Applying *EM³: Handover Framework* in a Project Parking Context

Ahmad Salman Khan [1], **Mira Kajko-Mattsson** [2]
[1]Department of Computer Science & IT, University of Lahore, Lahore, Pakistan
[2]School of ICT, KTH Royal Institute of Technology, Stockholm, Sweden

Abstract - *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 EM³: Handover Framework aiding organizations in constructing their own handover process models. In this paper, we evaluate it 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 setting.*

Keywords: self-to-self software transfer; participatory observation.

1 Introduction

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 system from developer to maintainer 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, software handover is still an under-researched and neglected domain. The published handover models are either too old or they are defined on a very general level [1] [2] [3] [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, 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 contexts, that is, in the project parking context.

In this paper, we report on the results of conducting a handover process at *E-Identity* using our recently developed handover process model – *EM³: Handover Framework. EM³* 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 setting. 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].

Parking implies that the project gets deactivated for some known or unknown period of time, teams working on the project get dissolved, and probably with time, the project will get reactivated (resumed), however, with new team members. Parking is also a type of self-to-self handover. The company (the first self) transfers a partially developed system to itself (the second self).

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 *EM³: Handover Framework*. Section 5 reports on results of the framework's implementation within the company studied and Section 6 rounds up the paper.

2 Company Description

E-Identity is a Swedish company based in southern Sweden. It develops a product for digital identity authentication. It has 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. To be able to continue with the project, the company had to park it.

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 *API 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. 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 the teams. Out of the ten

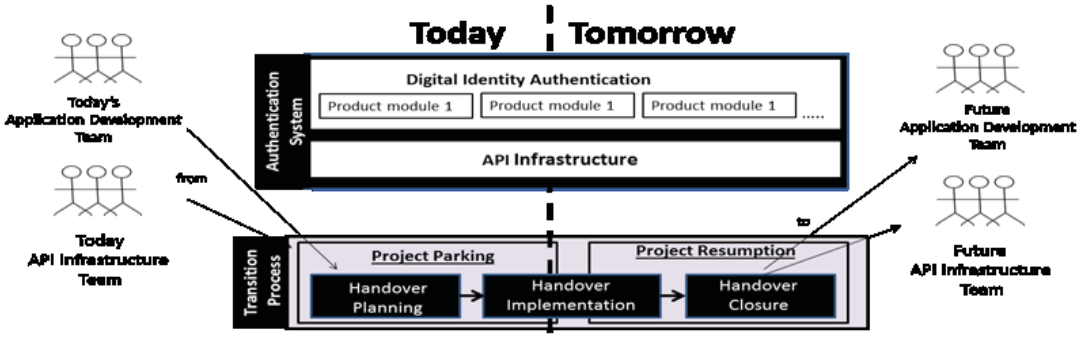

Figure 1. The handover context at E-Identity

people who stayed, four people were involved in the handover process. These were the following (1) *development team lead* responsible for documenting the system knowledge and generating a stable infrastructure development API release, (2) *project manager* responsible for managing the handover project, (3) *product owner* responsible for the product to be handed over, and finally, (4) *researcher* responsible for monitoring and 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* comprises the *handover closure* activities and the rest of the *handover implementation* activities.

3 Research Process

We followed the participatory research 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 the process. We also provided support to the company while implementing *EM*^{3}. 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, documentation study, and informal interviews. In this way, we could identify similarities and discrepancies between the *EM*^{3} practices and the handover process studied.

The *project parking* phase took three weeks to perform. During this time, we conducted four major steps that were 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.

In the *Acting in the Field* phase, we tried to act just as the company's "*local*" member with some minor exceptions [7]. 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.

The third phase, the *Recording Observations* phase, ran in parallel with the *Acting in the Field* phase. While doing our work, we matched it against *EM*^{3}, compared the framework's activities with the company's handover activities and evaluated their applicability. Wherever it was relevant, we suggested improvements. This helped the company to cover the gaps in their handover process and helped us gain feedback for improving our model.

Some *EM*^{3} activities could not be implemented in the process studied. Being such a case, we first asked whether the company conducted them in other handover contexts and inquired about their usefulness.

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 3, 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

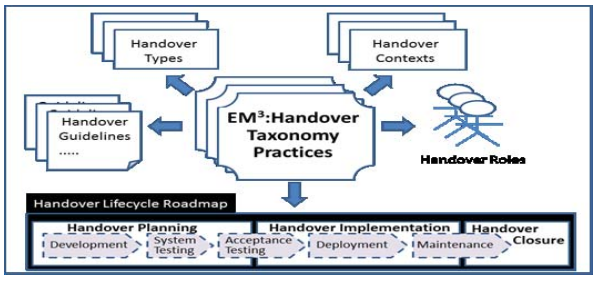

Figure 3. EM^3: Handover Framework

five parts such as (1) *Handover Types* designating 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 in the handover endeavors.

4.1 EM^3: Handover Taxonomy

EM^3: Handover Taxonomy comprises eight practices important for implementing software system handover. They constitute an improved version of the initial taxonomy of handover activities [11]. 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 process strongly depends on it. As shown in Table 1, the practice contains activities starting from planning a handover process, to managing it, to finally, evaluating it postmortem.

Before starting transition, organizations should identify its type and complexity. Transition might be self-to-self or it might be an external one where transitioners and transitionees are separate organizations. Transition might be of high complexity implying a handover of a large safety critical system among several parties or it might be as simple as a self-to-self handover of a system version.

As a next step, the transition team should create a transition plan and assure that important management plans 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 such as development, maintenance processes, to mention a few. A communication model should be in place for interacting and for transferring knowledge between different parties. Throughout the handover, the handover process should be continuously monitored and, at its end, it should be evaluated postmortem.

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. 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, as shown in Table 1, 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.

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 transitionees will not get enough time for learning them or that they may face compatibility problem.

4.1.3 Version and Configuration Management

The *Version and Configuration Management* (VCM) practice includes the activities required for keeping track of changes made to a software system before, during and after handover. This practice is critical for assuring that the system that has been handed over includes the right components. As shown in Table 1, it deals with placing the system under version and configuration management and 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 system delivery during different testing phases. They constitute platforms for identifying and tracking all the changes made to the system and for making important decisions on 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 trained so that they can work from the first day after handover. As shown in Table 1, the *Training* practice focuses on training planning, creating training material and on providing training. To ensure that the training is effective, the practice designates roles responsible for the training process.

4.1.5 Deployment

Deployment is a critical prerequisite for commencing software operation and maintenance. As shown in Table 1, the *Deployment* practice includes activities starting from defining a release scope and contents to preparing for installation, to

Table 1. EM3: Handover Taxonomy practices

+ stands for observed and performed, +(i) stands for inquired about 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

Activities	Sta.us	Phase
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

Activities	Status	Phase
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

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

Activities	Status	Phase
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

Activities	Status	Phase
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

installing and deploying the system, to finally, closing the deployment and planning for future releases.

4.1.6 Documenation

The *Documentation* 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, (2) activities for subjecting the documentation repository to SCM, and (3) mechanisms for controlling the status of the system repository.

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, then it becomes difficult for the maintenance team to understand, and thereby, difficult to evolve and change. If the data is defective, then the company may encounter the problem of a data loss.

Both maintainability types must be assessed before system handover. As shown in Table 1, one 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, one should assess their procedures for managing and controlling data and system maintainability.

5 Status

In this section, we present the results of implementing *EM*$^{3:}$ *Handover Taxonomy* activities at *E-Identity*. Due to space restrictions, we cannot report on the implementation of all of them. We only report on the most important activities. For more information, interested readers are welcome to study [8]. Finally, while participating in the handover process, we observed that not all the *EM*^{3} activities were implementable in the project parking context. To evaluate them, we inquired about their applicability and usefulness in other handover contexts within *E-Identity*. To distinguish them from the observed ones in Table 1, we mark them with +(i) standing for "inquired about and performed".

5.1 Management and Administration

E-Identity has implemented almost all the activities listed in the *Management and Administration* practice. As shown in Table 1, we could observe that all except for two activities were implemented. 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 yet been finalized. 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.

E-Identity experienced a self-to-self type of handover and, due to the 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 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 *Strategy 3*, the company decided to structure handover into two phases: (1) the *project parking* phase and (2) the *project resumption* phase. At the moment of writing this paper, the *project parking* phase got finalized and the *resumption phase* had not yet started.

According to *Strategy 2*, not the whole transition team could be designated in advance. Right now, they had a team for conducting the *project parking* phase. This team will get dissolved. New team will be created in the *project resumption* phase. According to *Strategy 1*, the future transitionees will be consultants instead of fixed-term employees. This will substantially reduce project restart time and cost.

Regarding *Strategy 4,* concerning the transfer of knowledge between the transitioners and transitionees, the company was aware that the two teams would not be able 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.

5.2 Maintenance Environment

E-Identity has implemented all but one activities listed in the *Maintenance Environment* practice. The activity of granting the transitionee permission to access hardware/software suites was not implemented. 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.

5.3 Version and Configuration Management

The company has implemented all except two *EM*^{3} activities for managing version and configurations. The two activities concerned the identification and tracking of the customizable configuration items. The reason for not

implementing them was that the company had only one customer. They did not experience any customization needs.

Regarding the activities that got implemented, we only had the opportunity to observe the accomplishment of their subset. As indicated in Table 1, we observed the complete accomplishment of the activities concerning the management of versions, configurations and baselines (Activities VC1 and VC 2.1). Due to the specific context of the handover process studied, we did not have however the opportunity to observe the establishment of post-delivery baselines (Activity VC2.2).

The company establishes four baselines: *developer test, system test, acceptance test,* and *deployment baselines*. 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.

5.4 Training

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 roles. 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.

The transitionees will be highly responsible for self-educating themselves by studying the documentation that has been created during the handover process.

5.5 Deployment

All the deployment activities as defined in the *Deployment* practice have been implemented *E-identity*. 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 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 that before handover the development team gave priority to meet the delivery deadlines, and hence, they put less emphasis on documentation quality. The documentation was of low quality before starting project parking. As a result, the company decided to develop the documentation standards to be used 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 identified yet. However, in normal handover cases, the company shares the repository by default due to the fact that the transitioner is the same as the transitionee.

5.7 Maintainability Management

The company has not fully fulfilled the *Maintainability* practice. As indicated in Table 1, it has not defined any procedures for assessing data maintainability. They claim that the reason is 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.

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.

5.8 Software System Transfer

The *Software System Transfer* practice was added to *EM³: Handover Framework* during this study. Hence, its activities mirror the activities that were conducted at *E-Identify*. It is worth mentioning that the company distinguished between three types of system components. There were (1) stable components ready to be used , (2) components under testing, and (3) components under development.

6 Final remarks

In this paper, we have reported on the results of implementing the taxonomy activities inherent in *EM³: 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 project parking context.

A quick scan through Table 1 shows that almost all of the EM³ activities have been implemented at *E-Identify*. 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 EM³'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 *EM³*'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 learn to know the transitionees. Finally, it has learned us the following lessons:

- In all transition contexts, one should designate a transition team including the representatives from the transitioners and transitionees. In the context when the transitionee is not yet known and the transition team only includes the transitioner representatives the only communication channel that is possible is a very detailed documentation of the company's products, processes, and technology.
- 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.
- 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 handover.

Even though *EM³: 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 evolve our framework. More handover contexts need be explored and more studies need be done to evaluate *EM³: Handover Framework*. We believe however, that this study has already provided evidence that *EM³: Handover Framework* is on the right path towards providing a fully-fledged support for creating handover process models.

7 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," Proceeding of Conference on Software Maintenance (CSM), 1993.

[3] T. M. Pigoski och J. Sexton, "Software Transition: A Casestudy," Proceedings of *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. Standardization, "ISO/IEC 14764:2006, Standard for Software Engineering- Software Life Cycle Processes-Maintenance," IEEE, 2006.

[6] T. Vollman. "Transitioning from development to maintenance", Proceedings of Conference on Software Maintenance, 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, ISBN:978-91-7501-739-6, 2013, http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-122270.

[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://it-scm.com/.

[11] A. S. Khan, M. **Kajko-Mattsson.** "Taxonomy of Handover Activities", in Proceedings of the 11th International Conference on Product Focused Software, 2010.