeks.
- *Unplanned emergency releases* encompassing urgent corrective measures. Their length ranges from one to five days.

In contrast to the pre-change status, today *Mobile Navigation* has knowledge of how many releases are deployed and when. It has control over the types of releases it deploys. In the last year, the organization deployed 28 releases in total. Out of them, three were major planned releases developed in 2-5 months, nine were minor releases developed in 2-3 weeks, and 16 were emergency releases developed in 1-7 days.

Problem 3, *releases and their scope are not planned*, is not fully remedied yet. We are however satisfied with the improvement results within this short period of time. Today, the release scope is defined and planned in advance. However, the original management habits of changing requirements on the fly still prevail. Therefore, the release scope is often changed. *CAB* still frequently adds urgent change requests for new releases without paying heed to the release plan. These urgent CRs takes up to 25% of the planned release scope. In the last year, all the major and the majority of the minor releases were affected by such urgent changes.

General Requirement Description:
Requirement ID, Requirement Title, Requirement Description, Requirement Type, Related to Requirement(s),
Requirements Evaluation Data
Requirements Priority (Rank), Requirement Severity
Other Description Data
Environment, Components, Affects Version, Fixed Version, Attachements,
Requirements Reporting Data
Requirements Reporting Date, Requirements Modification Date, Originated by, Requirements Owner,
Requirements Management Progress Data
Requirement Management Status, Requirement Age, Requirement Changes
Requirements Completion Data
Actual Completion Date, Planned Completion Date, Released In

Figure 4. Template with the attributes used for documenting requirements at Mobile Navigation today

Mobile Navigation meets problems of meeting the planned release deadline with all the promised features. In the last year, the release delivery of all the major releases was delayed by 2-4 weeks, and the delivery of minor releases was delayed by less than a week. Additionally, the delivered releases did not always implement all the features as planned for them. Much of the planned functionality had to be moved to the next release(s) instead.

In the last year, neither the major nor minor releases fully implemented all the initially planned features. In average, only 75% of the features were delivered as initially planned. Also, most of the releases were delayed. In average, all the major releases and about 50 % of minor releases were delayed. For about 60% of the features, it took more time to deliver them than it was initially planned. This was due to many reasons. Some of them were poor planning and lack of impact analysis.

Problem 4, *Lack of control over the change requests,* has been partially solved. The control over the change requests has been significantly improved. Because the release is scoped in advance and the change requests planned for it are documented in the change management tool, business and support teams have insight into its contents in advance. However, some problems still remain. Due to the already mentioned spontaneous scope changes and frequent release delays, many change requests often have to be postponed to the next release. In the last year, we observed about 20-30% of change requests being postponed.

Problem 5, *Lack of process control*, has been remedied to some extent. We have motivated people to use and update the change management tool. Today, the tool is always updated; hence the process transparency is greatly enhanced. However, the contents of the release to be delivered is still fuzzy for the business and support teams. This is mainly due to the still limited communication between the business, support and engineering teams and the frequent changes to the release scope. Usually, about 10% of the planned change requests get removed from all the releases two days before deploying them.

Problem 6, *Defective releases,* has been partly remedied. Being part of our release management process model, we have introduced a very general testing process. Hence, today testing is somewhat better organized. It encompasses unit, functionality, performance and deployment testing. Still, however, it does not cover activities such as code reviews, and integration and system testing are almost non-existent. As a result, many emergency releases have to be generated after the release gets deployed. In the last year, *Mobile Navigation* deployed 16 emergency releases. The problems encountered in them were due to the insufficient and nonsystematic testing and the inability to conduct impact analysis. To determine the impact, one needs, among many things, good knowledge of the system. Right now, only the *System Architect* has a complete system knowledge in the whole of the organization.

Problem 7, *Poor communication* has been only partially solved by improving oral and written communication. Regarding the oral communication, one has introduced daily stand up meetings. Regarding the written communication, referring to documentation, one has not done much. The system is still not sufficiently documented. For instance, the documentation of its architecture and design is missing. Also, the process is not well documented. Only change requests and requirements of new functions are recorded. The result is that the company lacks expertise in the system they manage.

VII. Conclusions

In this paper, we have given an account on our process improvement effort at *Mobile Navigation*. On our journey towards a better process, we encountered many obstacles. The most challenging obstacle however was to motivate our employees to change their habits. Many of the process improvement actions could not be easily completed due to the lack of their motivation.

Based on the fact that the acceptance of a process model and the change of the existing habits is a time-consuming process, we understood that to make progress, we had to address one problem at a time and not several problems as we did in this study.

Today, the recommended process model is not yet fully implemented. It still needs a lot of tuning. Despite this, we are satisfied with our results. They have helped us gain a little knowledge about our process in the following way:

- To know what requirements are managed, we created a very simple template for describing them. Additionally, we changed the requirements gathering process to ensure that the requirements follow our coarse-grained template structure. This little change has enabled us to track the requirements implementation on a very general level. However, the template is still coarse-grained. In the near future, we intend to implement the template as specified in [5].
- By defining the release types, we roughly know now where the development effort is spent. Feedback on the number of the releases providing new features and/or corrected defects has aided us in finding out where the effort goes to. Our statistics for the last year show that its major part goes to the development of new features. However, we are alarmed by the fact that as many as 16 emergency corrective releases were generated in the last year. As a remedy to this, in our next step, we will concentrate on improving our testing practice.
- We have not yet achieved a full control over the releases, their scope and length. Still the release scope frequently changes and the cycle length varies. This mainly depends on the (1) lack of impact analysis, and thereby, difficulties in mapping right features on the right releases, (2) management habits of changing the scope on the fly, and (3) lack of testing. As a remedy to this, we need to create an organizational culture that discourages spontaneous scope changes and embraces change in a more controlled way. We also need to define an impact analysis process model.
- By motivating all the teams to use and daily update change management tool, the control over and the documentation of the change requests has been significantly improved. However, due to the spontaneous scope changes, many change requests are still postponed to the next release.
- The use and update of the change management tool also resulted in the improvement of the control over the process. Today, one has insight into the project status and progress on a general level. Also, the business and support teams are aware about the release scope in advance. Still however, change requests get removed from the release just before deploying it.
- Introduction of the unit, functionality, performance and deployment testing has improved the system quality. However the activities such as code reviews, integration and system testing are still not covered by the process.
- Introduction of the daily meetings has improved the communication within a project. However, only to some extent. Still, system and process documentation need be defined.

The above-listed results show evidence of our success with respect to the process improvement. Using them as a basis, we dare say that our effort has aided *Mobile Navigation* to progress from the stage of knowing nothing to knowing at least something about their processes.

References

[1] Arthur L.J., Software Evolution: The Software Maintenance Challenge, John Wiley & Sons, 1988.

[2] Gresse von Wangenheim C. , Anacleto A. , Salviano C. F., Helping Small Companies Assess Software Processes, IEEE Software, January/February 2006.

[3] Kajko-Mattsson M., Corrective Maintenance Maturity Model, Department of Computer and Systems Sciences (DSV), Stockholm University and Royal Institute of Technology, December, 2000.

[4] Kajko-Mattsson M., Motivating the Corrective Maintenance Maturity Model (CM3), In Proceedings, Seventh IEEE International Conference on Engineering of Complex Computer Systems, 2001, pp. 112-117.

[5] Kajko-Mattsson M., Nyfjord, J., A Template for Communicating Information about Requirements and their Realization, In Proceedings, IAENG International Conference on Software Engineering, BrownWalker Press: Boca Raton, USA, 2008.

[6] Kajko-Mattsson M., Yulong. F., Outlining the model for the release management process, Society for Design and Process science, printed in the USA, June, 2005.

[7] Mc Caffery F., Taylor P. S., Coleman G., Adept: A Unified Assessment Method for Small Software Companies, IEEE Software, Published by the IEEE Computer Society, 2007.

[8] Microsoft Corporations, Release Management, Service Management Function 2004, http://www.microsoft.com/err/technet/itsolutions/techguide/msm/ , link tested at 1 Nov 2007

[9] Sommerville I, Software Engeneering , 2007: 498 pp.

[10] Sutton, Jr. S., The Role of Process in a Software Start-up, IEEE Software, July/August 2000.