Evaluating A Taxonomy of Handover Activities in One Swedish Company

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

Tommy Tyrberg
AMIS SOFTWARE
Sweden
tommy.tyrberg@norrkoping.mail.telia.com

Abstract—Handing over a software system from development to maintenance is still an under-researched domain. The software community has a hazy insight into its constellation and inherent activities. In this paper, we have evaluated a preliminary version of a taxonomy of handover activities within one Swedish software company. The evaluation is conducted in an in-house handover context only. Despite this, our results provide evidence of its enormous complexity, variability and strong dependency on many other software engineering processes.

Keywords – transition, training, deployment, predelivery maintenance, maintainability, maintenance plan.

I. Introduction

Despite the fact that software development is almost six decades old, the software community still possesses a fuzzy insight into many software engineering processes. One such process is handover, alias transition, a process during which one hands over a system from development to maintenance. Handover process is still an under-researched domain. To the best knowledge of the authors of this paper, there are only a few publications dealing with the subject [1, 5, 7, 8, 9, 10]. These are at least two decades old. Hence, they do not reflect current transition needs. As a result, software organizations do not have any guidelines aiding them in handing over their software systems from development to maintenance.

In this paper, we have evaluated a taxonomy of handover activities. The taxonomy in based on our work in [4]. The evaluation is focused on an in-house handover process that may or may not involve COTS suppliers. It is the result of a joint effort between the industry and academia (KTH and a company called AMIS Software). Our goal is to evaluate basic handover activities and provide insight into their industrial status.

The remainder of this paper is as follows. Section II presents our research method used during this study. Section III describes the taxonomy whereas Section IV reports on its status. Finally, Section V makes conclusions and suggestions for future work.

II. Method

Our work consisted of three primary steps. These are (1) literature study (2) handover taxonomy design, and (3) taxonomy evaluation.

In the literature study step, we searched for publications related to handover. The sources we searched them in were digital libraries such as IEEE Xplore, ACM, Springer, Wiley & Sons, and the like, using keywords handover, transition, transfer out, transfer in, rollout and deployment. Unfortunately, we could manage to find only a handful of publications. These are [1, 5, 7, 8, 9, 10]. Hence, our work was entirely based on them.

During the taxonomy design step, we extracted handover activities from the available literature and clustered them into process components that we believed were coherent and logically consistent. These components constitute a preliminary version of a taxonomy of handover activities, which we have published in [4].

In the next step, we evaluated the taxonomy with the Senior System Engineer at AMIS Software, the third author of this paper. For this purpose, we developed a semi-structured questionnaire consisting of the following questions: (1) Is this activity performed? (2) Who exactly performs it? (3) When is it performed? (4) How is it performed? and (5) Is there any activity that we have missed? During interviews, these questions were strongly expanded with many additional “on the fly” questions that had led to a better exploration of the handover field. Their choice was strongly dependent on the activity discussed at hand.

When evaluating the taxonomy, we focused on only an in-house transition in a context in which software systems were either continued to be maintained by the developers or handed over to a separate in-house maintenance team.

Due to the fact that the COTS suppliers may be involved in the process, we also studied how they were engaged in the process. However, we did not focus our evaluation on how the COTS suppliers and other third party suppliers were involved. This is because the third author of this paper has limited experience in cross-organizational handover processes.

The evaluation of the taxonomy within AMIS Software is one of the initial steps in our construction of a software handover process model. Due to its complexity, we had to focus on exploring its activities within only one company before studying it in a broader industrial context. Hence, the results presented herein are highly context-dependent and strongly related to what happens within the company involved in this study. Therefore, they cannot be generalized. More similar studies need to be made to evaluate the credibility of our taxonomy in a wider industrial setting. Despite this, we strongly believe that our work has contributed to taking a tiny little step forward towards building theory about software handover process.
III. TRANSITION PROCESS

Our taxonomy consists of seven components. These are presented in Table 1. They are Management and Administration, Environment Version and Configuration Management, Deployment, Training, Documentation, and Maintainability.

A. Management and Administration

The Management and Administration component includes activities required for managing and administrating the transition process [10]. Since a maintenance team is responsible for post deployment support, it is important to identify it as early as possible (Activity MA-1). Identifying a maintenance team however is not enough. One must also define a group responsible for the transition (Activity MA-2). According to [8], a separate transition group should be formed within a maintenance organization.

<table>
<thead>
<tr>
<th>Activity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MA-1.1 Identify maintenance organization. (D, D, D/M, COTS) (DV, ST, AT, DP)</td>
<td></td>
</tr>
<tr>
<td>MA-1.2 Establish a transition team. (D, D, D/M, COTS) (DV, ST, AT, DP)</td>
<td></td>
</tr>
<tr>
<td>MA-1.3 Create a transition plan. (D, D, D/M, COTS) (ST)</td>
<td></td>
</tr>
<tr>
<td>MA-1.4 Define transition budget. (D, D, D/M, COTS) (ST)</td>
<td></td>
</tr>
<tr>
<td>MA-2.1 Define manpower requirements. (D, D, D/M, COTS) (ST)</td>
<td></td>
</tr>
<tr>
<td>MA-2.2 Define maintenance facility requirements. (D, D, D/M, COTS) (ST)</td>
<td></td>
</tr>
<tr>
<td>MA-2.3 Define other transition elements (not covered in this study).</td>
<td></td>
</tr>
<tr>
<td>MA-2.4 Develop management plans. (D, D, D/M, COTS) (ST)</td>
<td></td>
</tr>
</tbody>
</table>

A transition group should create a transition plan (Activity MA-3) identifying the activities to be performed during transition, their timing and order, and roles involved. It is impossible to list all its constituent activities. Hence, we only focus on the most important ones. These are determination of a transition budget (Activity MA-3.1), time frames for the transition (Activity MA-3.2), standard operating rules and procedures for managing and coordinating the transition process (Activity MA-3.3), and resources required for the transition process itself (Activity MA-3.4).

As [8] states, some parts in the transition plan overlap with the maintenance plan. These concern determination of maintenance resources including both manpower and facility requirements (Activity MA-3.4). The manpower requirements define the size of the maintenance team, the roles involved in it and the order of recruiting the roles (Activity MA-3.4.1).
is recommended to hire software configuration management roles first, testers second, and programmers last [8]. The facility requirements (Activity MA-3.4.2) include hardware and software equipments.

Finally, a number of management plans should be developed well before the transition starts (Activity MA-4). These plans include Software Configuration Management Plan (SCMP), Software Quality Program Plan (SQPP), Software Test Plan (STP), and Training Program Plan (TPP).

B. Maintenance Environment

The Maintenance Environment component includes activities required for setting up the maintenance environment covering hardware and software suites and maintenance support suite. It then includes activities for installing the system and its data and for transferring the modification requests [10].

Hardware and software suites encompass a collection of hardware and software systems providing a platform for installing the transitioned system. Maintenance organizations starting from scratch may have to determine their needs for hardware and software suites, acquire and install them (Activities ME-1 and ME-2). For maintenance organizations that continue with maintaining a certain pool of systems for a particular developer, it may be sufficient that they assess current hardware and software suites and attend to their deficiencies, if any (Activities ME-3 and ME-4). The same applies to maintenance support suite that must be determined or assessed (Activities ME-5 and ME-6). The suite stands for a pool of software tools required for performing maintenance. If it is insufficient or defective, then it must be repaired or supplemented with new tools. Finally, it must be installed (Activity ME-7).

It is only after maintenance support tools have been installed, that one may start installing baseline software and data (Activities ME-8 and ME-9). Baseline software is the version of the transitioned software system. Data, on the other hand, is baseline and operational data.

Many modification requests get reported during development. However, due to various reasons, not all of them get resolved by then. Some of them will have to be resolved in the post deployment phase by the maintainer instead. For this reason, they must be handed over to the maintenance group (Activity ME-10) which places them in a Modification Request Repository and attends to them (Activities ME-11).

The modification requests, their number, severity and criticality aid the maintainer in understanding the system and its condition. They create a basis for planning his work and for starting a relation between maintainer and acquirer.

C. Version and Configuration Management

The Version and Configuration Management component includes activities needed for tracking and controlling changes in the software system. Its prerequisite activity is the establishment of a configuration baseline corresponding to a frozen system version (Activity VC-1) [10]. Having the baseline in place, the maintainer takes it over and puts it under configuration management, preferably, as much automated as possible (Activity VC-2). By a software system, we mean all system parts including code and all levels of system documentation.

During software evolution, many organizations create multiple variants of the same configuration item. To track changes in them requires version management and control. Therefore, the maintainer should place the modified software under version control (Activity VC-3).

D. Deployment

The Deployment component contains three activities required for installing the software system at the acquirer’s site. First, one develops installation procedures such as procedures for generating software builds with the associated documentation and training materials (Activity DP-1). Second, one installs the system (Activity DP-2). Finally, one plans future releases, by planning future updates and by determining the structure of the software packages, forms of deploying them and of release notes (Activity DP-3) [8, 10]

E. Training

Maintainers cannot perform their duties in a satisfactory way and end-users on the acquirer’s side cannot properly use the system if they do not possess system knowledge. Hence, they should be provided proper training well before taking over their maintenance and end-user responsibilities.

The Training component contains activities for providing training to the stakeholders of the system. The stakeholders involved are mainly acquirer and maintainer. The component includes training on the system and its architecture (Activity T-1) [10]. Since software systems evolve with time, it also includes the development and update of training material covering both system operation and maintenance (Activities T-2 and T-3) [7]. Maintainer should also be provided with training on new technology to be used in operation and maintenance of the system (Activities T-4). Finally, the maintainer should provide on-site support in critical cases (Activity T-5) [8]. On-site support fulfills both training and support purposes. According to [10], it is the maintenance team that should provide on-site support.

F. Documentation

The Documentation component includes activities for enabling the transition of documentation. Its activities mainly deal with basic procedures for creating a documentation repository and for establishing the services to be provided by it (Activity D-1, D-2) [5]. Since software systems evolve with time, the documentation should also remain up-to-date. Hence, the repository should be subjected to the same level of configuration management as code (Activity D-3) [10]. Finally, standards for developing documentation should be defined so that all the stakeholders have a common understanding of the structure and terminology used when documenting their systems (Activity D-4).

G. Maintainability

System maintainability is a quality attribute that has to be considered throughout the whole software lifecycle. Its
management is the main ingredient in the predelivery maintenance phase during which maintainers continuously monitor and control maintainability [8]. In the transition phase, maintainers should make final assessment of maintainability to assure that it conforms to the maintainability requirements. The same applies to data maintainability (Activity M-1 and M-2).

IV. STATUS

AMIS Software is a Swedish organization providing services and solutions ranging from military defense to civil products. The company is expanding its business operations globally with its headquarters in Sweden.

At AMIS Software, development team is responsible for the system design and development. After the system becomes operational, the responsibility for evolving and maintaining it is context-dependent.

In most of the cases, the organization practices “cradle-to-grave” approach implying that maintenance stays with development. This means that enhancements of systems with new features and corrective maintenance are continued by the same groups. Here, no transfer of responsibilities occurs between development and maintenance groups. However, some form of responsibility transfer from self-to-self still applies. This corresponds to Context 1 as identified in [3] and illustrated in Figure 1.

In the remaining cases, system enhancements with new features still stay with the development group, however, corrective maintenance responsibilities are now taken over by a separate in-house maintenance group. This corresponds to Context 2 as identified in [3] and illustrated in Figure 1, in which the responsibilities for correcting major versions and putting the corrections into minor revisions are transferred to a separate maintenance group.

In this section, we match the status to our taxonomy of handover activities on the two-above mentioned contexts. Even if the in-house handover process may somewhat involve COTS suppliers, our status report is still focused on the in-house handover.

A. Management and Administration

AMIS Software implements all the activities defined in the Management and Administration component. Regarding the maintenance and transition teams, their constellations vary depending on the system size, the complexity of the transition process and the number of actors involved.

The constellation of a maintainer group depends on (1) whether the whole system is developed in-house, (2) whether COTS-components are used during its development, or (3) whether the development is outsourced to a third party. In the first case, either developers continue with evolution and maintenance or a separate maintenance group manages only corrective tasks. In the second and third cases, AMIS Software maintains their own system parts and the COTS and third party suppliers maintain their own ones.

All the parties involved in maintenance are identified very early. COTS suppliers and third-party maintainers are designated by business management very early in the development cycle. Regarding the in-house maintenance teams, they are determined in the initial project phases by project management.

Early composition of a maintenance group is not fixed. It may be changed later on. Usually, the reasons are changes to the understanding of the complexity of the maintenance required. For instance, AMIS Software may have initially planned that the developers will continue with maintenance. At the end of a project, however, the development group may realize that combining evolutionary development and corrective maintenance would imply too much burden. As a result, they may decide to create a separate in-house maintenance group or outsource it to a third party.

At AMIS Software, a transition group is established only for large projects. Just as the constellation of maintenance group varies so does the constellation of a transition group. At a minimum, it consists of a developer and acquirer. This is the case when the development group continues to evolve and maintain the system. At a maximum, it includes developer, acquirer, maintainer and external consultants. The maintainer in turn may include COTS, third-party and in-house maintainers. Regarding the consultants, they are usually specialists whose role is to relieve transition group members from various complex and time consuming tasks.

The designation of a transition group is made by project management. The group is identified when one has a good overview of the system and its size and complexity. This usually takes place right before system testing after one has integrated the whole system. Based on the status of the integrated system, one acquires understanding of the requirements for the transition group. If one suspects that the transition is going to be complex, one starts planning and creating the group as early as possible.

Planning of the transition group is critical. Usually, the individuals involved in it are engaged in many other projects. Hence, identifying the transition group and scheduling the availability of its members is critical for succeeding with the transition process. In case of delays, the whole transition process may be substantially delayed. At its worst, the whole delivery may be jeopardized.

AMIS Software creates transition plans. The plans are created by project management during system testing. They (1) determine budget specifying who pays to who and for what, (2) schedule of the transition process, (3) establishment of transition procedures, and (4) the determination of transition and maintenance resource requirements.
AMIS Software reviews maintenance manpower and facility requirements. They do it in an early project phase. The resources however are only estimated. As the project proceeds, they are continuously adjusted.

AMIS Software creates many management plans. Among the most important ones are (1) Software Test Plan (STP) and (2) Training Program Plan (TPP) and (3) Deployment Plan. The timing of and the roles involved varies depending on the plan. The main role that is always involved is developer who, depending on the plan, is then supported by acquirer, maintainer, COTS and third-party suppliers.

B. Maintenance Environment

All the activities as defined in the Maintenance Environment component are conducted at AMIS Software. They all take place in the time interval starting at the end of the system testing phase and ending in the early deployment phase.

The first four activities are trivial at AMIS Software. They concern acquisition of new or assessment of the existing hardware and software suites. They are conducted by both developers and maintainers. At AMIS Software, the maintainers’ hardware and software technological suite is the same as of developers’. Hence, the assessment activities at AMIS Software correspond to a quick evaluation of the suite. Usually, one does not discover any deficiencies. If there are any, then they are often of a trivial character.

Assessment and installation of maintenance support software is considered to be a trivial activity as well. Being involved in maintenance for years, AMIS Software has already many maintenance support tools in place. Sometimes, however, some of them may have to be replaced or complemented with new tools that may facilitate maintenance work.

The installation of baseline software is conducted by developer and maintainer, whereas the roles involved in the installation of data vary depending on the complexity of the system being installed. If simple, then only maintainer and acquirer are involved. If complex, then one will often discover various mismatches between the data structure and format in the baseline data and the software system managing the data. In such a case, developer will have to manage various fixes to the software system.

Regarding the in-house systems, both developers and maintainers have access to Modification Request Repositories containing all modification requests. At AMIS Software, maintainers assist developers in system and acceptance testing during which they generate requests.

No significant transfer of modification requests takes place for the in-house systems. As soon as the system is delivered, the predelivery modification requests and predelivery problem management process become postdelivery following a more complex format and process design [6].

C. Version and Configuration Management

Establishing SCM baseline implies freezing a certain system version. At AMIS Software, three SCM baselines are established: system test baseline, acceptance test baseline, and operational baseline. All these baselines are defined at different points in time and they are all available to the maintenance group. The reason is the fact that maintainers start maintaining the system already during system testing.

The constellation of the roles involved in the maintenance group varies depending on who is going to maintain what. They range from only developers, to developers and maintainers, to developers and maintainers supported with COTS and third-party maintainers.

Regarding the system test baseline, it is established right before one starts with system tests. The system undergoes many changes at this stage which are made by both developers and maintainers. The changes are made to the software system but also to routines and procedures related to the use of the system, to the data to be used, or to workarounds, in cases when other changes are impracticable.

The acceptance test baseline is established right before acceptance testing starts. It is installed on the premises of the acquirer and the developers and maintainers attend to all the problems encountered. All problems detected during this phase should be solved as much as possible and the solutions should be re-tested.

Finally, the operational baseline is determined in the deployment phase. It is distributed to the company’s customers. At this moment, post-deployment maintenance starts. In its early phase, the changes made to the software system are still performed by maintainers and developers to be gradually taken over by only maintainers.

The choice of the roles involved in establishing baselines depends on the size and complexity of the system. They may range from:

- Only developers in cases when developers both develop and maintain the product.
- Developers and maintainers in cases when there is a separate in-house maintenance group.
- Project management encompassing product manager, developers and maintainers in cases of larger systems.
- Installation group consisting of project management and other internal and external parties in case of very complex systems. The internal parties are hardware installers and the external ones are COTS or third-party maintainers.

The use of configuration and version management strongly varies at AMIS Software. Very complex embedded systems may be put under very advanced in-house SCM tool managing complex dependencies between hardware and software components. Simple systems may be configured with only Excel.

Some systems are neither put under configuration and version management control. Most of them are very simple. However, it is worth mentioning that one very complex software system has not been put under configuration and version management control for 30 years. Its entire configuration has been managed manually by meticulously updating all the changes to both its code and documentation. The reason why this has been possible is the fact that the product keeps all its components in only one version. It does not need to re-construct its former versions.
Placement of the system under configuration and version control is conducted by technicians under the supervision of the same roles that are involved in establishing the SCM baselines. The same applies to the point in time when the systems are put under the SCM control.

D. Deployment

AMIS Software develops installation procedures for all their systems. They start developing them in the initial phases and finalize them at the end of a project. The roles involved are developers to be later supported by a transition group. An example of installation procedures are procedures for rolling back certain features.

AMIS Software makes three types of installations for three different baselines. The roles involved are the same as those involved in establishing baselines and putting them under configuration and version control.

Planning of future releases is conducted by the product management and maintenance group. Its complexity depends on whether it is an in-house product or whether third-party components are used. In the latter case, the planning is made very cautiously bearing in mind the fact that system update with a new COTS versions can lead to comprehensive changes and costs, often comparable to the development of a totally new system. This is particularly true if the third-party components have been “tweaked” in some way implying that a similar “tweaking” must be applied to a new release, often a far from trivial task.

According to the third author of this paper, organizations should have a clear policy defining how to introduce new releases. Such a policy is often missing which results in problems such as staying with old versions of COTS software until they become difficult or impossible to support and integrate with other software.

Some releases may be run on one node whereas others may be run on several nodes. Hence, the distribution aspect is very important. It guides the way one defines forms of deploying the system. Simple non-distributed systems may be distributed via CDs or Internet whereas complex distributed systems may require careful planning of human resources and time scheduling. For instance, they may require that one will have to send individuals to various distributed nodes who will simultaneously deploy the system on those nodes. Both the determination of the distribution and deployment is made by the development maintenance group late in the acceptance testing phase.

Even management of release notes varies depending on the complexity and size of the new releases. In simple cases, a release note corresponds to a structured email to be sent during deployment. In complex cases, releasing major changes implies creation of separate projects corresponding to a training process to be presented in Section IV E. Acquirer usually has customer centers on their sites providing all types of daily support to their software system users. In some initial operational phases, AMIS Software may send over a support technician to the acquirer’s site. The organization may also provide a permanent on-site Service Engineer, a role coming from the supplier who provides permanent on-site support.

E. Training

All types of training as listed in our taxonomy are performed at AMIS Software. However, they vary in complexity, training needs, timing and the parties involved. In addition, AMIS Software conducts training on technology required for operating and maintaining the system. All training consists of two main stages in which one first prepares training material and then provides training.

A preparation for training on software systems, their structure and operation is one of the most complex, painstaking and time-consuming processes. Hence, preparations may start during development as soon as one has a fair overview of the system. They imply creation of an installation infrastructure that resembles the real environment, creation of a database and its population with data relevant for the training at hand, creation of all kinds of routines for enabling feature and data execution and rollback, and creation of relevant exercises and supporting texts.

 Provision of training on software systems, their structure and operation is still a very complex process, however, not as painstaking as its preparation. At AMIS Software, it consists of four phases during which different groups are trainers and trainees. The phases are (1) Training of software professionals, (2) Training of super users, (3) Training of managers, and (4) Refresher and new user training.

The training in the first phase is conducted by COTS suppliers or by in-house developers. The training in the remaining phases may be conducted by different groups: in-house developers, support personnel, operations personnel, or a special educational group dedicated to only training.

Training of software professionals may be provided by either COTS suppliers or in-house developers. Regarding training on COTS components, it may start as soon as AMIS Software has decided which components to buy. Regarding training on an in-house system, it starts as soon as the system is executable. Whether in-house or not, the professionals to be trained at this stage are the ones who are going to be involved in developing, testing, support, on-site support and maintenance. The goal is to make sure that they are prepared for conducting their future work.

Due to the system complexity and for pedagogical reasons, system training of professionals may be conducted in several sub-stages, where each sub-stage is dedicated to a specific part such as general orientation, OS and DBMS and system-specific matters.

In the vein of the existing handover models [4], we have listed white-box testing as a training activity in our taxonomy. Our results, however, show that white-box testing is not normally conducted for the training purposes at AMIS Software. However, when attending to software problems during system and acceptance testing, maintainers conduct various forms of debugging and white-box testing aiding them in understanding both the system and the problem being investigated. Hence, we conclude that debugging and white-box testing may be regarded as important training drivers.

Training of super users is conducted right before the users start using the system, preferably, the day before. This training and its preparations are not as complex as the
training of software professionals. However, it still requires complex and difficult preparations such as creating of a training environment, migration of data to this environment and creation of user adapted training material. Super users play the role of educators within their organizations. Hence, they have to be thoroughly trained for this task.

After having trained the most important roles, one trains the remaining roles. These are mainly managers who may or may not use the system in the future, however, they need to get acquainted with the system, its purpose and main features. Preparing training materials for managers is trivial in comparison to preparing training material for software professionals and super users. Also, provision of the training may be based on simple system demonstrations that do not always require meticulous preparations of the training data.

Finally, AMIS Software provides continuous training to either new hires or employees who need refresher training. Preparations of this type of training takes place after the system gets operational.

Training on support and maintenance processes is not a big issue at AMIS Software. The organization has already the processes in place which are often easily adaptable to the new system context. The support and maintenance processes may however be quite complex in cases when a new system and its needs for support and maintenance strongly diverge from the needs of the already existing systems. In such a case, the support and maintenance process owners adapt their existing processes to the new needs and make sure that the support and maintenance personnel is properly trained. The adaptation of the processes may start as early as during development and training on them may start as early as during system testing.

Development or modification of a system often implies acquisition of new or change of old technology used, such as OS, testing tools, database, supporting tools, compilers, and other things. For this reason, software professionals need to be trained about them. This type of training is either performed by the technology supplier or AMIS SOFTWARE. It is conducted on an as-needed bases whenever one knows that a new technological device will be used within development, support and maintenance.

F. Documentation

At AMIS Software, the establishment of a system documentation repository takes place in the very early development phases. The repository is organization-wide and all relevant roles within the organization have access to it. Due to very high and ambitious requirements for keeping the documentation up to date and consistent, AMIS Software imposes very strict controls; only designated individuals are allowed to create and update it, the normal user can retrieve a copy but cannot update it or put it back into the repository.

The roles responsible for documentation vary with system size and complexity. Usually, developers manage system documentation of simpler systems and a documentation group manages documentation of very complex systems.

All central documentation repositories provide basic services such as creation, update, storage and retrieval of documentation items. Except for the cases mentioned in Section IV. C, almost all repositories implement version and configuration management services. Finally, AMIS Software defines standards for how documentation should be developed. These standards concern document structure, format and similar.

G. Maintainability

As pointed out in [2], AMIS Software does not create any model of maintainability but they have guidelines for developing system architecture and for writing code. It is recommended that these guidelines be used for continuous maintainability assessment during the whole development process. However, the recommendations are not always followed.

Irrespective of whether the maintainability guidelines are followed or not during early development, the system maintainability is assessed during system testing. It is mainly done by the QA team. However, at this late stage, major changes are difficult to accomplish, and these usually have to be deferred to future releases.

Data maintainability applies to both baseline data and operational data. Baseline data are all the basic data that are required for making system work, such as for instance, initial data. Operational data, on the other hand, are the data that provide input for running the system or they constitute output of a system operation, such as for instance, parameterized commands of a particular function.

Management of data maintainability is considered a challenging task. One does not only have to maintain it for all kinds of systems but one also has to consider all kinds of system and environmental contexts and dependencies. Finally, one must determine which stakeholder needs what data and on what data aggregation level. A minor deficiency in the quality of data may have disastrous effect on system testing, operation and maintenance. In the worst case, it may jeopardize the system delivery.

Data maintainability is mainly managed by the acquirer. AMIS Software supports the acquirer in various data management tasks. Maintaining data maintainability is a very costly activity. It can vary by an order of magnitude between similar systems depending on the degree of maintainability.

V. FINAL REMARKS AND SUGGESTIONS FOR FUTURE WORK

In this paper, we have created a first version of taxonomy of handover activities and evaluated it against an industrial handover process at AMIS Software. During our study, we have identified some discrepancies in our taxonomy and the industrial process studied. These are:

- Current predelivery maintenance and handover process models advocate that the maintainer be designated and that the maintenance needs be identified in the initial project phases. Our study shows that the maintainer designation and maintenance needs may change with time due to the changed understanding of the maintenance complexity. Hence, it may not always be
appropriate to designate the maintainer at the beginning of a project.

- Current transition models suggest that maintainer be responsible for the maintenance process. In our study, however, we have observed that developer, acquirer, COTS suppliers and third-party suppliers are involved as well.

- According to [8], a separate transition group should be formed within a maintenance organization. Our study, however, indicates that its constellation is dependent on the transition complexity, involving only developer and maintainer at a minimum and developer, acquirer, maintainer and external consultants at a maximum.

- Current models suggest establishment and deployment of one SCM baseline. Our study has shown that at least three SCM baselines need to be identified and deployed in the transition process (system test, acceptance test and operational baselines).

- The transition models suggest training of all parties concerned. They have however not clarified its complexity. Our study shows that sometimes training may be as complex as the transition process itself.

- Current transition models suggest create/update training material on system, its structure and operation. However, our study identifies a need to set up a training environment for system operation training.

- The transition models classify white-box testing as a training activity. Our study has shown that maintainer does not treat it as a training driver; rather as support while resolving predelivery software problems [1].

- Current transition and predelivery maintenance models only mention system maintainability. Our study reveals that data maintainability is as critical as system maintainability, sometimes even more critical.

- Finally, we would like raise the issue of the use of supporting version and management tools versus strict manual control of system configurations and versions. Our study has shown that it is possible to achieve full manual control over the system and its traceability if one imposes strict documentation process discipline.

   Even if we have studied only one in-house transition process, we have been able to slightly touch on more complex transition processes distributed among several parties. Based on this study and on our graphical illustration of the placement of its activities on the phases in Table 1, we may already conclude that the transition is a process stretching from the initial development phases to the initial post-deployment phases. The wide time span of its activities indicates strong process variability and parallelism with many other processes.

   We also conclude that transition process strongly varies depending on the system to be handed over, its criticality and complexity and the number of parties involved. Even less complex processes require good discipline. If performed in an ad hoc manner, their poor performance may jeopardize the delivery, and thereby, business opportunity and survival.

   Our taxonomy, as it is now, can already be found useful for the companies practicing transition. It can also provide feedback for the academics when learning about the handover, when creating handover process models or when developing and improving other processes whose performance and maturity may strongly depend on the handover phase.

   Our taxonomy is still in its infancy. It needs to be expanded with more activities and complemented with various process scenarios in various industrial contexts. It is only then we will be able to develop a generic handover process model. When striving towards that end, however, we feel that it may be difficult to develop a generic process model due to its strong process variability and time span. The process model will have to implement flexibility. Due to its business criticality, it will also have to be strongly risk-driven and well integrated with both the predelivery and postdelivery maintenance processes. It is only then one may assure that the organization may succeed with their handover processes.

REFERENCES


