Applying *EM^3: Handover Framework* in a Project Parking Context

Ahmad Salman Khan, Mira Kajko-Mattsson
School of ICT, KTH Royal Institute of Technology, Sweden
askhan@kth.se, mekm2@kth.se

Summary

A well-defined handover process model is imperative and critical for succeeding with the transfer of a software system from one party to another. Despite this, there still do not exist any up-to date handover process models. Recently, however, we have developed *EM^3: Handover Framework* aiding organizations in constructing their own handover process models. The framework was originally explored within sixty one companies. In this paper, we apply and evaluate *EM^3: Handover Framework* in one Swedish software organization via participatory observation. Our goal is to examine the framework's applicability and usefulness in a real-world industrial scenario. The handover process studied was of a self-to-self type and it was conducted in a project parking context. Our results show that our framework is fully applicable in an industrial handover setting. Almost all of its activities were relevant and fully applied in the context studied.

1 INTRODUCTION

Even if software engineering has been with us for more than five decades, it has still not reached a satisfactory level with respect to the definition and establishment of some of its processes. Certainly, it has explored various ways of developing software systems, and inarguably, it has reaped many development methods and approaches. It has not, however, explored all the corners and edges of the whole software engineering domain, especially the ones taking place after software development. One such a corner is software system handover alias transition, a domain implying a transfer of a software system from one party to another, usually from the party conducting development to the party conducting evolution and/or maintenance [1].

A well-defined software system handover process model is imperative and critical for planning and managing a software handover and for alleviating many handover problems. Failing to transfer a software system may lead to serious consequences such as loss of productivity, loss of maintainer credibility, loss of system and maintenance process quality, and sometimes, even loss of business. Despite this, there still do not exist any up-to date handover process models that designate important process features that are necessary for conducting a systematic and disciplined software system transition. Regrettably, the issue of software handover is still a strongly under-researched and neglected domain. So far, the research community has not paid enough attention to the handover process. The published handover process models are either too old or they are defined on a very general level [2] [3] [1] [4] [5] [6].

Lack of appropriate software system handover process models leads to the fact that companies do not have any process models to follow while performing their handover processes, or if they do have them, then they still may feel insecure whether their models appropriately reflect the complexity of the handover process domain. One such a company is *E-Identity,* a company that has commissioned us to conduct software system handover. Although the company has developed its own handover process model, they still felt very insecure in conducting it in one of their very unique and intricate handover cases. Therefore, they asked us to assist them in their handover process in the role of an expert and supervisor.

In this paper, we report on the results of conducting a handover process at *E-Identity* using our recently developed handover process model – *EM^3: Handover Framework* where *EM^3* stands for *Evolution and Maintenance Management Model*. Our goal was to observe the implementation of our framework and examine its applicability and usefulness in an industrial scenario. The handover process studied was of a self-to-self handover type, it was conducted in a project parking context and its evaluation was made via participatory observation [7].

The remainder of this paper is as follows. Section 2 briefly presents the company and its handover process. Section 3 describes our research process and Section 4 briefly describes *EM^3: Handover Framework*. Section 5 reports on results of the framework's implementation within the company studied and Section 6 analyzes the implementation results. Finally, Section 7 rounds up the paper by providing concluding remarks and suggestions for future work.

2 COMPANY DESCRIPTION

E-Identity is a Swedish company based in southern Sweden. It develops a product for digital identity authentication. In 2012, it encountered a financial crisis which did not allow it to continue developing its product. However, the company was strongly determined to continue with its development as soon as it recovered from the crisis.

As shown in Figure 1, the company's system consisted of two parts. These were *API infrastructure* and *digital identity authentication product*. Different teams were responsible for these parts. The *infrastructure development* team was responsible for developing the API infrastructure and for establishing a platform for the application development. The *application development* team then used the APIs to develop the digital identity authentication product for the end-user customer. Here, the *application development team* was an internal customer to the infrastructure development team.

Before the crisis, the company employed about 25 people. At the moment of writing this paper, the company had to dismiss about 15 people and dissolve all the teams. The people who stayed were (1) three members of the *infrastructure development team* having the roles of *development team lead, project manager and product owner,* (2) three members of the application team having the roles of application developers, network administrator, (3) two company lawyers, (4) CEO, and (5) a secretary. Out of these, four people were involved in the handover process. These were the following:

1. *Development team lead* responsible for product development and maintenance. He had a complete domain and system knowledge. During handover, he documented the domain and system knowledge and generated a stable infrastructure development API release.
2. *Project manager* responsible for managing the handover project. He designed and monitored the handover process and reported on its progress to the product manager.
3. *Product owner* responsible for the product to be handed over. He approved the handover planning, implementation and closure. He also led the daily start-up meetings and made high-level decisions.
4. *Researcher*: responsible for supervising the handover process.

At *E-Identity* the handover process activities are classified under three categories. As shown at the bottom of Figure 1, these are *handover planning*, *handover implementation* and *handover closure*. *Project parking* stage mainly comprises activities dealing with *handover planning* and a few activities dealing with *handover implementation*. *Project resumption,* on the other hand, comprises *handover implementation* and *handover closure* activities.

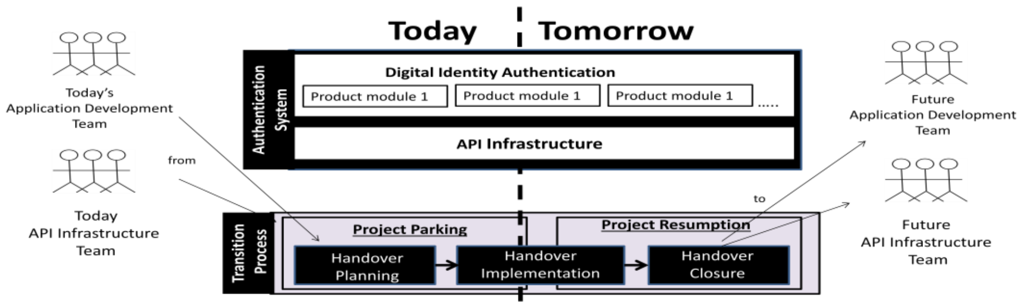

Figure 1. The handover context at E-Identity

3 RESEARCH PROCESS

In this section, we describe our research process. We first present the research approach in Section 3.1 and the research steps in Section 3.2.

3.1 Research approach

In this study, we followed the participatory research approach where we played the role of an active participant [7]. This means that through participating in the process, we tried to understand the handover process studied by actively observing what did happen and what did not happen in the process. We also provided support to the company while implementing the *EM*^{3} practices. In this way, we gained a close familiarity with the process and the people performing the process. To get as much intimacy with the handover process as possible, we used a wide range of data collection methods such as direct observation, active participation, collective discussions, brainstorming sessions, analysis of the organizational documents, and informal interviews. In this way, we could identify similarities and discrepancies between the *EM*^{3} practices and the handover process studied. In other words, we were able to obtain accurate and detailed knowledge about the handover process, and thereby, we were able to evaluate its usefulness and applicability in an industrial setting.

3.2 Research steps

The handover process studied took three weeks to perform and the duration of our participatory observation research corresponded to the part of the handover process corresponding to the *project parking* phase. During this time, we conducted four major steps that are typical of a participant observation method [7]. These were (1) *Establish Rapport,* (2) *Acting in the Field,* (3) *Recording Observations*, and (4) *Analyzing Data*.

The first phase, the *Establish Rapport* phase, lasted for only one day. We visited the company studied, we acquainted ourselves with the company's employees and acquired some introductory information about the company's situation. Here, we had a meeting with the project manager who provided us with the big picture of the organization, the system and the processes used for managing the system. We then spent the remaining time of the day on becoming friends with the company's employees. We strongly believed that establishing a good relationship would promote cooperation.

In the *Acting in the Field* phase, we tried to act just as the company's "*local*" member with some minor exceptions [7]. In addition to acting as a local, we had to get a thorough understanding of the company, its product and processes. For this reason, we studied all the organizational documentation that was relevant and available. Just because not much documentation was in place, we continued our study via informal discussions with the company's employees.

As an active team member, or now as a *local*, we worked on implementing the handover process using *EM*^{3}*: Handover Framework*. Being fully integrated with the company's transition team, we played the role of *an active participant observer.* Here, we conducted all kinds of activities starting from participating in the morning startup meetings, planning the handover process, conducting and monitoring the handover process, documenting the process, brainstorming, and supporting the company's management in decision making.

The third phase, the *Recording Observations* phase ran in parallel with the *Acting in the Field* phase. While observing the process, we followed the implementation of almost all of the *EM*^{3} framework practices, compared the framework's activities with the company's handover activities and evaluated their applicability. Wherever it was relevant, we suggested improvements in the light of *EM*^{3} *Framework.* This helped the company to cover the gaps in their handover process and helped us gain feedback for improving the *EM*^{3}*: Handover Framework.*

Regarding the *EM*^{3} activities that could not be implemented in the process studied, we first asked whether the company conducted them in other handover cases and inquired about their usefulness using mainly semi-structured interviews. We also conducted the interviews with the purpose of finding the reasons behind the handover process studied, its main challenges, and the actions that could be taken to remedy the challenges. Finally, in the *Analyzing Data* phase, we studied each of the *EM*^{3} activities in order to find out whether it was fully or partially implemented and to find out reasons for their non-adherence to the executed handover process. It is these findings that constitute the contribution of this paper.

4 EM^3: HANDOVER FRAMEWORK

EM^3: Handover Framework provides a skeletal structure of six different parts that are necessary for creating handover processes. It is a result of an explorative study made in 61 companies [8]. As shown in Figure 2, its central part is *EM^3: Handover Taxonomy* – a set of component practices including the activities that play a significant role in executing a handover process. The taxonomy activities may be used for orchestrating handover processes using the framework's other five parts such as (1) *Handover Types* designating three types of software handover, (2) *Handover Contexts* placing handover within software lifecycle, (3) *Handover Roles* identifying the main responsibilities in the handover process, (4) *Handover Lifecycle Roadmap* designating time spaces in the handover lifecycle phases, and (5) *Handover Guidelines* providing support to the organizations in their handover endeavors. Due to the space restrictions, we do not explain each part of the framework. We rather focus on *EM^3: Handover Taxonomy* - the core part of *EM^3: Handover Framework.*

4.1 EM^3 Handover Taxonomy

EM^3: Handover Taxonomy comprises eight practices important for implementing software system handover. These are (1) *Management and Administration*, (2) *Maintenance Environment*, (3) *Version and Configuration Management*, (4) *Training*, (5) *Deployment*, (6) *Documentation*, (7) *Maintainability Management,* and (8) *Software System Transfer*. In this section, we briefly describe them and their constituent activities. To be able to follow our descriptions, we strongly advise our reader to follow the *EM^3* activities in Table 1.

4.1.1 Management and Administration

The *Management and Administration (MA)* practice includes the activities required for handling and controlling the handover process. The success of the overall handover process strongly depends on this practice. As shown in Table 1, the practice contains activities starting from planning a handover process, to managing it and to, finally, evaluating it postmortem.

Before starting transition, the organizations should identify its type and complexity. For instance, transition might be of a self-to-self type where the transitioners are the transitionees or it might be of an external type where the transitioners and transitionees are separate teams or organizations. The transition might be of a high complexity implying a handover of a large safety critical system among several organizations or it might be as simple as a self-to-self handover of a newly developed small system version (see Table 1).

As a next step, the transition team should create a transition plan and assure that important management plans supporting the transition process are in place. Being guided by parameters such as, for instance, transition deadline, resource constraints and the like, the transition plan should define transition manpower resource requirements, budget and schedule. The management plans, on the other hand, should plan for the processes that interact with the transition process. Examples of them are development, maintenance, quality management, to mention a few. A communication model should be

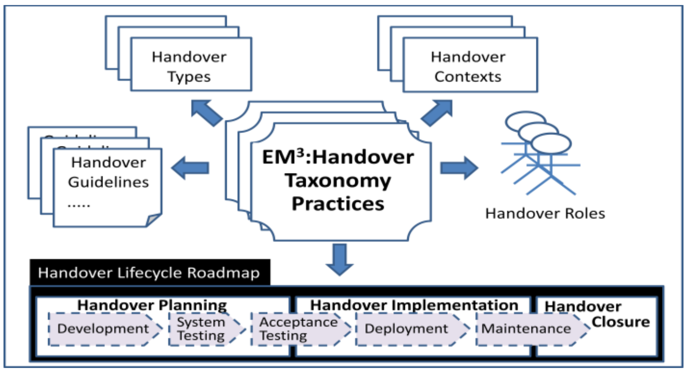

Figure 2. EM3: Handover Framework

in place for interacting and transferring knowledge between different parties participating in the handover process. Finally, the handover process should be continuously monitored and it should be evaluated postmortem for future improvements.

Determining the transition type and complexity is a prerequisite for defining a transition strategy, for establishing a transition team, for defining a transition process, and for designating a transitionee. A transition team should from now on manage and administer the transition process. It should enlist all its core activities, and the activities that are part of other processes, the processes that either impact or are impacted by the transition. Failing to identify them may jeopardize the whole transition process. Finally, all the stakeholders involved, including the transitionees, should agree upon the design of the transition process to be executed.

4.1.2 Maintenance Environment

The transitionee has to have the environment that is right from the beginning. Hence, the *Maintenance Environment* (ME) practice includes the activities that are required for determining the needs for hardware suites, software suites and maintenance support suites and activities required for their installation (see Table 1).

The needs should be determined in advance in cases one transfers a newly developed system. In other cases, the current suites should be assessed whether they still fulfill their function. Here, one should identify their potential adequacies and deficiencies and assure that they are compatible across all the environments, that is, the environments of the transitioners, transitionees and of the customers. If the suites are not determined or assessed in advance, then there is a risk that they will not be delivered on time, that the transitionee will not get enough time for learning them or that they may face compatibility problems, and thereby, waste valuable time. This, in turn, implies a risk that the customer will not get efficient support during the initial phases of system operation.

4.1.3 Version and Configuration Management

The *Version and Configuration Management* (VCM) practice includes the activities required for keeping track of the changes made to a software system before, during and after handover. The practice is critical for assuring that the system that has been handed over includes the right components that are compatible with the components on the transitioner, transitionee and customer's sites. As shown in Table 1, it deals with placing the system under version and configuration management and it deals with managing baselines.

It goes without saying that it is significant to baseline the software system to be handed over. In the context of a system handover, at least two groups of baselines are relevant. These are test and postdelivery baselines. The test baselines are created before the delivery of the system during different testing phases. They constitute platforms for identifying and tracking all the changes made to the system and for making important decisions on the delivery and handover. The postdelivery baselines, on the other hand, are created just after the system delivery. They constitute important platforms for synchronizing the changes across the development, maintenance and operational environments and for assuring that they have identical or as identical as possible system copies.

4.1.4 Training

People involved in handover must be properly trained so that they will be able to work with the system from the very first day after handover. For this reason, as shown in Table 1, the *Training* practice focuses on training planning, creating training material and on providing training. The trainees are mainly maintenance and support teams, acquirer and the topics to be taught are system structure and operation, maintenance and support processes, and technology. In addition, the practice advocates early involvement of the maintainers in attending to modification requests, performing white-box testing and onsite support as an alternative form of training.

To ensure that the training is effective, the practice also designates the roles responsible for the training process. These are (1) the role responsible for managing the overall training process and the (2) role responsible for providing the training. Finally, the component practice suggests that the educational policies be developed and followed by all the handover projects.

4.1.5 Deployment

Deployment is a critical prerequisite step for commencing software operation and maintenance. As shown in Table 1, the *Deployment* practice includes activities starting from defining release scope and contents to preparing for installation, to installing and deploying the system, to finally, closing the deployment and planning for future releases.

The starting steps within deployment are identifying the scope and contents of the release and creating a team responsible for it. Afterwards, preparations for deployment include development of installation procedures, identification of the stakeholders impacted by the release, preparations of the release and build documentation and updates of access rights. It is especially important that installation procedures are in place to back out releases in case of failures. Organizations cannot afford to keep their systems shut down for longer periods of time while installing new releases. Finally, the system deployment is reviewed and a decision is made on the deployment closure. In parallel with the above-listed deployment activities, it is important to plan for future releases. The planning includes activities for identifying features to be deployed in the next release, activities for determining the impact of the externally acquired components, activities for estimating release effort, size and time, and the activities for defining hardware/software infrastructure requirements.

4.1.6 Documentation

The *Documentation* component practice focuses on establishing a system documentation repository and mechanisms for controlling its status. Both developers and maintainers need a central location for storing software system documentation and for assuring that nothing gets lost while handing over a software system. As shown in Table 1, the practice includes (1) activities for establishing a system documentation repository providing rudimentary services such as *access, store, update, remove, grant access rights*, (2) activities for subjecting the documentation repository to SCM, and (3) mechanisms for controlling the status of the system repository. The practice also suggests that the documentation standards be established, and finally, that the documentation repository be handed over to the transitionee.

4.1.7 Maintainability Management

The *Maintainability* practice includes assessment of two types of maintainability: (1) *system maintainability* referring to the ease with which one changes the system, and (2) *data maintainability* referring to data integrity, correctness and consistency. If the system is not maintainable, it then becomes difficult for the maintenance team to understand, and thereby, difficult to evolve and change. If the data is defective or inconsistent, then the company may encounter the problem of a critical data loss.

Both maintainability types must be assessed before system handover. As shown in Table 1, the company must define appropriate system and data maintainability attributes, define rules for adhering to them, identify milestones for assessing them, and finally, assess them. After finalizing the handover process, the organizations should assess their procedures for managing and controlling data and system maintainability.

4.1.8 Software System Transfer

The *Software System Transfer* practice was added to *EM^3: Handover Taxonomy* during this study. While observing the company's process, we realized that our taxonomy missed the practice of keeping track of the status and readiness of the software system to be handed over. As shown in Table 1, the *Software System Transfer* practice focuses on monitoring the status of software components, managing modification requests, and on actually transferring the system.

During the development and testing phases, one should continuously monitor the progress of the system. Its components may still be either under development, testing or they may be ready for deployment. Continuous monitoring of system components is pivotal for evaluating the system status and for making decisions on whether to hand it over or not.

To be able to monitor the system status, its changes or the need for system changes, the practice suggests creating a template for managing information about modification requests and for placing the modification requests in a modification request repository. The status of modification requests must be monitored during the handover phase. Their severity and complexity will then provide a basis for decision making.

The last major activity of the practice includes the transfer of the system components, of the operational data and of all the unresolved modification requests. Regarding the operational data, the maintenance team needs the data that was operated on while the problem got encountered. Hence, it is enough that they have access to its replica only. They do not need to access its master copy. Providing the maintenance team with the master copy only implies the risk of unintentional corruption. Finally, the practice advocates the monitoring of the system after the handover and the signing off of the software system handover after approval of all the parties involved.

5 STATUS

In this section, we present the course of implementing *EM$^{3:}$ Handover Taxonomy* at *E-Identity*. When doing it, we try to follow the order of the activities as listed in the Tables 1. Sometimes however, for presentational reasons, we had to change their order. Due to space restrictions, we could not report on the implementation of all the *EM^3* activities. Therefore, we only report on the most important ones. For more information, the interested readers are welcome to study [8] . Finally, while participating in the handover process, we could not observe the implementation of all the *EM^3* activities. Some of the *EM^3* activities were not relevant in the handover context studied. However, to be able to evaluate those *EM^3* activities, we inquired about their applicability and usefulness in other handover processes that the company had experienced. To distinguish them from the observable ones, we mark them with +(i) standing for “inquired and performed”.

5.1 Implementing the Management and Administration practice

E-Identity has implemented almost all the activities listed in the *Management and Administration* practice. As shown in Table 1, we could observe the implementation of all except for two activities. At the moment of writing this paper, the company could not evaluate the transition process postmortem due to the fact that the transition project had not been finalized yet. Neither could it define any additional manpower resources required for the whole transition process. Due to financial reasons, their resources were restricted to simply what they had.

Concerning all the other activities in the *Management and Administration* practice, they were all followed by the company studied. *E-Identity* experienced a self-to-self type of handover and, due to unavailability of the transitionee, it deemed the transition process to be of a very complex nature. For this reason, their transition strategy focused on the following four sub-strategies: (1) *Strategy 1* determining the future transitionees, (2) *Strategy 2* designating a future transition team, (3) *Strategy 3* designing the transition process, and (4) *Strategy 4* establishing ways of transferring knowledge.

Regarding the design of the transition process (*Strategy 3*), the company decided to structure it into two phases: (1) the *project parking* phase to be conducted by the transitioners and (2) the *project resumption* phase to be conducted by the transitionees. At the moment of writing this paper, the *project parking* phase got finalized and the *resumption phase* had not yet started. However, the company had outlined an overall transition process that would facilitate the future project restart. The process included the activities for preparing for the project restart and for restarting the project. The preparations included preparing the organization and its product and processes for the project parking and resumption, recruiting transitionees, and training them.

According to *Strategy 2*, not the whole transition team could be designated in advance. Part of the team was created for conducting the *project parking* phase. This part comprised project manager, product owner, development team lead, and researcher. Another part of the team will be created in the *project resumption* phase in the future. The members that will stay on this team will be project manager and product owner. The members that will leave the team will be the development team lead and researcher. In addition, some of the new hires will join the transition team. According to *Strategy 1*, the future transitionees will be consultants instead of fixed-term employees. The company claims that this will substantially reduce project restart time and cost.

Regarding *Strategy 4,* the strategy concerning the transfer of knowledge between the transitioners and transitionees, the company was aware of the fact that the two groups would not have the opportunity to communicate with each other. For this reason, *Strategy 4* dealt with creating a documentation of the company’s products, processes, and technology. The documentation would constitute the main channel of communication.

Overall, the transition process studied was carefully planned and continuously monitored. Its plan focused on determining current and future manpower resources, budget required, facility resources and training effort and time required for parking and resuming the project. The estimated manpower resources for resuming the project were five to eight developers and the estimated time for resuming development was two months. The facility resources would not change; the new team would use the company’s current facilities.

5.2 Implementing the Maintenance Environment practice

E-Identity has implemented all the activities listed in the *Maintenance Environment* practice. We had the opportunity to observe the implementation of all except for one activity, the activity of granting the transitionee access permission to hardware/software suites. This is because the transitionee has not yet been designated.

The implementation of all the *Maintenance Environment* activities went very smoothly, mainly thanks to the fact that the transition took place within one and the same company. The hardware and software suites and maintenance support suites were already determined and installed. The company did not need to determine any new suites. Neither did they need to assure that the suites matched each other. They all did by default.

Regarding the matching of the company’s suites with the customer’s suites, it was the product owner who assured that the company’s hardware and software suites matched for all the groups involved. However, while assessing the hardware/software and maintenance support suites, the company encountered two deficiencies. First, the hardware/software suites were not efficient enough. Therefore, the company decided to migrate their system to another application server – JBoss [9], with the purpose of achieving better efficiency soon after the project resumption. As a result, they had to re-assess their current hardware and software suites to make sure that they were compatible with JBoss. Regarding the maintenance support suites, when assessing them, the company had discovered that some of the development tasks that had been shown as resolved in Jira (a tool managing requests) were not integrated into the main branch in their central version control repository – Git [10], the repository in which the code was stored. The solution was only available on the developer’s local machines. It was decided that the *project resuming team* would attend to this deficiency in the future.

5.3 Implementing the Version and Configuration Management practice

The company has implemented all except for two *EM*^{3} activities for managing version and configurations. The activities that have not been implemented concerned the identification and tracking of the customizable configuration items during handover. The reason for why they were not implemented was the fact that the company had only one customer. Hence, they did not experience any customization needs.

Regarding the activities that got implemented, we only had the opportunity to observe an accomplishment of a subset of them. As indicated in Table 1, we observed the complete accomplishment of Activities VC1 and Activity VC 2.1, that is, the activities concerning the management of versions, configurations and baselines. We did not have however the opportunity to observe the accomplishment of Activity VC2.2 concerning the establishment of post-delivery baselines. Nonetheless, we inquired about how they were performed in normal handover cases.

The company establishes four baselines: *developer test, system test, acceptance test,* and *deployment baselines*. Due to the fact that its developers continue with maintenance, the activity of assisting developers in attending to problem reports during acceptance testing was not relevant. Developers attend to all problems during both system and acceptance testing. Finally, the company identifies and tracks configuration items, keeps track of the changes made to the baselines, and notifies its relevant stakeholders about the changes made.

While following the handover process at E-identity, we observed an additional baseline that we had not recognized in our model. It is a release baseline. It is a separate release branch that is created during deployment. It includes all the changes made to the software system during deployment. Its changed code is then merged with the main branch which is then synchronized with the all the development, maintenance and operational branches.

Whenever problems arose during the handover phase, the transition team discussed their nature and made decisions on whether to resolve them or not. Problems of critical nature were urgently attended to whereas problems of less serious nature were to be resolved after the handover. Finally, the project manager created post-delivery baselines for operation and maintenance of the system.

5.4 Implementing the Training practice

Training was regarded as one of the most important practices of the company's handover process. Hence, all the EM^3 training activities had been implemented. However, as shown in Table 1, at the moment of conducting this study, the company only implemented the training activities from T1 to T5.1, the activities focusing on the designation of the role responsible for the training, on planning of the training, and on parts of preparing training material. Regarding the remaining activities, the company will perform them in the *project resumption* phase. Its trainees are the transitionees and the planning for their training focused on creating a thorough system and process documentation on different granularity levels. Since their product is an electronic notary public, the company had also created training material on legal constraints.

The future trainee group would constitute staff of five to eight developers and the future trainer would be the transitionee himself and the project manager, the only employee who has stayed. With this, we mean that the new hires will be highly responsible for self-educating themselves under the mentorship of the project manager. Their training will comprise studying practically everything starting from acquiring an overview of the system, overview of the company and of its processes and ending up on performing various tasks such as studying documentation, attending to modification requests, writing test cases, exercising the system, and so on. All this will be based on the training material that corresponds to the documentation that had been created or updated during the handover process. It was also planned that the initial tasks would be simple in nature. They should help the transitionee to gradually understand the system. The company estimated that each transitionee would require two months to learn the system in order to start working productively.

5.5 Implementing the Deployment practice

All the deployment activities as defined in the *Deployment* practice have been implemented by the company studied, that is, the company has defined and planned the scope and type of the releases, defined installation procedures, installed the system, closed the deployment and planned for future releases. At the moment of our study, we did not have the opportunity to experience the full deployment process to an external customer. We only observed the internal deployment process.

The company had two types of deployment: (1) *internal deployment* of infrastructure API transferred from the *infrastructure development team* to the *application development team,* and (2) *external deployment* of a ready application from the *application development team* to its external end-user customers. The main reason for conducting the internal deployment during handover was to provide an updated and stable version of API to the *application development team* before freezing the system. The *application development team* would then continue their work on developing the application after project parking.

The steps in the internal deployment process studied were (1) establish a deployment branch for the release, (2) compile and verify the deployment branch by performing deployment readiness tests, (3) make changes to the deployment branch code to solve the problems encountered during testing, (4) integrate those changes in the main branch, (5) de-install the former system version, (6) install the new system version, and, (7) install the operational data. Finally, the company closed the deployment by reviewing the whole deployment process and by making sure that it ended in a correct manner.

5.6 Implementing Documentation practice

The company had implemented all the activities in the *Documentation* practice. As indicated in Table 1, some of the activities were however partially accomplished. These concern defining organizational policies for developing documentation standards and creating mechanisms for controlling the quality of system documentation. The reason is the fact that the development team gave priority to meet the delivery deadline, and hence, they put less emphasis on documentation quality. Therefore, the documentation was outdated with low quality before the beginning of the project parking phase. The company realized this during the project parking phase, and therefore, decided to develop the documentation standards in the future.

Some other activities could not be observed while conducting our study. These concern sharing documentation standards and documentation repository with the transitionee. The reason is the fact that the transitionee has not been decided yet. However, in normal handover cases, the company shares the repository by default due to the fact that the transitioner was the same as the transitionee. The activities that we could observe concerned the establishment of system documentation repository, definition of the services to be provided by the repository, subjecting the repository to SCM and the sub-activities of the establishment of the documentation standards. Finally, the company had no mature documentation standards. At the moment of conducting this study, they had no other standards than the templates for describing modification requests.

5.7 Implementing Maintainability Management practice

The company has not fully fulfilled the *Maintainability* practice component. As indicated by Table 1, it has not defined any procedures for assessing data maintainability. They claim that the reason is the fact that the system is not yet fully operationalizable. Hence, it does not have any operational data to consider.

The company has only partially defined procedures for assessing system maintainability. This means that it has defined various quality attributes concerning mainly architectural design and coding standards, however, it has not documented them. The definition of system maintainability lies in the heads of the employees instead. Despite this, the company uses it on a regular basis while assessing the system state at the end of each development tollgate.

Finally, at the moment of conducting our study, the company realized that one important maintainability attribute was missing. It concerned the traceability between the system documentation and code. The system documentation played the most important role in the company's handover process. It was a prerequisite for resuming system development and it was the only source of information for the *project resumption team*. For this reason and for the reason of attending to the traceability problem, the company revised all the documentation and made sure that it conveyed as much information as possible about the system and its status and that it could be easily traced across its code and documentation.

5.8 Implementing Software System Transfer practice

The *Software System Transfer* practice was added to *EM^3: Handover Framework* during this study. Just because its activities mirror the activities that were conducted in the company studied, the company has implemented all of them. However, as shown in Table 1, at the moment of conducting this study, we were able to observe the implementation of only Activities ST1 to ST3, that is, monitoring of the status of system components, making decisions on the components to be handed over and managing modification requests. We could not observe the implementation of Activity ST4 dealing with the actual system transfer. The reason is the fact that the transitionee has not yet been designated, hence, this activity would take place in the *project resumption phase*.

From the handover perspective, the company distinguished between three types of system components. There were (1) stable components, (2) components under testing, and (3) components under development. The stable components were ready to be used in the system. The components under testing would need to undergo low and high-level testing. The components under development would need to be developed from scratch. All these components were managed with the already-mentioned modification requests managed by Jira.

6 ANALYSIS

A quick scan through Tables 1 shows that almost all of the EM^3's activities have been implemented in the handover process studied. Not all of them however were directly observable due to its specific handover case. The activities that could not be observed either concerned general prerequisite handover activities or the activities to be performed in the *project resumption* phase; the phase that the company has not performed yet. Out of the total of EM^3's activities, 66% could be directly observed and 21/% were inquired about. Only 6% were partially performed and as few as 4/% were not performed at all. Finally, 2% of the activities were not applicable and 1% of the activities was not relevant.

Except for a new component practice, *Software System Transfer*, and its activities, our study has not led to any additions of new activities. It has rather led to the confirmation that almost all the *EM^3*'s activities were easily applicable in the handover context studied. It has also helped us identify a new context of a handover process where transitioners will never have the opportunity to learn to know the transitionees.

In this study, we have distinguished between two groups of practices: (1) the practices that contributed to the confirmation that they were applicable and relevant in the context studied, however, they did not lead to any new knowledge or lessons learned, and (2) the practices that both confirmed their relevancy and applicability and also contributed to new knowledge and lessons learned.

To the first group belong *Maintenance Environment*, *Version and Configuration Management*, and *Maintainability Management.* Here, however, we could confirm that clear evidences and consequences of non-synchronization of code at developer's machine with the central version control repository could lead to a series consequence of losing important code changes. Moreover, we could confirm that separate baselines should be established for making changes in the code without altering the master baseline and that the presence of issue tracking software was significant for handover. It provides an overall picture about the existing problems in the system. To the second group belong *Management and Administration*, *Training*, *Documentation,* and *System Transfer*. They have helped us to learn the following lessons:

- In all transition contexts, one should designate a transition team. Initially, such a team is not complete, but it at least includes a set of representatives from both the transitioners and transitionees' sides. In the context of our study, however, the team included only the representatives from the transitioner side. The transitionee was not yet known. For this reason, the only communication channel that was possible was a very detailed documentation of the company's products, processes, and technology. There was no other way for those two groups to communicate.
- The specific context of the handover process studied forces the transitionee to be both the trainer and trainee. This means that the new hires will be responsible for self-educating themselves using the documentation created during the *project parking* phase.
- Handover does not only take imply a transfer from the parties involved in development to the parties involved in maintenance. It may very well take place internally between two development teams where one team develops a development platform to be then used for developing a system by another team.
- During handover, it is important to keep track of the software system and the health and progress of its components. For this reason, one needs to clearly distinguish between (1) stable components, (2) components under testing, and (3) components under development. Only then one may make decisions on their deployment, and thereby, handover.
- We noticed that the status of modification requests was decisive in handover. The magnitude and nature of modification requests aids in revising the handover decision.
- Finally, we identified new documents that were especially created for the unique handover context studied. The included creation of *project startup procedure*, *project status* and *developer skill set* documents. These documents would help the future *transitionee* to understand the system status and restart the project in an optimal way.

7 Epilogue

In this paper, we have reported on the results of implementing the taxonomy activities inherent in *EM^3: Handover Framework.* Our goal was to observe their implementation and examine its applicability and usefulness in a real-world industrial scenario. The handover was of a self-to-self handover type and it was conducted in a context where the company studied was forced to park a development project due to financial problems.

Even though *EM^3: Handover Framework* has been originally explored within sixty one companies and has shown to be useful in this study, we strongly advise the software community to continue to explore the handover domain and to evolve our framework. More handover contexts need to be explored and more studies need to be done to evaluate *EM^3: Handover Framework* and assure its usefulness and accuracy in an industrial setting. We believe however, that this study has already provided evidence that *EM^3: Handover Framework* is on the right path towards providing a fully-fledged support for creating handover process models.

References

[1] T. Pigoski, Practicle Software Maintenance: Best Practices for Managing Your Software Investment, John Wiley & Sons, 1996.

[2] T. M. Pigoski och C. S. Looney, ”Software Maintenance Training: Transition Experiences,” 1993.

[3] T. M. Pigoski och J. Sexton, ”Software Transition: A Casestudy,” i *International Conference on Software Maintenance ICSM*, 1990.

[4] I. O. Standardization, ”ISO/ IEC 15288, Systems and software engineering- System life cycle processes,” IEEE, 2008.

[5] I. O. f. Standardization, ”ISO/IEC 14764:2006, Standard for Software Engineering- Software Life Cycle Processes- Maintenance,” *IEEE,* 2006.

[6] T. Vollman, ”Transitioning from development to maintenance,” i *Conference on Software Maintenance*, San Diego, CA, 1990.

[7] J. T. Howell, Hard Living on Clay Street: Portraits of Blue Collar Families. Prospect Heights, Illinois, Waveland Press, Inc, ISBN 0881335266, 1972.

[8] A. S. Khan, A Framework for Software System Handover, Stockholm: KTH, Software and Computer systems, SCS, http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-122270, ISBN:978-91-7501-739-6, 2013.

[9] JBoss, ”JBoss Applicaiton server,” JBoss, 2013. [Online]. Available: http://www.jboss.org/.

[10] G. Hub, ”Git open source distributed version control system,” Git, 2013. [Online]. Available: http://git-scm.com/.

[11] Atlassian, ”Jira Project Tracker,” Atlassian, 2013. [Online]. Available: http://www.atlassian.com/software/jira/overview.

Table 1. EM^3: Handover Taxonomy practices with activities part 1

+ stand for observed and performed, +(i) stand for inquired and performed, -- stands for not performed, P stands for partially performed and NA stands for not applicable. Plan stands for handover planning, Impl stands for handover implementation and Clos stands for handover closure.

Activities	Status	Phase
MANAGEMENT AND ADMINISTRATION		
MA 1: Determine/redetermine type and complexity of transition	+	Plan
MA 2: Define a strategy for transition process	+	Plan
MA 3: Designate a transitionee	+	Plan
MA 4: Establish a transition team	+	Plan
MA 5: Define a transition process	+	Plan
MA 5.1: Identify core transition activities	+	Plan
MA 5.2: Identify activities of other processes that impact or are impacted by the transition	+	Plan
MA 6: Agree upon the executed transition process	+	Plan
MA 7: Create/adjust a transition plan	+	Plan
MA 7.1: Define/adjust parameters guiding the design of the transition plan	+	Plan
MA 7.2: Create the transition plan using the parameters	+	Plan
MA 7.3: Define transition resource requirements	+	Plan
MA 7.3.1: Define manpower requirements	+	Plan
MA 7.3.1.1: Define maintenance manpower requirements	+	Plan
MA 7.3.1.2: Define developer manpower resources	+	Plan
MA 7.3.1.3: Define transition team manpower resources, if any	+	Plan
MA 7.3.1.4: Define other manpower resources, if any	NA	Plan
MA 7.3.2: Define maintenance facility requirements	+	Plan
MA 7.4: Determine transition budget	+	Plan
MA 7.5: Create a transition schedule	+	Plan
MA 8: Develop management plans necessary for transition	+	Plan
MA 9: Determine a communication model to be used within transition	+	Plan
MA 10: Monitor the transition process	+	Impl
MA 11: Evaluate the transition process postmorten	+	Clousre
MAINTENANCE ENVIRONMENT		
ME 1. Manage hardware/software suite needs	+	Plan
ME 1.1: Determine hardware/software suite needs	+	Plan
ME 1.1.1: Determine hardware and software packages constituting the hardware/software suites	+	Plan
ME 1.1.2: Assure that hardware/software suite needs match the developer's hardware/software suites	+	Plan
ME 1.1.3: Assure that hardware/software suite needs match the customer's hardware/software suites	+	Plan
ME 1.2: Install hardware/software suite	+	Impl
ME 1.3: Grant the transitionee access permission to hardware/software suites	+(i)	Impl
ME 1.4: Assess current hardware/software suite, if any	+	Plan
ME 1.5: Remedy the deficiencies in hardware/software suite , if any	+	Impl
ME 2: Manage maintenance support suite	+	Plan
ME 2.1: Determine maintenance support suite	+	Plan
ME 2.1.1: Determine software packages constituting the maintenance support suite	+	Plan
ME 2.2: Install maintenance support suite	+	Impl
ME 2.3: Assess maintenance support suite	+	Plan
ME2.4: Remedy the deficiencies in maintenance support suite , if any	+	Impl
VERSION AND CONFIGURATION MANAGEMENT		
VC 1: Manage version and configuration	+	Impl
VC 1.1: Define rules to uniquely identify, name and label the configuration items and their relationships	+	Plan
VC 1.2: Define how the configuration items are to be selected, grouped and classified	+	Plan
VC 1.3: Decide on how to identify and track changes made to customizable configuration items during handover	NA	Plan
VC 1.4: Put software under configuration management	+	Impl
VC 1. 5: Place software under version control management	+	Impl
VC 2: Manage baselines	+	Impl
VC 2.1: Establish test baselines (system test baseline, acceptance test baseline)	+	Impl
VC 2.1.1: Assist developers in attending to problem reports during acceptance testing	NR	Impl
VC 2.1.2: Identify and track customizable configuration items during handover	NA	Impl
VC 2.1.3: Keep track of the changes made to the baselines	+	Impl
VC 2.1.4: Notify all the stakeholders involved about the changes made to the system	+	Impl
VC 2.2: Establish post-delivery baselines (operational baseline and maintenance baseline)	+(i)	Closure
VC 2.2.1: Check whether the reported problems are not of critical nature	+(i)	Closure
VC 2.2.2: Synchronize system changes made during system handover in all the environments (operational, development and maintenance)	+(i)	Closure
VC 2.2.3: Assure that the identical copies (or as identical copies as it is possible) are installed in the operational, development and maintenance environments	+(i)	Closure
VC 2.2.4: Accept and approve the system for operation and maintenance	+(i)	Closure
TRAINING		
T 1: Designate the role responsible for managing the training process	+	Plan
T 2: Plan training	+	Plan
T 2.1: Identify training topics to be taught (e.g. system, maintenance process, support process, technology, legal aspects)	+	Plan
T 2.2: Identify the trainee groups	+	Plan
T 2.3: Determine training needs of each trainee group with respect to the training topics	+	Plan
T2.4: Define methods of training	+	Plan
T 3: Create/update training material	+	Plan
T 4: Identify the role responsible for providing the training	+	Plan
T 5: Prepare for training	+	Plan
T 5.1: Adapt the training material to the trainee group and its needs	+	Plan
T 5.2: Setup training environment, if required	+(i)	Plan
T 6: Provide training	+(i)	Impl
T 7: Involve maintainers in attending to modification requests	+(i)	Impl

Table 1. EM^3: Handover Taxonomy practices with activities part 2

+ stand for observed and performed, +(i) stand for inquired and performed, -- stands for not performed, P stands for partially performed and NA stands for not applicable. Plan stands for handover planning, Impl stands for handover implementation and Clos stands for handover closure.

Activities	Status	Phase
T 8: Involve maintainers in white box testing and debugging	+(i)	Impl
T 9: Provide onsite support, if needed	+(i)	Impl
T 10: Develop educational policies providing guidance for developing educational plans	+(i)	Plan
T 11: Develop project specific educational plan using policy guidelines	+(i)	Plan
DEPLOYMENT		
DP 1: Define/continuously re-define the scope and contents of the release	+	Plan
DP 2: Determine type of release (major/minor)	+	Plan
DP 3: Create a deployment team	+	Plan
DP 4: Develop installation procedures	+	Plan
DP 4.1: Develop rollback procedures	+	Plan
DP 4.2: Develop installation manuals	+	Plan
DP 4.3: List organizations and stakeholders affected by the new release	+	Plan
DP 4.4: Prepare release and build documentation	+	Plan
DP 4.5: Define/continuously update the access rights to release components	+	Plan
DP 5: Installation	+	Impl
DP 5.1: Take a backup of the system release to be de-installed	+	Impl
DP 5.2: Perform deployment readiness test	+	Impl
DP 5.3: Distribute and deliver the system and /or system components at a correct location and time	+	Impl
DP 5.4: Install the new system version	+	Impl
DP 5.5: Install operational data	+	Impl
DP 5.6: Record any incidents, unexpected events, issues or deviations from the release plan	+	Impl
DP 5.7: Perform deployment verification tests	+	Impl
DP 6: Deployment Closure	+	Clos
DP 6.1: Review the system deployment	+	Post
DP 6.2: Close the deployment	+	Post
DP 6.2: Close the deployment	+	Post
DP 7: Planning for future releases	+(i)	Plan
DP 7.1: Plan updates of future releases	+(i)	Plan
DP 7.1.1: Identify features to be deployed in the next release	+(i)	Plan
DP 7.1.2: Determine the impact of the externally acquired components on the planned release and vice versa, if relevant	+(i)	Plan
DP 7.1.3: Estimate release size, effort, time and hardware/software infrastructure required	+(i)	Plan
DP 7.2: Determine the system distribution structure	+(i)	Plan
DP 7.3: Determine forms of deployment software	+(i)	Plan
DOCUMENTATION		
D 1: Establish a system documentation repository	+	Plan
D 2: Define services to be provided by the system documentation repository	+	Plan
D 2.1: Identify different types of services to be provided by the system documentation repository	+	Plan
D 2.2: Determine groups of access rights to the services	+	Plan
D 3: Subject system documentation repository to SCM	+	Impl
D 4: Establish documentation standards	+	Plan
D 4.1: Define organizational policies/rules/guidelines for developing documentation standards	P	Plan
D 4.2: Share documentation standards with the maintenance team during handover	+(i)	Impl
D 4.3: Develop templates for documentation according to the defined policies/rules/guidelines	+	Plan
D 4.4: Create rules for updating the system documentation repository	+	Plan
D 4.5: Create mechanisms for controlling the status of the system documentation repository	P	Plan
D 5: Transfer the documents from the documentation repository to maintainer	+(i)	Impl
MAINTAINABILITY MANAGEMENT		
MM 1: Assess system maintainability	P	Plan
MM 1.1: Define system maintainability attributes	P	Plan
MM 1.2: Define rules and guidelines for adhering to the system maintainability	P	Plan
MM 1.3: Identify milestones for assessing system maintainability	P	Plan
MM 1.4: Assess system maintainability using the system maintainability attributes	P	Impl
MM 1.5: Assess procedures for managing and controlling system maintainability	P	Plan
MM 2: Assess data maintainability	--	Plan
MM 2.1: Define data maintainability attributes	--	Plan
MM 2.2: Define rules and guidelines for adhering to the data maintainability	--	Plan
MM 2.3: Identify milestones for assessing data maintainability	--	Plan
MM 2.4: Assess data maintainability using the data maintainability attributes	--	Impl
MM 2.5: Assess procedures for managing and controlling data maintainability	--	Plan
SOFTWARE SYSTEM TRANSFER		
ST1: Monitor status of software components	+	Impl
ST1.1: Identify stable software components ready to be used in the system to be handed over	+	Impl
ST1.2: Identify software components under testing stage	+	Impl
ST1.3: Identify software components under development stage	+	Impl
ST2: Make decision on the components to be handed over	+	Impl
ST3: Manage modification Requests	+	Impl
ST3.1: Create a template for managing information about modification requests and their management	+	Impl
ST3.2: Place modification requests in a modification request repository	+	Impl
ST3.3: Use modification requests to revise the handover decision	+	Impl
ST4: Transfer software system	+(i)	Impl
ST4.1: Transfer the agreed upon software components	+(i)	Impl
ST4.2: Transfer the replica of the operational data	+(i)	Impl
ST4.3: Transfer modification requests	+(i)	Impl
ST4.4: Monitor the system after handover	+(i)	Clos
ST4.5: Singoff the handover closuer	+(i)	Clos