Core Handover Problems

Ahmad Salman Khan
School of ICT
Royal Institute of Technology (KTH)
Sweden
+46-8-16 28 46
askhan@kth.se

Mira Kajko-Mattsson
School of ICT
Royal Institute of Technology (KTH)
Sweden
+46-8-16 16 70
mekm2@kth.se

ABSTRACT

Even if a handover process is a critical stage in the software lifecycle, little is known about the problems encountered when transferring a software system from development to maintenance. In this paper, we have elicited five core handover problems as faced by five IT organizations today. These are (1) insufficient system knowledge, (2) lack of domain knowledge, (3) insufficient communication, (4) inadequate documentation, and (5) difficulties in tracking changes

Categories and Subject Descriptors

D.2.9 [Management]: Software Process Models

General Terms

Management

Keywords

Deployment, maintenance, documentation, pair programming, system knowledge

1. INTRODUCTION

Handing over a software system from a developer to a maintainer is a critical stage in the lifecycle of a software system. Due to their complex nature, they are marred by many problems which may substantially jeopardize the business of all the parties involved. Despite this, we know little about the types of problems the organizations encounter when transferring software systems from development to maintenance.

In this paper, we elicit the core problems as faced by five IT organizations involved in the handover, alias transition, of a software system from development to maintenance. Our goal is to map out the main handover problems, identify their impact and provide a basis for creating suggestions for their solutions.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

S3M'10, June 21, 2010, Limerick, Ireland.

Copyright 2010 ACM 1-58113-000-0/00/0010...\$10.00.

Presently, we are in the process of exploring the handover process [4], [5], [6]. When doing it, we encounter many problems of identifying its context, scope, dimension and inherent phases and activities. This is because not much research has been done about the handover process. Right now, there are only three process models describing it [1], [7], [9]. They are either too general or too old or they designate handover maturity levels without suggesting their inherent activities. For this reason, on purpose, we only elicit the core problems that might be relevant for all types of handover contexts [5].

The remainder of this paper is as follows. Section 2 describes the method followed when exploring the handover problems. Section 3 briefly presents the handover process. Section 4 presents the problems faced during handover, and finally, Section 5 presents conclusions and suggestions for future work.

2. Method

Our method comprised four steps. These are (1) *interview questionnaire design*, (2) *choice of the organizations studied*, (3) *interviewing*, and (4) *problem compilation*.

During the *questionnaire design* step, we developed a simple semi-structured questionnaire with the goal of identifying and discussing problems as faced during system handover. It consisted of only three questions. These are:

1. *What main problems have you encountered within a handover process?*
2. *What impact have these problems had within your organization?*
3. *What suggestions do you have for remedying the problems?*

In the second step, *choice of the organizations studied*, we chose the organizations involved in our study. The choice was made according to a convenient sampling method. This means that we selected the organization willing to provide us with the information about their problems.

We were able to find five organizations. Due to the sensitivity of the nature of information provided during the interviews, we keep their names confidential. However, as evident from Table 1, the organizations studied are working in diverse domains. They are also of different sizes; two of them are large organizations, two of them are medium-sized, and one is a small organization.

Regarding the large and medium-sized organizations, three of them conduct handover process as part of their development

Table 1. Organizations Studied

Org	Number of Employees	Domain	Role Interviewed	Country
1	80000	Telecom	Project Manager	Sweden
2	50	IT Security	Senior Software Engineer	Sweden
3	40	IT Security	Software Engineer	Netherlands
4	2000	Telecom	IT Development Manager	Pakistan
5	8	Mobile Application	Developer	France

lifecycle processes. They develop and maintain large and complex systems involving separate development and maintenance teams. Regarding the fourth organization, they have experienced a handover process as part of a company acquisition. Finally, regarding the small organization, they develop a small application that is developed and maintained by the same group of developers. Despite this, they were able to provide us with the information on problems as encountered during the handover process where developers handover a software system from self to self.

In the third step, *the interviewing step*, we interviewed the representatives from the companies chosen. Due to the fact that our interviewees were not located in Stockholm, we had no other choice than to conduct the interviews over the telephone. The duration of the interviews varied depending upon the complexity of the organization's handover activities. Two interviews required 50-60 minutes of discussion. One interview was completed within 30 minutes only while the remaining two organizations, due to the complex nature of their handover processes, required a series of several consecutive interviews to be conducted during various points in time.

During the interviews, we used the questions as defined in the first step. These questions, however, only provided a basis for having free discussions, during which we asked additional questions for clarifying the information provided by our interviewees and for getting more in-depth information about the problems.

During the *problem compilation* step, we organized the collected data and mapped out all the elicited problems, their impact and solutions. While doing so, we combined all the problems, analyzed them, identified the duplicates and merged them. When compiling the results, we described the elicited problems according to the following pattern: (1) problem name, (2) problem description, (3) problem impact, and (4) problem solution. It is this pattern that we follow when reporting on the problems in Section 4.

3. Handover Process

Handover is a process during which development team transfers the responsibilities for the system to maintenance team. It is not possible to view software handover in isolation during the life cycle of the system. Activities performed before transferring a software system have great impact on the successful transition. Similarly, postdelivery maintenance activities are significantly affected by the handover phase.

Hence, software handover activities can be divided into three stages. These are (1) predelivery maintenance, (2) handover and (3) postdelivery maintenance. As shown in Figure 1, the handover process is placed between predelivery and postdelivery phases. In reality, however, parts of the handover process are run in parallel with the final phases of predelivery process and the initial phases of the postdelivery process [4], [5], [8].

3.1 Predelivery Maintenance

During the predelivery stage (running in parallel with development), one prepares the software system and the maintenance team for future maintenance. Both development and maintenance teams are involved in the predelivery maintenance. Developers build in maintainability into the system. Maintainers regularly control that maintainability status and prepare themselves for future maintenance. Activities performed during predelivery phase have a great impact on successful transition. [2], [3], [7].

3.2 Handover

During handover stage, the responsibilities for the software system are transferred from development team to the team performing maintenance and support [1], [7], [9]. In this phase software, hardware, data and knowledge is transferred from developer to maintainer [3].

3.3 Postdelivery Maintenance

During the postdelivery phase, one conducts support and maintenance [2], [7]. This is the last stage for handover. According to [7], some handover activities are performed in the initial postdelivery maintenance phase. This mainly concerns developer's support to the maintainer.

4. Handover Problems

We have identified five core problems for the overall handover process. These are (1) *insufficient system knowledge*, (2) *lack of domain knowledge*, (3) *insufficient communication*, (4) *inadequate documentation*, and (5) *difficulties in tracking changes*. In this section, we present these problems. While doing this, we follow the pattern of first describing the problem, then its impact, and finally, the solution as suggested by the organizations studied.

4.1 Insufficient System Knowledge

All the organizations studied have pointed out insufficient system knowledge as a key problem within a handover process. Both support and maintenance teams must possess thorough system knowledge in order to perform their duties after handover.

Figure 1. Software Handover Process [1][2][7][9]

However, the level of the required knowledge differs for both teams. The maintenance team has to possess detailed knowledge down to the code level, since they are responsible for modifying and correcting the system. The support team, on the other hand, needs to be updated about system functionality and about some basic technical issues so that they can keep the system operational on a daily basis and provide support in cases when customers face difficulties when using the system.

In absence of detailed system knowledge, maintenance teams cannot properly make corrections and modifications. Moreover, they cannot anticipate the ripple effect of their changes. Hence, this problem has resulted in many system malfunctions after the modifications have been made in the initial postdelivery phases.

In absence of system operational knowledge, support teams cannot guide customers and resolve their daily issues properly. This has resulted in many customer dissatisfaction problems and customer complaints. As a solution to the problem, the organizations studied have pointed out different approaches to transfer system knowledge.

- Maintenance and support teams should get acquainted with the system as early as possible. According to our interviewees, the teams should start interacting with the system in various ways, such as for instance, by participating in design and code inspections or by developing parts of the system.
- Maintenance and support teams should participate in defining requirements during the requirements elicitation phase. They should also participate in system acceptance testing. In this way, when taking over the system, they already possess essential knowledge required for providing support and maintenance services.
- Transfer of system knowledge becomes more complex in cases when the system handover takes place between different organizations. As a solution to this problem, our interviewees have suggested that a few personnel from development organization should visit the maintenance organization. They should not only share system knowledge with the maintenance organization but also assist them in setting up the maintenance environment and in providing initial maintenance and support.

4.2 Lack of Domain Knowledge

Two of the organizations studied have mentioned that the maintenance team does not possess sufficient domain knowledge. Hence, they cannot properly understand the system. According to these organizations, domain knowledge is a prerequisite for acquiring system knowledge.

In one organization producing IT security products, the maintenance team simply did not possess domain knowledge about basic security concepts such as cryptography, computer security, security certificates and the like. Instead of learning it from publically available literature, they tried to learn it by studying the system to be handed over. They simply did not possess domain knowledge about security, and therefore, they could not properly acquire the system knowledge. It was also difficult for the development team to guide maintainers for how these concepts were realized in the system. As a solution, the organization has suggested that the developers check the maintainer organization's domain knowledge before designating it.

4.3 Insufficient Communication

Developers and maintainers must communicate frequently during the handover phase. This aids them in sharing information, remedying all types of misunderstandings, clarifying assumptions and/or providing quick feedback helpful for troubleshooting and maintaining the system. In absence of such communication, maintainers make changes on the basis of wrong assumptions about the system. All this increases handover duration, cost and substantially decreases customer satisfaction.

Two of the organizations studied have claimed that agile development model has significantly reduced the problem of insufficient communication. The communication gap between the parties involved in handover has been remedied with the daily meeting and pair programming practices. The maintainers get involved in them at the late development phases and throughout the whole handover phase. They become part of teams and pairs. During the daily meetings, they learn most of the issues about the system. Here, they may raise questions and get immediate feedback on them. When pair programming, they not only understand the system from the pairing partner, but also get hands on experience of the system they are going to maintain in the future.

4.4 Inadequate Documentation

All the organizations studied have pointed out the problem of inadequate or insufficient documentation at the moment when the system is being handed over. Documentation is considered as a primary source of information and the first point of reference in case of difficulties with or misunderstandings of the system. Maintainers cannot perform their duties adequately right after the handover process if the system is not properly documented.

The organizations studied have pointed out two types of documentation. These are (1) system documentation and (2) operational documentation. System documentation includes software requirement specification, architecture documents and design documents. Operational documentation includes manuals for installing and operating the system. According to our interviewees, both documentation types are inadequate or insufficient.

Even if the organizations studied develop the documentation by following organization-wide documentation standards, they still fail to deliver adequate and sufficient documentation. This is mainly because they are so much in a hurry to deliver the system so that they wait with completing the documentation. They aim to finish the documentation right after the delivery. As a solution, one organization has suggested to hire technical writers for documenting the system. These technical writers work closely with the development team. The documentation should be understandable and detailed and it should not include unnecessary information so that it becomes difficult to search for relevant information. Another organization keeps the documentation at the same level of version and configuration management as code. Whenever they make changes to code, they are required to update the associated documentation.

4.5 Difficulties in Tracking Changes

Two of the organizations studied have difficulties in tracking changes made to the system, once the system is handed over to the maintenance team. Due to high influx of problem reports right after the delivery, maintenance teams make many changes to the system. These changes, however, are not always traceable by the development teams who need them for integrating in the next system version.

As a solution, the organizations studied use version control repository for storing code and associated documentation. This repository keeps track of changes made to code. Both development and maintenance teams synchronize their code frequently with this repository. One studied organization transfers the system to another organization for continuous development due to company acquisition. In this case, they perform parallel development at both organizations during handover. Hence, tracking changes on both sites becomes even more complex. They have resolved this problem by developing a central system repository to be used by the two parties involved.

5. Final Remarks

In this paper we have elicited five core problems as faced by five IT organizations involved in the handover, alias transition, process. Our goal is to map out the main handover problems, identify their impact and provide a basis for creating suggestions

for how to deal with them. The elicited problems have made us learn the following lessons:

- System knowledge transfer is a core factor for a successful handover. It includes both operational and detailed design and implementation knowledge. This activity becomes more complex if knowledge is transferred between different organizations. As a solution, involvement of maintainers and support personnel during development is of paramount importance.
- It is important to assess the level of domain knowledge possessed by the personnel involved in handover. Technical expertise of the maintainer does not guarantee that he is a suitable candidate for maintaining the system in the future. Domain knowledge is an important prerequisite for acquiring system knowledge which, in turn, is an important prerequisite for succeeding with maintenance. Sometimes organizations neglect this important factor while choosing their future maintenance partners.
- Success of the handover activities may be substantially jeopardized due to lack of or insufficient communication between the participating parties. Developers, maintainers and support personnel need to communicate on a very frequent basis during the handover process. Agile practices such as daily meetings and pair programming have shown to significantly reduce the communication gap between the participating parties.
- Changes made to software system during the handover and postdelivery maintenance phases need to be efficiently communicated among all the parties involved in evolving and maintaining the system. Otherwise, the developers and maintainers may arrive at two inconsistent system versions. As a suggestion, a common version control system and even a common system repository should be used for tracking all changes made to the system.
- Good quality documentation is essential for understanding and supporting the system. However, it is not always correct, consistent and complete at the time of the handover. Here, all the organizations were unanimous on that point. However, the delivery pressure is too high for them to complete the documentation in time. As a suggestion, technical writers should be hired to develop documentation with assistance from development team.

Right now, we are in the process of exploring the handover process. When doing it, we encounter many problems of identifying its context, scope, dimension and inherent phases and activities. This is because not much research has been done about the handover process. For this reason, on purpose, in this initial study, we have only elicited the core problems that might be relevant for all types of handover contexts [5]. Irrespective of whether maintenance stays with development or not, these core problems however indicate a strong need for a predelivery maintenance process and identification of its contexts as well [2], [3], [4].

Even if the problems discussed herein are significant, they are only a tip of the iceberg. Still, we have not sufficiently explored the handover process, its context, dimension, scope and measurement needed for managing it. For this reason, we believe that after having explored the process in greater detail, we should repeat the problem elicitation process again, this time however, to find more granule problems specific for the handover process and its inherent activities.

6. REFERENCES

- [1] April, A., Abran, A. 2008. *Software Maintenance Management, Evaluation and Continuous Improvement*. John Wiley & Sons, ISBN 978-0470-14707-8.
- [2] Kajko-Mattsson, M., Grimlund Glassbrook A., Nordin, M. 2005. Taxonomy of Predelivery/Prerelease Maintenance Activities. In *Proceedings of Conference on Software Engineering and Knowledge Engineering, Knowledge Systems Institute* (2005), 247–252.
- [3] Khan, A.S., Kajko-Mattsson, M., Tyrberg, T. 2009. Comparing EM3: Predelivery Maintenance Model with its Industrial Correspondence. In *Proceedings of IEEE International Conference on Principles of Information Technology and Applications (PITA'09)*, ISBN: 978-83-60810-22-4, IEEE (2009) 573–582.
- [4] Khan, A.S., Kajko-Mattsson, M. Demarcating the Scope of a Predelivery Maintenance Process. In *Proceedings of 5th International Conference on Software Engineering Advances, IEEE* (2010), in press.
- [5] Khan, A.S., Kajko-Mattsson, M. 2010. Demarcating the Scope of a Handover Process. In *Proceedings of 5th International Conference on Software Engineering Advances, IEEE* (2010), in press.
- [6] Khan, A.S., Kajko-Mattsson, M.: Taxonomy of Handover Activities. In *Proceedings of Software Maintenance Maturity Model Workshop (S3M'10)*, in press.
- [7] Pigoski, T. M. 1997. *Practical Software Maintenance*. John Wiley & Sons, ISBN: 0-471-17001-1, (1997), 135–138.
- [8] Tuofaili, H., Kajko-Mattsson, M. 2010. Benchmarking a Software Transition Process. In *Proceedings of Software Maintenance Maturity Model Workshop (S3M'10)*, in press.
- [9] Vollman, T. 1990. Transitioning from development to maintenance. In *Proceedings of Software Maintenance conference*, (San Diego, CA, USA, 1990) ISBN: 0-8186-2091, 189-199.