A Framework for Software System Handover

Ahmad Salman Khan

Doctoral Dissertation in
Computer and Systems Sciences
School of Information and Communication Technology
KTH - Royal Institute of Technology, Stockholm, Sweden, 2013
Academic dissertation for the Degree of Doctor of Philosophy in Computer and Systems Sciences at KTH - Royal Institute of Technology to be publicly defended on June 4, 2013 at Sal D, Forum 100, Isafjordsgatan 39, Kista.

© Ahmad Salman Khan, 2013

Tryck: Universitetsservice US-AB
Acknowledgement

I would like to pay my gratitude to my supervisor Associate Professor Mira Kajko-Mattsson for her guidance throughout my studies. Her utmost dedication for work was always inspirational to me. I am specially obliged to her for preparing me as a researcher and coaching me in the art of scientific writing. I will remain indebted to her with every sentence that I will write during my professional carrier.

I am thankful to my research group fellows Natalja Nikitina and Gudrun Jeppesen for their philosophical discussions and for providing valuable feedback on my research. My special thanks go to my colleagues and friends Irfan Younis, Khurram Shahzad, Salman Niazi, Shahzad Saleem, Imran Mahmood, Mudassar Aslam, Nima Dokooehaki, Shahab, Adeel Tajammul, Adil Razzaq, Siraj Rathore, Syed Jafri, Omer Malik, Ali Shami, Jamshaid Sarwar, Nasim, Nowshad Painda Mand, Shahid Raza, Muhammad al Badawi, Kathrin Dannmann, Katharina Rasch, Fernando Herrera, Francesco Robino and many others. It was always refreshing to interact with you.

I communicated with many professional from the IT industry during my studies. I am especially thankful to Mr. Ahsan Saleem, Mr. Imran Shahid, Mr. Tommy Tyrberg, Mr. Furqan Khan, Mr. Eric and Mr. Asif Fayyaz for their cooperation and useful input in my research.

It would not be justified if I do not mention the staff at Software and Computer Systems unit at School of Information and Communication Technology at Royal Institute of Technology. I am thankful to Ms Sandra Gustavsson and Mrs. Marianne Hellmin for being available all the time for solving administrative issues at very short notices.

I am grateful to Higher Education Commission of Pakistan for sponsoring my studies. I am also thankful to Mr. Thomas Sjöland, the head of the Software and Computer Systems unit for providing the best office facilities and comforting environment that was essential for good productivity.

My special thanks go to my parents, for their affection and support throughout my life, especially my mother for being the first teacher in my life. Last but not least, I would like to deeply acknowledge my wife Noor Fatima, for her supportive, caring and understanding nature and for showing patience with me especially during the last year of my PhD studies.
Abstract

A transfer of a software system from a party responsible for its development to a party responsible for its maintenance is referred to as software system handover or transition. It does not only include a transfer of a software system or system part, but also a transfer of all the knowledge, experience and responsibilities that are required for managing the system during a software system lifecycle. It is a complex process overarching many different processes such as development, testing, version and configuration management, deployment, and maintenance, just to mention a few.

Handover is a critical process in the software lifecycle and it is important that it is performed as smoothly as possible. It can become very expensive and protracted if the organizations are not well prepared for performing it. The productivity of the groups taking over the system and the quality of their outputs may be strongly dependent on how handover is performed. Overall, failing to transfer software systems may lead to loss of productivity, loss of maintainer credibility, loss of system and maintenance process quality, and sometimes, even loss of business. Therefore, a good handover process model is imperative and critical for planning for and managing transition and for alleviating many transition problems.

Regrettably, the issue of the software handover is still a strongly under-researched and neglected domain. There still do not exist any up-to-date handover process models that designate important features of a handover process. So far, the research community has not paid much attention to this particular process domain. The published handover process models are either too old or they deal with handover on a very general level.

In this thesis, we suggest a framework for software system handover that provides a basis for creating handover process models. The framework is called EM³: Handover Framework and it is part of Evolution and Maintenance Management Model (EM³). The framework comprises six major parts: taxonomy of handover activities, handover contexts, handover types, handover roles, handover lifecycle roadmap, and handover guidelines. These parts may be combined in different ways for creating handover process models for specific handover contexts.

The research method used was of qualitative character including explorative case studies and participatory studies. It is based on data collected within 61 companies during which the theory about the handover process domain was explored, evolved and evaluated. The primary research tools used were semi-structured interviews and observation.

EM³: Handover Framework guides organizations in implementing their handover processes and in identifying problems and challenges they may face before, during and after the handover. Organizations can use it to conduct their own handover processes and researchers can further evaluate it in other industrial contexts, extend it, and/or use it for suggesting their own handover process models.
# Table of Contents

## Part I

### Chapter 1  Introduction

1. Problem Statement .......................................................... 2
2. Research Goal ................................................................. 2
3. Contribution .............................................................................. 3
4. Bodies involved ............................................................................. 3
5. Target Groups .............................................................................. 4
6. Transition vs. Handover Terminology ............................................. 4
7. Limitations ................................................................................. 4
8. Thesis Outline .......................................................................... 5
9. List of Publications ..................................................................... 6

### Chapter 2  Software System Handover

1. Software Handover ...................................................................... 11
2. Software Maintenance and Transition ........................................... 13
3. Software Maintenance .................................................................. 14
4. Pre-delivery Maintenance ........................................................... 14
5. Software Transition ..................................................................... 15
6. State of Art .................................................................................. 15
7. Transition Models ......................................................................... 18
8. Transition Model 1 ........................................................................ 18
9. Transition Model 2 ........................................................................ 20
10. Transition and IT outsourcing ...................................................... 22
11. Outsourcing Definition ............................................................. 22
12. Main challenges of IT outsourcing implementation ....................... 23
13. Transition Process in IT outsourcing .............................................. 23
14. Discussion .................................................................................. 26

### Chapter 3  Research Process

1. Overview of Research Phases ....................................................... 27
2. Research Phases .......................................................................... 28
3. Choice of PhD topic ...................................................................... 28
4. Initial Study ................................................................................. 29
5. Literature Study, First Round ......................................................... 29
6. Scope Study ............................................................................... 30
7. Creation of Handover Taxonomy and Elicitation of Handover Problems ......................................................... 31
8. Small-Scale Exploration ............................................................... 32
9. Large-Scale Exploration ............................................................... 33
10. Final Study ................................................................................ 34
11. Literature Study, Second Round ................................................... 34
12. Acting on Handover Taxonomy ...................................................... 35
13. Experiences ................................................................................. 36
14. Research Approach ................................................................. 37
15. Quantitative versus Quantitative Research ................................. 37
16. Why qualitative Research Approach? ........................................... 38
17. Types of Studies ......................................................................... 39
18. Sampling .................................................................................... 40
19. Research Tools ......................................................................... 40
20. Validity Threats ......................................................................... 41
3.6.1 Internal validity – Credibility
3.6.2 External validity – Transferability
3.6.3 Reliability – Dependability
3.6.4 Objectivity – Confirmability
3.6.5 Construct validity
3.7 Bodies involved

Chapter 4 Research Contribution
4.1 Initial Study publications
4.2 Small-Scale Exploration publications
4.3 Large-Scale Exploration publications
4.3.1 Large-Scale Exploration publications - Part 1
4.3.2 Large-Scale Exploration publications - Part 2
4.4 Final Study publication

Chapter 5 EM³: Handover Framework
5.1 Handover types
5.2 Handover contexts
5.2.1 Product lifecycle perspective
5.2.2 Process lifecycle perspective
5.2.3 Development lifecycle perspective
5.3 Handover Lifecycle Roadmap
5.4 Handover Roles
5.5 Handover Guidelines
5.6 Handover Practices
5.6.1 Management and Administration
5.6.2 Maintenance Environment
5.6.3 Version and Configuration Management
5.6.4 Training
5.6.5 Deployment
5.6.6 Documentation
5.6.7 Maintainability Management
5.6.8 Software System Transfer

Chapter 6 Epilogue
6.1 Contribution
6.2 Future work

Bibliography

Part II Appendices
Appendix A EM³: Handover Taxonomy versions
Appendix B Questionnaires
Appendix C Templates
Appendix D Bodies involved
Appendix E EM³: Handover Taxonomy Guidelines
Appendix F EM³

Part III Publications
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 2.1.</td>
<td>Transition definitions</td>
<td>13</td>
</tr>
<tr>
<td>Table 2.2.</td>
<td>Pre-delivery and post-delivery responsibilities of maintainer…</td>
<td>16</td>
</tr>
<tr>
<td>Table 2.3.</td>
<td>Pigoski transition model activities</td>
<td>21</td>
</tr>
<tr>
<td>Table 2.4.</td>
<td>Pigoski transition activities</td>
<td>22</td>
</tr>
<tr>
<td>Table 2.5.</td>
<td>Transition activities for offshore transition</td>
<td>25</td>
</tr>
<tr>
<td>Table 5.1.</td>
<td>Management and Administration practice activities</td>
<td>75</td>
</tr>
<tr>
<td>Table 5.2.</td>
<td>Maintenance Environment practice activities</td>
<td>77</td>
</tr>
<tr>
<td>Table 5.3.</td>
<td>Version and Configuration Management practice activities</td>
<td>79</td>
</tr>
<tr>
<td>Table 5.4.</td>
<td>Training practice activities</td>
<td>81</td>
</tr>
<tr>
<td>Table 5.5.</td>
<td>Deployment practice activities</td>
<td>82</td>
</tr>
<tr>
<td>Table 5.6.</td>
<td>Documentation practice activities</td>
<td>84</td>
</tr>
<tr>
<td>Table 5.7.</td>
<td>Maintainability Management practice activities</td>
<td>85</td>
</tr>
<tr>
<td>Table 5.8.</td>
<td>Software System Transfer</td>
<td>87</td>
</tr>
<tr>
<td>Table 6.1.</td>
<td>Research contribution</td>
<td>91</td>
</tr>
</tbody>
</table>
List of Figures

Figure 2.1. Handover process in the system lifecycle ................................. 11
Figure 2.2. ISO/IEC 14764 Maintenance Process ..................................... 13
Figure 2.3. Transition plan topics ISO/IEC 14764 ................................. 17
Figure 2.4. ISO/IEC 15288 System Lifecycle Processes ............................ 18
Figure 2.5. Procedures for SSA function.................................................... 20
Figure 2.6. Pigoski’s Transition model ...................................................... 22
Figure 2.7. Zhu Outsourcing stages.......................................................... 24
Figure 2.8. Transition plan topics for outsourcing ..................................... 24
Figure 2.9. Lever outsourcing stages ....................................................... 25
Figure 3.1. Research process phases......................................................... 28
Figure 3.2. Taxonomy of handover activities............................................. 31
Figure 3.3. Handover Taxonomy Exploration part 1 and part 2...................... 33
Figure 4.1. Software lifecycle roadmaps and contexts................................. 49
Figure 5.1. EM³: Handover Framework................................................... 67
Figure 5.2. Software system handover types............................................. 68
Figure 5.3. Handover contexts in the product lifecycle roadmap................. 69
Figure 5.4. Handover contexts in the process lifecycle roadmap................. 70
Figure 5.5. Handover contexts in the development lifecycle roadmap......... 71
Figure 5.6. EM³: Handover Framework activity spaces.............................. 72
Figure 5.7. EM³: Handover Framework roles.......................................... 73
Figure 5.8. EM³: Handover Taxonomy ....................................................... 74
Part I
1. Introduction

Transfer of responsibilities between two parties is performed in many different areas of our daily life. Examples and contexts are many. For instance, in the context of a presidential transition or handover, a newly elected president is preparing himself for ruling the country [1]. Here, it is not only a matter of the incoming president taking over the responsibilities of the outgoing president, but it is also a matter of designating new government personnel, training them, and assuring that they are ready to take over the country in as smooth way as possible. The goal is to make the transition as seamless as possible for the incoming president and his administration. If the presidential transition fails, then the country may not be sufficiently postured both administratively and militarily, and thereby, get exposed to various internal and external threats [2] [3].

In the context of economic transition, an economy may be transitioning from a centrally planned economy to a free market [4] [5]. This implies many changes to be done such as, for instance, changes of the responsibilities of the state and the private enterprises. Here, the state changes its role from the provider to the enabler of economic growth and the private enterprises take over the role of the drivers of the market economy [6]. To assure a smooth transition, one must plan for and manage the changes to be done to large-scale and small-scale privatizations, make changes to governance and banking, introduce competition policies, and liberalize the prices and economic activities.

In the context of healthcare or transitional care, transition implies a transition of providing care to patients from one set of services to another [7]. Patients may have different and complex care needs depending on (1) whether they need to transfer from child to adult health services, (2) whether they need daily elderly service, or (3) whether they need urgent assistance in case of acute illness or other. The goal is to ensure that the patients are smoothly transferred between different care takers and/or different care levels. To assure smooth transitional care, one must plan for it, assure that a well-trained staff is available, have solid information about the patient and his clinical status, and educate the patient and his family. One must also have a process for coordinating the transitional care so that one does not jeopardize the patient’s safety and well-being.

Another example of a transition might be in a context of transferring products from one organization to another [8]. For instance, a manufacturer of cars transfers the responsibilities for providing post-sale support services to a support organization. The manufacturer should not only transfer the support responsibilities but also knowledge about the cars, software and hardware needed for servicing the cars, provide replaceable components, documentation and operational instructions necessary for understanding the car design and for operating and servicing the car.

In the context of software systems, we may find many examples of transition. For instance, a development organization transfers or hands over a software system to a maintenance organization [9]. Here, the development organization does not only transfer the responsibilities for evolving and maintaining the system in the future but it also transfers
the system itself, its documentation, operational data, knowledge about the system and many other issues.

Just as with all the above-described types of transition alias handover, it is important that a software system transition is performed in a smooth way. Therefore, a good transition process model is imperative and critical for planning and managing a transition and for alleviating many transition problems. Failing to transfer software systems may lead to loss of productivity, loss of maintainer credibility, loss of system and maintenance process quality, and sometimes, even loss of business.

1.1 Problem Statement

Even if the transition process is critical for succeeding with transferring software systems from development to maintenance, there still do not exist any up-to-date transition process models that designate important features of a transition process [10].

Regrettably, the issue of the software transition is still a strongly under-researched and neglected domain. So far, the research community has not paid much attention to the transition process. The published transition process models are either too old or they deal with handover on either a very general level or in an unstructured way [9] [11] [12] [13].

By being too general, the published transition models provide space for misinterpretations, and by being unstructured, they fail to point out process features that are critical for the transition process. Therefore, we conclude the software engineering community lacks transition process models on an appropriate granularity level that meets the industrial requirements and needs.

1.2 Research Goal

The primary goal of this thesis is to explore the domain of the software system transition alias handover, to contribute to the increased understanding of its domain, scope, activities and the roles involved, and thereby, to create a basis for creating software system handover process models. The goal fulfillment is driven by the following research questions:

- **Research Question 1**: What are the activities important for planning and executing software handover process?
- **Research Question 2**: What are the different contexts of software handover?
- **Research Question 3**: What are the guidelines for implementing handover activities?
- **Research Question 4**: What points in time in the system lifecycle do handover activities get implemented?
- **Research Question 5**: What are the different roles participating in the handover process?
1.3 Contribution

The contribution of this thesis is *Evolution and Maintenance Management: Handover Framework* abbreviated as *EM³: Handover Framework*. The framework is a conceptual structure including six parts that constitute building blocks of a handover process. The six parts are (1) Handover Taxonomy, (2) Handover Types, (3) Handover Contexts, (4) Handover Roles, (5) Handover Lifecycle Roadmap, and (6) Handover Guidelines. *EM³: Handover Taxonomy* has the central importance in the framework. It comprises component practices including activities that are required for transitioning (handing over) software systems from development to maintenance. The taxonomy provides a platform for creating handover process models. It is structured into eight categories of handover activities that are put into the following practices: (1) Management and Administration, (2) Maintenance Environment, (3) Version and Configuration Management, (4) Deployment, (5) Training, (6) Documentation, (7) Maintainability Management and (8) Software System Transfer. In addition to the taxonomy activities, this thesis contributes with the following:

- **Handover Types** identifying different types of handover.
- **Handover Contexts** identifying various transition contexts pointing out the parties involved and the points in time in the software system lifecycle in which transition process takes place.
- **Handover Roles** identifying the roles and their responsibilities that are crucial for conducting the transition process.
- **Handover Lifecycle Roadmap** designating suitable phases in the software system lifecycle to implement handover activities.
- **Handover Guidelines** providing explanations and suggestions for constructing their handover processes.
- **Handover Problems** listing currently perceived transition problems within the software industry, the problems that may substantially jeopardize the business of all the parties involved.

1.4 Bodies Involved

Every step of our research involved interaction with the software industry. As many as sixty one organizations were involved in this study. Out of them, three organizations intensively supported our research by exposing their transition processes to our detailed scrutiny, five organizations contributed to eliciting problems, eighteen organizations contributed to determining the contexts of the transition process, thirty six organizations participated in a large scale exploration of the handover domain and one organization enabled us to implement and evaluate *EM³: Handover Framework* in the context of their handover process.
1.5 Target Groups

Our main target group includes the organizations involved in software handover. These are acquirer, developer, and maintainer organizations. Our taxonomy will guide them in implementing and improving their handover processes and in identifying problems and challenges they may face before, during and after the handover. Developers and maintainers can use the taxonomy as guidelines on how to conduct and work together during the software handover. Acquirers will be able to get insight into how their acquisition processes interact with the software handover processes.

Our second target group includes academia. Students and researchers can gain a detailed insight into the software handover work done so far. Students can use our work to learn about the process whereas researchers can extend our framework and use it for suggesting their own handover process models.

Finally, we should not exclude standardization organizations as our third target group. We believe that all the theory that we have generated while constructing our framework is an important feedback for revising and improving the currently existing software engineering standards.

1.6 Transition vs. Handover Terminology

The currently available literature uses two terms when referring to the process of transferring software systems from development to maintenance. These terms are handover and transition. According to Merriam-Webster dictionary [14], these terms mean merely the same thing. Handover means to give something over to the control or possession of another party, whereas transfer means to give over the legal possession or ownership to another party.

The transition term has been perceived differently in industry. It has been used for referring to a “transition of technology” where a system’s outdated technology is replaced with the new one [15]. The transition term is also used when referring to a transfer of a software system to operation [16]. However for us, the term transition is used when referring to a transfer (handover) of a software system from development to maintenance and the responsibilities for managing the system [9]. We consider the term handover more elaborative to represent the domain under discussion. However, since literature uses the terms transition and handover interchangeably, so do we so as well.

1.7 Limitations

Software handover is a very complex and diverse process whose activities are widely spread in the software lifecycle and strongly overlap with the activities in the development and maintenance processes. When choosing the topic of this thesis, we were not aware of its magnitude, complexity and enormous diversity. Therefore, due to resource and time limitations, we suggest a framework for software handover to be used for creating
handover process models. It was not possible for us to suggest a process model due to the following reasons:

- There are many types of handover. Our study mainly deals with software system handover from developer to maintainer. It does not focus on handover taking place due to acquisition and merges of software organizations. Also, it does not cover handover taking place in global software development (GSD), where multiple teams located at different geographical locations work on developing the same system. However, we believe that our work may be easily adapted to these domains and serve their specific needs.

- This thesis lists activities, roles and handover lifecycle phases to be used when implementing software handover. However, the choice of the roles to be involved in the process, the choice and sequence of the execution of the handover activities and their placement on the handover lifecycle phase differ depending on the context and type of handover. Hence, this thesis only designates the main handover roles, suggests a handover lifecycle roadmap, suggests the placement of the activities on the handover lifecycle roadmap, and provides guidelines aiding in the construction of handover processes. We cover various categories of activities in the handover taxonomy. However, it is not necessary to implement the whole taxonomy to achieve successful handover. Depending on the context at hand, organizations can utilize part of EM’s taxonomy that fulfills their particular needs.

- The handover framework described in this thesis can be utilized for transitioning small, medium and large scale systems in small, medium and large organizations. In general, small and medium-sized organizations may already take full benefit by implementing part of the taxonomy according to their handover contexts when transferring their small and medium sized systems. It is only in case of large scale systems and organizations, where the organizations need to implement the whole or major part of the taxonomy in order to make sure that they succeed with the handover process.

- Software systems handover strongly overlaps with other processes that are conducted in different software system lifecycle phases. These are development, testing, deployment, quality management and version and configuration management. All of the activities belonging to those processes are important for making the handover process successful. Our taxonomy however only lists their activities that are critical for the handover process.

1.8 Thesis Outline

The thesis consists of three major parts, namely Part I, Part II and Part III.

Part I consists of five chapters and they are the following:
• Chapter 1 – Introduction providing an overview of the handover domains. The chapter discusses the research problem, research goal and research contribution.

• Chapter 2 – Software System Handover discussing the background and state of the art in the area of software handover.

• Chapter 3 – Research Process discussing the overall research process that we followed in our research. The chapter describes each stage of our research, research approaches, research tools, validity threats and presents the organizations that participated in this research.

• Chapter 4 – Research Contribution summarizing our research contribution in the handover domain. Here, we provide the summaries of our research publications constituting the combined result of our research work.

• Chapter 5 – EM³: Handover Framework presenting the framework of software handover and suggestions for how to use it.

• Chapter 6 – Epilogue rounding up our overall research results and suggesting future work.

Part II consists of four appendices. These are the following:

• Appendix A – Handover Taxonomy Versions presenting different versions of EM³: Handover Taxonomy thus illustrating how it evolved with time.

• Appendix B – Questionnaires presenting the questionnaires used in the interviews.

• Appendix C – Templates presenting the templates used for collecting and organizing data in different stages of research work.

• Appendix D – Bodies Involved presenting the details of organizations involved in different phases of research.

• Appendix E – EM³: Handover Taxonomy Guidelines presenting guidelines to be used when constructing software system handover process models.

• Appendix F – EM³ providing an overview Evolution and Maintenance Management Model and its components.

Part III comprises the fourteen publications that constitute the research body of this thesis.

1.9 List of Publications

In total, we have written fourteen publications dealing with software system handover and two publications dealing with the related subject of pre-delivery maintenance. Due to their substantial amount, we cannot include all the papers in this thesis. Therefore, we exclude the papers on pre-delivery maintenance process and we only focus on the handover process instead. To get a full overview of our research, we advise our reader to even study our work on pre-delivery maintenance. Below, we list all our publications. The papers dealing with the handover process are 1-14 and the papers dealing with the pre-delivery maintenance process are 15-16.
Paper 1: Core Handover Problems
Ahmad Salman Khan and Mira Kajko-Mattsson, in proceedings of the 11th International Conference on Product Focused Software, PROFES 2010, pp. 135-139, ACM
Ahmad Salman Khan contributed with the 80% effort to the paper.

Paper 2: Taxonomy of Handover Activities
Ahmad Salman Khan and Mira Kajko-Mattsson, in proceedings of the 11th International Conference on Product Focused Software, PROFES '10, pp. 131-134, ACM
Ahmad Salman Khan contributed with the 80% effort to the paper.

Paper 3: Demarcating the Scope of a Handover Process
Ahmad Salman Khan contributed with the 80% effort to the paper.

Paper 4: Evaluating A Taxonomy of Handover Activities in One Swedish Company
Ahmad Salman Khan contributed with the 45% effort to the paper.

Paper 5: Towards Taxonomizing of Core Software Handover Activities
Ahmad Salman Khan contributed with the 80% effort to the paper.

Paper 6: Evaluation of Handover Activities from the Perspective of the System Acquisition and Front-End Support
Ahmad Salman Khan contributed with the 45% effort to the paper.

Paper 7: Walking a Software Handover Process
Ahmad Salman Khan contributed with the 45% effort to the paper.
Paper 8: Identifying and Tackling Diversity of Management and Administration of a Handover Process
Ahmad Salman Khan and Mira Kajko-Mattsson, 12th International Conference on Product-Focused Software Process Improvement, Torre Canne, Italy, 2011, pp. 156-170, Lecture Notes in Computer Science; Springer
Ahmad Salman Khan contributed with the 80% effort to the paper.

Paper 9: Managing Versions and Configurations in a Software Handover Process
Ahmad Salman Khan, Mira Kajko-Mattsson, 8th International Conference on Computing Technology and Information Management (ICCM), 2012, Volume 1, Pages 252-258, IEEE
Ahmad Salman Khan contributed with the 80% effort to the paper.

Paper 10: Evaluating the Role of Maintenance Environment Activities in Software Handover
Ahmad Salman Khan, Mira Kajko-Mattsson, 8th International Conference on Computing Technology and Information Management (ICCM), Seoul, 2012, Volume1, Pages 230-237, IEEE
Ahmad Salman Khan contributed with the 80% effort to the paper.

Paper 11: Evaluating a Training Process in a Software Handover Context
Ahmad Salman Khan contributed with the 80% effort to the paper.

Paper 12: Analyzing Deployment from the perspective of Software Handover and Maintenance
Ahmad Salman Khan and Mira Kajko-Mattsson, in proceedings of 8th International Conference on Computing Technology and Information Management, ICCM 2012, pp. 244-251, 2012 Seoul, IEEE.
Ahmad Salman Khan contributed with the 80% effort to the paper.

Paper 13: Management of Documentation and Maintainability in the Context of Software Handover
Ahmad Salman Khan contributed with the 80% effort to the paper.

Paper 14: Evaluating and Acting on the Handover Framework in a special Case of Project Parking
Ahmad Salman Khan contributed with the 80% effort to the paper.
Other publications

Paper 15: Comparing EM3: Predelivery maintenance model with its industrial correspondence
Ahmad Salman Khan, Mira Kajko-Mattsson, Tommy Tyrberg, in proceedings of International Multiconference on Computer Science and Information Technology, IMCSIT’09, 2009, pp 573-582, IEEE
Ahmad Salman Khan contributed with the 80% effort to the paper.

Paper 16: Demarcating the Scope of a Predelivery Maintenance Process
Ahmad Salman Khan contributed with the 80% effort to the paper.
2. Software System Handover

This chapter presents the software system handover domain as described in the current scientific literature. Its goal is to provide background to the handover domain and its related subjects. The chapter first presents the different definitions of handover in Section 2.1. It then discusses software handover in the context of a software system lifecycle and pre-delivery maintenance in Section 2.2. It continues to describe transition models available today in Section 2.3. Finally, Section 2.4 describes IT outsourcing and its relation with handover.

2.1 Software Handover

Just like all other disciplines of life, software systems undergo many stages starting from their conception and ending on retirement. As illustrated in Figure 2.1, their main stages include initial development, evolution and maintenance, and retirement. They, in their turn, consist of sub-stages or phases. For instance, initial development consists of phases such as requirements specification, design, implementation, testing and delivery. The evolution and maintenance phase include similar phases as in development. They however differ from development with respect to the fact that different roles may conduct evolution and maintenance and that these roles have to now consider the restrictions of the existing system.

In addition to the above-mentioned stages and their phases, there are processes that are not directly classified as development and or maintenance. These processes prepare software systems and organizations for future maintenance and they manage the transfer of the systems from the groups responsible for developing the systems to the groups responsible for evolving and maintaining them. As shown in Figure 2.1, two of such processes are pre-delivery maintenance and handover. Both processes may be difficult to map on the typical software lifecycle stages due to the fact that they may run in parallel with both development and maintenance.

Pre-delivery maintenance is the process during which both development and maintenance organizations/teams prepare themselves and their systems for future maintenance. It is the

![Handover process in the system lifecycle](A simplified figure of system lifecycle)
process that is a prerequisite for conducting software handover. It is discussed in Section 2.2.2. Software handover, on the other hand, is the process where the maintenance organizations/teams take over the responsibilities for managing a software system from the organization/team performing development.

As shown in Figure 2.1, handover is illustrated with the arrow between the initial development and evolution and maintenance stages. The scope of the handover process however does not take place exactly on the meeting point between the two stages. On the contrary, the handover process should start in parallel with the initial phases of development and end in the initial phases of evolution and maintenance. As Pigoski claims, “you cannot deliver a software system on Friday and start maintaining it on Monday. You must prepare for this” [9]. This preparation is done by performing pre-delivery maintenance and handover processes, the two processes that together prepare the software maintenance organizations/teams for performing their future work.

Not much research has been done within software handover. Therefore, there are not so many definitions in the literature. We have succeeded to identify six definitions that have been proposed by T. Vollman [17], T. Pigoski [9], A. April [11], ISO/IEC 14764 [12], ISO/IEC 15288 [13] and ITIL Service Transition [18]. All these definitions use the term “transition”. They are listed in Table 2.1.

According to Definition 1 (the Vollman’s definition - see Table 2.1), transition is a major milestone during which the software items cease to be the responsibility of the developer and become the responsibility of some other organization. This definition does mention the term developer for representing the team handing over the system. However, it does not use the term maintainer standing for the team taking on the responsibility for the transitioned system. It rather calls it “other organization.”

According to Definition 2 (Pigoski’s definition - see Table 2.1), handover is a controlled and coordinated activity during which a software system is transferred from the team performing initial software development to the team performing post-delivery software maintenance and support. Here, the definition distinguishes between both development and maintenance teams. The development team transfers the software, hardware, documentation and knowledge of the system. The maintenance team gets these artifacts and prepares itself for future maintenance and support [9].

There are two interesting observations in Definition 2. First, it says that the transition occurs after initial development of the system. Second, unlike Definition 1, it uses both the term maintenance and support to be taken over by the team taking on the responsibilities for the system.

Transition has also been defined in Military Handbook for mission critical computer resources software support [19]. It is exactly the same as Definition 2 [9]. In addition, it defines the software transfer term in relation to transition where the “software transfer is the
Table 2.1. Transition definitions

<table>
<thead>
<tr>
<th>Title</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Definition 1 Thomas Vollman</td>
<td>The lifecycle of many large software systems includes a major milestone for delivery and a transition phase during which the software item ceases to be the responsibility of the developer and becomes the responsibility of some other organization. [17]</td>
</tr>
<tr>
<td>Definition 2 Thomas Pigoski</td>
<td>A controlled and coordinated activity during which the software system is transferred from the team performing initial software development to the team performing postdelivery software maintenance and support. [9]</td>
</tr>
<tr>
<td>Definition 3 Alain April</td>
<td>The process of transferring the responsibilities for the acquired software products from the project manager to the software support organization. [11]</td>
</tr>
<tr>
<td>Definition 4 ISO/IEC 14764</td>
<td>Controlled and coordinated sequence of actions wherein software development passes from the organization performing initial development to the organization performing software maintenance. [12]</td>
</tr>
<tr>
<td>Definition 5 ISO/IEC 15288</td>
<td>The purpose of transition is to establish a capability to provide services specified by stakeholder requirements in the operational environment. [13]</td>
</tr>
<tr>
<td>Definition 6 ITIL Service Transition</td>
<td>A change in state, corresponding to a movement of an IT Service or other Configuration Item from one lifecycle status to the next. [18, p. 25]</td>
</tr>
</tbody>
</table>

point when the Software Support Activity (SSA) assumes responsibility for Post Deployment Software Support (PPDS). It is the last step in software transition” implying that software transition ends once the software is transferred to the support team.

Definition 3 (the April's definition - see Table 2.1) defines transition as the process of transferring the responsibilities for the acquired software products from the project manager to the software support organization. It identifies the roles such as project management and software support organization. According to this definition, the term – software support stands for the overall responsibilities and activities of the software maintenance personnel”. [11]

Definition 4 (ISO/IEC 14764 definition – see Table 2.1) resembles Definition 2. However, it uses the term sequence of actions rather than the term activity and it uses the simple term software maintenance, rather than the terms post-delivery software maintenance and support.

Definition 5 (ISO/IEC 15288 definition – see Table 2.1) does not use the term development or maintenance. It rather considers transition as providing services in the operational environment. Here, the transition concept strongly overlaps with deployment. Finally, Definition 6 (ITIL Service Transition definition – see Table 2.1) considers transition as a change in the lifecycle status of a service or a configuration item.

2.2 Software Maintenance and Transition

In general, software transition has been discussed as part of a maintenance process in the literature. Hence, it feels natural to provide an overview of the maintenance process and its relevance to the transition process. In this section, we describe software maintenance and its relation to the software transition.
2.2.1 Software Maintenance

Software maintenance is the totality of activities required for providing cost-effective support to a software system. Its activities are performed during the pre-delivery stage as well as the post-delivery stage [12]. Maintenance process can be a simple process where only one software engineer attends to defects or it can be a complex process where a separate maintenance team or even organization is fully dedicated to performing maintenance services [20].

ISO/IEC 12207 standard on “System and Software Engineering – Software Lifecycle Processes” establishes a framework for software lifecycle process. The standard places maintenance process as a Technical Process under the System Context Processes category [21]. Since the standard scope is wide and it covers all the lifecycle processes, it discusses maintenance process at a general level and it does not discuss the transition process at all.

A more detailed and comprehensive discussion on maintenance process can be found at ISO/IEC 14764 standard on “Software Engineering Software Life Cycle Processes – Maintenance” [12]. ISO/IEC 14764 defines software maintenance as “the totality of activities required to provide cost-effective support to a software system. The activities are performed during the pre-delivery stage as well as the post-delivery stage” [12, p. 4].

ISO/IEC 14764 describes in detail the management of the maintenance process. It provides guidance that applies to planning, execution and control, review and evaluation, and closure of the maintenance process. According to this standard, the maintenance process contains activities necessary for modifying an existing software system. As shown in Figure 2.2, the standard lists six activities inherent in the maintenance process. These are Process Implementation, Problem and Modification Analysis, Modification Implementation, Maintenance Review/Acceptance, Migration, and Retirement.

The standard describes different maintenance types. It classifies modification requests into two major categories [12]. These are Correction and Enhancement. The Correction category contains corrective and preventive type of maintenance, while the Enhancement category contains adaptive and perfective type of maintenance [12, p. 4].

2.2.2 Pre-delivery Maintenance

There are not many publications dealing with the pre-delivery maintenance. The main publication is mainly the ISO/IEC 14764 standard and some book chapters written by Thomas Pigoski [9].

ISO/IEC 14764 classifies maintenance activities into two categories. These are pre-delivery maintenance activities and post-delivery maintenance activities. Pre-delivery activities are performed before the system delivery and their goal is to prepare the software system and maintenance organization for future maintenance. Post-delivery maintenance activities begin after the delivery of the system and continue till the system gets retired [9, p. 81]. The standard emphasizes the importance of early involvement of the maintainer in
the development process and describes the maintainer’s responsibilities during the pre-delivery and post-delivery stages in the maintenance plan.

ISO/IEC 14764 puts emphasis on designing a maintenance strategy. The strategy consists of maintenance concept, maintenance plan and resource planning. The maintenance concept should discuss the scope of maintenance, define the overall maintenance process, designate the maintainer, and estimate maintenance cost.

The maintenance strategy provides a basis for designing a maintenance plan. The plan should be a comprehensive document including topics like definitions, maintenance concept, pre-delivery and post-delivery roles and the responsibilities of the maintainer, hardware and software resources required, data management, and training. The maintainer’s responsibilities are listed in the Table 2.2. As can be seen there, it places the development of transition plan as part of the pre-delivery maintenance responsibilities.

It is worthwhile to mention that Thomas Pigoski participated as the editor of the standard. Therefore, the topics discussed in the standard and the book chapters and articles published by Pigoski [9] strongly overlap with each other.

2.2.3 Software Transition

Not much has been done within the area of software transition. The work we could base our work was mainly done by [17] [9] [22] [23] [12]. In this section, we first provide an overview of the handover state of art. We then describe the two transition models on which we based our research.

2.2.3.1 State of Art

There are a few works, models, and standards that were relevant for our study. These are the models created by Thomas Vollman, Thomas Pigoski, and transition related standards
Table 2.2. Pre-delivery and post-delivery responsibilities of maintainer [12]

<table>
<thead>
<tr>
<th>Pre-Delivery roles and responsibilities of the maintainer</th>
<th>Post-Delivery roles and responsibilities of the maintainer</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) Process Implementation</td>
<td>(14) Interface with other organizations</td>
</tr>
<tr>
<td>(2) Establish Infrastructure</td>
<td></td>
</tr>
<tr>
<td>(3) Establish Human Resource Process</td>
<td></td>
</tr>
<tr>
<td>(4) Establish the Software Maintenance Process</td>
<td>(13) The schedule estimating method</td>
</tr>
<tr>
<td>(5) Develop the maintainability plan</td>
<td></td>
</tr>
<tr>
<td>(6) Monitor development execution for maintainability</td>
<td>(10) Factors that determine organizational maintenance priorities</td>
</tr>
<tr>
<td>(7) Develop the transition plan</td>
<td>(9) Improve the process</td>
</tr>
<tr>
<td>(8) Participate in the maintainer in development activities</td>
<td>(11) The process for assigning a priority to a work packages</td>
</tr>
<tr>
<td>(9) Interface with other organizations</td>
<td></td>
</tr>
<tr>
<td>(4) Maintenance Review/Acceptance</td>
<td>(5) Migration</td>
</tr>
<tr>
<td>(5) Migration</td>
<td>(6) Retirement</td>
</tr>
<tr>
<td>(7) Problem Resolution (includes Help Desk)</td>
<td>(8) Train personnel (maintainer and user), as applicable</td>
</tr>
<tr>
<td>(8) Particpation by the maintainer in development activities</td>
<td></td>
</tr>
<tr>
<td>(9) Interface with other organizations</td>
<td></td>
</tr>
</tbody>
</table>

including ISO/IEC 14764, ISO/IEC 15288 and ITIL Service Transition. They all provide different angles of software transition. Below, we briefly present them.

Thomas Vollman identifies management and technical issues of transition in the context of large software systems. His work includes identification of main procedures important for software transition. Moreover, he identifies the major roles participating in the transition. We describe them later in the Section 2.3.

Pigoski divides transition into two parts. These are (1) software development transition and (2) software maintainer transition [9, p. 120]. The software development transition is concerned with the activities to transfer the software development responsibilities from developer to maintainer. It includes transfer of hardware, software data and experience. The maintainer transition includes activities for smooth transition from development to post-delivery maintenance. It includes activities for staffing, training, development of configuration management plans, hardware and software installation, replicating maintenance problems and creating software builds [9, p. 121].

ISO/IEC 14764 standard places transition as part of the maintenance process execution. It discusses software transition at a very abstract level. It does not provide any information regarding transition process, its placement in the system lifecycle, its activities and the scope of those activities. The only information we found was the transition plan and its candidate topics. These topics are listed in Figure 2.3. As can be seen there, a transition plan should address hardware, software, data, support services and transfer of experience from development to maintenance organization. Moreover, it should identify the tasks needed to implement transition strategy including staffing, training, installation, and replicating maintenance problems. It should also assess knowledge transfer and documentation, assess the test environment readiness, and list outstanding problems and new requirements by priority.
The ISO/IEC 15288 standard describes two major activities for transition. These are (1) plan transition and (2) perform transition. Planning for transition includes defining a transition strategy and preparing the operation site according to the installation requirements, whereas performing transition includes tasks to deliver, install and activate the system in an operational environment. The standard only discusses transition from development to operation, and it does not discuss transition from development to maintenance.

It is worthwhile to mention that we were not able to find any publication mapping transition on the lifecycle of a software system. As can be seen in Figure 2.4, ISO/IEC 15288 categorizes transition as a technical process, however it does not show the span of the transition process on the software system lifecycle.

ITIL Service Transition is a standard by Office of Government Commerce (OGC). The ITIL standard claims to provide guidance for transitioning a new or changed service to operation. Its goal is to “assist organizations seeking to plan and manage service changes and deploy service releases to the production environment successfully. [18, p. 27].

ITIL pointed out seven process areas [18, p. 61] for successful service transition. These are Transition Planning and Support, Change Management, Service Asset and Configuration Management, Release and Deployment Management, Service Validation and Testing, and Evaluation and Knowledge Management.

The focus of ITIL Service Transition is on defining activities and best practices for deploying service releases to the production environment. It is mainly concerned with decommissioning of an older service and deployment and operation of the new service.

- The transfer of hardware, software, data, support services, and experience from the developer to the maintainer
- The tasks needed for the maintainer to implement the software maintenance strategy (e.g., staffing, training, installation, replicating maintenance problems)
- Assessing knowledge transfer and documentation
- Outstanding problems and new requirements by priority
- Assessing the test environment readiness
- Transfer of as-built configuration information for the source code and object code, including open or deferred problem reports and new requirements, number and location of media masters which may need to be updated during maintenance.

Figure 2.3. Transition plan topics ISO/IEC 14764
The standard provides detailed and comprehensive descriptions of its process areas without pointing out which of their activities are critical for the handover process.

### 2.3 Transition Models

We based our work on two transition models. These are (1) Transition Model 1 and (2) Transition Model 2. Transition Model 1 discusses transition approach described by Thomas Vollman [17] while Transition Model 2 describes transition activities defined by Thomas Pigoski [9]. Both transition models do not have any specific names. Therefore, we distinguish them by calling them Transition Model 1 and Transition Model 2.

#### 2.3.1 Transition Model 1

*Transition Model 1* was described by Thomas Vollman in a research publication [17]. It identifies management and technical issues of transition with respect to large software systems implemented in a public sector organization. It proposes an approach to transition planning and implementation. It identifies five major roles participating in the transition process. These are *Project Management Office (PMO)*, *Technical Development Group (TDG)*,
Software Support Activity (SSA), Vendor and Customer. The customer organization advertises a competition for developing the system. A vendor wins the contract and it is the vendor’s responsibility to develop the system and deliver the hardware, software, documentation and training material. A separate contract is awarded for independent verification and validation.

Project Management Organization (PMO) at the customer side is responsible for monitoring the contract progress. The Technical Development Group (TDG) is responsible for providing necessary technical directions. It includes representatives from PMO, future user and if possible, future maintainer. The maintenance responsibilities are assigned to a separate team in the user organization called Software Support Activity (SSA). SSA is responsible for all aspects of software maintenance including acceptance, installation, future upgrades, documentation, testing, delivery and training. [17]

Transition Model 1 points out management and technical problems of transition. Management problems include inadequate staff and training for operation and support of the system. Technical problems include significant loss of corporate knowledge when the development ends and when SSA starts providing support. This results in sharp decrease in the ability to support the system. Other problems include unavailability of software and software support suites for providing support services.

Transition Model 1 claims that the transition plan is the key element for successful transition. It argues that the transition plan should be developed by SSA as the ultimate burden of its implementation would fall on SSA. The transition plan should identify transition problems and then define different tasks to solve the problems.

Transition Model 1 divides SSA function into nine categories. As shown in Figure 2.5, these are Management and Administration, Software Engineering and Testing, Configuration Management, Software Generation and Distribution, Training, Technical Documentation, On-site support, Deployment and Installation and Quality Assurance. [17]

The goal of the Management and Administration procedure is to ensure that management function is in place to coordinate and control the activities of various participants in the maintenance program. The goal of Software Testing and Engineering procedure is to ensure that the structure is in place to efficiently modify and test the software during the maintenance phase. [17]

Regarding the Configuration Management procedure, Transition Model 1 claims that the SCM baseline must be established when the initial installation and test activities have been completed and reviewed. This includes entering software baseline, support software and database into configuration management. Regarding Software Generation and Distribution, it is important that a software product does not only work correctly at the SSA site but also that it works correctly at the user’s site. SSA should develop procedures to facilitate to the system user to use and understand the system. [17]
Regarding Training, SSA training program should cover the overview of the system, its architecture and how to use it. SSA may or may not be responsible for the overall training. However, it should develop a training program plan and at least train its own personnel. SSA should also define its standard operating procedures and it should develop and update the training material for both system operation and maintenance. [17]

Regarding Technical Documentation, Transition Model 1 claims that development team pays less attention on technical documentation. If documentation is not up-to-date, then there should be a plan to update the documentation during transition or maintenance. Documentation should be stored in a repository and it should be at the same level of configuration management as code. Documentation should follow the standards. If the standards do not exist, then they should be defined. [17]

Regarding On-site Support, one of the major duties of SSA is to provide onsite support. A set of training material and resources required must be identified and made available to the onsite support personnel. Finally, in the Installation and Deployment procedure, SSA is responsible for installing and for establishing installation plans. [17]

Regarding Deployment and Installation, the installation plans must be established defining interfaces, deployment schedule and necessary technical support and training material. Regarding Quality Assurance, the quality program must be in place and quality assurance activities must be performed in parallel to transition. The quality program must be reviewed for improvements at the end of transition.

2.3.2 Transition Model 2

Thomas Pigoski describes software transition in three chapters in his book on the topic of software maintenance [9]. Moreover, he presents transition in two research articles [22] [23]. The contents of the articles have been derived from the book chapters. Hence, we are discussing them together.
Table 2.3  Pigoski transition model activities

| Establish the organization for transition | Provide requests to user and all other concerned |
| Get identified as a maintenance organization | Take code from developer and put under automated SCM. |
| Establish transition group | Perform version control |
| Influence development contract | Exercise builds procedure |
| Hire SCM personnel first, tester second and programmers last | Provide something to user |
| **Perform pre-delivery function** | Setup an early help or trouble desk |
| Attend review | Provide system training to user |
| Do independent verification and validation. | Perform testing function |
| Analyze metrics | Conduct white box testing |
| **Establish a sold SCM program** | Perform a maintainability assessment of the software |
| Take development modification requests | Perform software modifications |
| Place requests in a status accounting database | Fix low priority development modification requests |

The book chapters introduce transition, the two parts of transition mentioned above (software development transition and software maintainer transition), maintenance concept, major transition issues and sample transition plan [9, p. 117]. Moreover, they cover practical experiences gained during transition in form of case studies. According to [9, p. 117], software transition is the transfer of software product from the developer to the maintainer.

Transition Model 2 describes the maintenance concept by combining characteristics such as the scope of software maintenance, tailoring of the maintenance process, designation of future maintainer and estimation of lifecycle costs [9, p. 90]. It also points out major issues in transition. These are transfer of knowledge, documentation, communication with developer and user, training, cost of transition and designation of future maintainer [9, p. 129].

Transition Model 2 outlines a transition process consisting of six phases. As shown in Figure 2.6, these are (1) establish an organization for handover, (2) perform pre-delivery function, (3) establish a solid SCM program, (4) provide something to user, (5) perform testing function, and (6) perform software modifications. Table 2.3 lists their inherent activities.

The application of Transition Model 2 is described in two case studies in which a newly built system was handed over from development to maintenance. In the first case study, Pigoski focused on the structure and capability of the maintenance team and discussed the problem of lack of documentation. In the second case study, he focused on placing system under automated version control and transfer of knowledge to the maintenance team through
training. He also pointed out the transition activities which must be followed in the pre-delivery and post-delivery phases. These are listed in the Table 2.4.

2.4 Transition and IT outsourcing

In this section, we discuss the information technology outsourcing domain and its relationship to transition. The outsourcing domain is related to handover and its ideas might be found useful within our research [24] [25] [26] [27].

2.4.1 Outsourcing Definition

Outsourcing is defined as the transfer of activities and processes previously conducted internally to an external party [28]. It is a process of transferring the responsibilities for a specific business function from an employee group to a non-employee group [29]. It can be performed domestically or internationally. Domestic outsourcing is known as onshore outsourcing, whereas international outsourcing is known as offshore outsourcing [28].

There are many functions in an organization which may be outsourced. These include application development, maintenance and support, software quality assurance, security functions, technical support and transaction processing [30] [31] [32]. In general, organizations outsource the non-core functions so that the organizations themselves may focus on core functions for gaining more competitive advantage [33] [34] [35] [36] [37] [38].

Table 2.4. Pigoski transition activities [23]

<table>
<thead>
<tr>
<th>Pre-delivery maintenance activities</th>
<th>Post-delivery maintenance activities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Develop and select a post deployment software support strategy</td>
<td>Implement the transition plan</td>
</tr>
<tr>
<td>Influence mission critical computer resources product definition</td>
<td>Acquire and install all equipment</td>
</tr>
<tr>
<td>Propose software quality requirements</td>
<td>Acquire maintenance resources requirements</td>
</tr>
<tr>
<td>Identify maintenance resource requirements</td>
<td>Staff train personnel</td>
</tr>
<tr>
<td>Develop the computer resources lifecycle management plan</td>
<td>Manage post deployment software support strategy</td>
</tr>
<tr>
<td>Develop the transition plan</td>
<td>Conduct post deployment support strategy operations</td>
</tr>
<tr>
<td>Ensure software supportability</td>
<td>Provide software product logistic support</td>
</tr>
<tr>
<td>Evaluate software quality</td>
<td>Evaluate and maintain software quality</td>
</tr>
<tr>
<td>Certify software document and technical data</td>
<td>Perform configuration management function</td>
</tr>
</tbody>
</table>
2.4.2 Main challenges of IT Outsourcing

Implementation

Offshore outsourcing is considered to be a more challenging task than onshore outsourcing [39, p. 140]. We have identified different problems which may occur during outsourcing to an offshore company. These concern time zone differences, knowledge transfer, cultural differences, rigorous communication of requirements and difficulties in managing dispersed teams. These problems can be tackled by appointing middlemen, by increasing a number of contact persons, and by designing a special interface between the client and offshore supplier. [39]

Transfer of system knowledge and business process knowledge emerged as a major challenge in implementing offshore services [40]. The outsourcing from in-house to offshore or onshore supplier is a challenging task. In case of onshore outsourcing, the in-house staff may have to also be transferred to the supplier [40, p. 82]. However, in case of offshore outsourcing, knowledge transfer to supplier becomes a critical issue. Here, the in-house staff has to intensively communicate with offshore supplier to transfer system knowledge and business processes knowledge [40, p. 82].

A good way of transferring knowledge is that offshore supplier employees visit the customer and learn the system and processes. The employees then continue working on the system after the transition. It is very important that the offshore supplier retain the employees, otherwise the customer has to arrange training sessions again for the new employees, which is considered a time consuming activity [40, p. 82]. It is also important that the process and system documentation are ready before the system is transitioned to the offshore team. The customer must appoint its own project manager who should be responsible for communicating with the project manager on the offshore supplier side.

2.4.3 Transition process in IT Outsourcing

We found four publications describing transition as part of an outsourcing process. These are (1) Zhu et al. [41] and (2) Scott Lever’s publication discussing transition as part of outsourcing stages [42], (3) a book by E. A. Sparrow on offshore outsourcing models [40], and (4) a book by M. J. Desouza on outsourcing process implementation [43]. We briefly describe them below.

Zhu et al. describe an outsourcing process that comprises four stages [41]. As shown in Figure 2.7, these are planning, developing, implementing and surviving. During the implementation stage, the outsourcing transition takes place and it requires a detailed transition plan [29]. The transition plan identifies all the tasks that must be performed for successful transition. The nature of the transition plan topics depends on the type of function to be transitioned. The suggested topics are listed in Figure 2.8. Moreover, the parties involved in transition should develop a checklist comprising all the items identified in the transition plan.

Another outsourcing model was presented by Scott Lever [42]. It comprises four stages including discovery, negotiation, transition and assessment. It is shown in Figure 2.9. The
transition phase in this process discusses employee transition and computer application migration. A subset of the employees from the client organization is transitioned to the vendor organization. After transition, they work as employees on the vendor site. The applications developed on the client side are migrated to the vendor. After transition, the vendor is responsible for managing those applications.

The third model is suggested by E. A. Sparrow [40]. It identifies activities required for implementing transition as a part of offshore outsourcing. These are listed in Table 2.5.

The fourth publication, written by M. J. Desouza and K. C. Bonifazi, discusses transition issues as part of the outsourcing process [43, p. 142]. It points out seven different issues with project transition during outsourcing to a vendor. These are the following:

- **Developing a transition plan**: Transition plan should discuss all the issues that are significant for a successful transition. The transition plan should be communicated properly with all the stakeholders.

- **Training and knowledge transfer**: Training and knowledge transfer is concerned with the transfer of knowledge to the vendor organization. The customer should share enough knowledge so that the vendor can understand the requirements and build the system. Too little knowledge will result in misunderstandings about the functionality of the system. Too much knowledge, on the other hand, may result in exposure of confidential information to the vendor organization [43, p. 143]. The customer and vendor organizations must
Figure 2.9. Lever outsourcing stages [42]

establish a combined team for knowledge sharing. An informal information sharing mechanism must be in place such as, for instance, some form of discussion groups or documentation repository.

- **Communication protocols**: Both parties may belong to different countries and cultures and may follow different processes. Hence, it is very important to agree upon the terminology used and its meaning during the communication [43]. A communication plan should be developed to share timely accurate information about the status of the outsourcing project [29].

- **Issue resolution**: The issues arising between customer and vendor should be resolved at the beginning of transition. One example is the disagreement on the process of communication to inform about status updates from the vendor side. If not resolved immediately, it becomes very costly to change the process in the later stages.

- **Intellectual property protection and security provision**: The vendor should have access only to the knowledge and assets that are required to perform its tasks. The customer can minimize the system access by security mechanisms such as password required for accessing the system from the vendor site. The customer can ensure that, for example, only 20 passwords will be provided to the vendor so that no more than 20 staff members can access the system from the vendor site. [43, p. 147]

- **Contingency processes**: The organization must have contingency plans if the transition becomes unsuccessful. For example, it may happen that after transition, the system installed at the vendor site becomes overloaded due to heavy traffic of transactions. [43, p. 148].

- **Standards and integration issues**: Standard and integration issues are concerned with the integration of part of system that was developed by the vendor with the system that is already operational at the client site.

<table>
<thead>
<tr>
<th>Table 2.5. Transition activities for offshore transition [40]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activities</td>
</tr>
<tr>
<td>Detailed project planning</td>
</tr>
<tr>
<td>Transfer of responsibilities to the offshore project</td>
</tr>
<tr>
<td>Implementation of contract management process including invoicing and payment procedures</td>
</tr>
<tr>
<td>Introduction of change request processes</td>
</tr>
<tr>
<td>Implementation of problem management and escalation procedures</td>
</tr>
<tr>
<td>Risk mitigation activities</td>
</tr>
<tr>
<td>Communication program</td>
</tr>
<tr>
<td>Career development help and support for those staff whose work is moving overseas</td>
</tr>
<tr>
<td>Liaison with trade unions or other employee representatives</td>
</tr>
</tbody>
</table>
2.4.4 Discussion

The core publications dealing with transition were written almost two decades old. Thomas Vollman was the pioneer in the handover domain and his work was then extended by Thomas Pigoski. The publications discuss transition at a general level. They however provide an important basis for our research.

The transition description provided in IEC/ISO 14764 and IEC/ISO 15288 strongly overlaps with the work of Thomas Pigoski. Hence, we could not gain any additional knowledge regarding transition in these standards.

The focus of ITIL Service Transition is different from our research on handover. ITIL Service Transition focuses on defining activities and best practices for deploying service releases in the production environments successfully. It is mainly concerned with decommissioning of an older service and deployment and operation of the new service.

When studying the outsourcing literature, we realized that the outsourcing research is more concentrated on what, why and when to outsource. Less attention has been paid on the how question, that is, how to outsource. Additionally, the how part focuses more on the vendor selection, negotiation and contract writing. Less information is available on the issues related to pure system handover. The outsourcing process models discuss transition as part of outsourcing implementation. They discuss it at a very general level. They however point out that training, documentation; knowledge transfer and communication are the main challenges in the outsourcing implementation.
3. Research Process

In this chapter, we describe our research work. We first give an overview of our research phases in Section 3.1 and then provide their detailed descriptions in Section 3.2. We continue with presenting the experiences gained during this research in Section 3.3.

Just because our research focused on generating theory in the handover process domain, we chose the qualitative study approach. We describe, motivate its choice and present the research tools in Sections 3.4 and 3.5. Finally, we discuss validity threats in Section 3.6 and present the bodies involved in Section 3.7.

3.1 Overview of Research Phases

Our research process consisted of five main phases where each phase had a clearly defined goal. The phases are illustrated Figure 3.1 and they are (1) Choice of PhD Topic, (2) Initial Study, (3) Small-Scale Exploration, (4) Large-Scale Exploration, and (5) Final Study.

In the first phase, the Choice of PhD Topic phase, we chose a research topic. Our goal was to choose a topic that would fulfill the needs of the Pakistani software industry. The choice ended in a decision of creating a model of a handover process.

After having chosen the research topic, we decided to investigate the topic domain. Our goal was to find out what had been achieved within the handover domain and to create a basis for further research. Therefore, in the Initial Study phase, we studied the current state of art, identified the scope of the handover process and its strongly related pre-delivery maintenance process, elicited handover problems, and created a preliminary version of the taxonomy handover activities. The Initial Study phase was conducted via literature and case studies within the industry.

Our next step was to evaluate the preliminary taxonomy in the industry. We did it in the Small-scale Exploration phase during which we confirmed the relevance of the taxonomy, acquired feedback for improving it, elicited roles involved in the process, and their placement within software lifecycle. At this phase, we also realized how comprehensive and complex the domain was.

In the next phase, the Large-Scale Exploration phase, we studied the handover process within a large number of companies. Our goal was to explore as much information about the handover process as possible. The results of this phase helped us to further improve the taxonomy of handover activities and identify issues that provided input to create EM³: Handover Framework. It also made us realize that the handover process was closely related to the outsourcing domain. In order not to omit anything, we decided to conduct a new round of literature study, this time however, with a specific focus on the domains that were closely related to the handover domain. As a last phase in our research, in the Final Study phase, we evaluated the handover framework via participatory research in an industrial setting in one Swedish company.
3.2 Research Phases

In this section, we present our research phases in greater detail. Our descriptions follow the order of the research phases illustrated in Figure 3.1.

3.2.1 Choice of PhD Topic

Our research was supported by Higher Education Commission of Pakistan [44]. The goal was to create a critical mass of highly qualified manpower in high-tech fields to build up the capacity of universities, research & development organizations and industries in Pakistan through addition of PhDs in priority fields. For this reason, it felt natural to choose a topic that was attractive to the Pakistani industry and served its primary needs. To find appropriate Pakistani companies that could support our research, we consulted Pakistan Software Export Board [45]. They advised us to contact one Pakistani company in Lahore to which we refer to as Main Company in this thesis\(^1\). Main Company is a large multi-national company providing development, evolution and maintenance services to various

\(^1\) The company’s name is fictitious.
organizations worldwide. Main Company was on a high maturity level according to the CMM scale\(^2\); hence, it constituted an excellent industrial platform for doing our research.

Together with our supervisor, we arranged a workshop with Main Company’s management and developers during which our supervisor first presented her research within evolution and maintenance and then raised various industrial problems and challenges. Afterwards, the company’s problems were identified and discussed. Out of them, handing over software systems stood out as the most intricate and unsettled process. Although the company was on a high CMM maturity level, it had difficulties in managing their handover processes. Therefore, the choice of the research topic became obvious.

Due to the fact that we conducted our PhD studies in Sweden and the company was situated in Pakistan, we initially decided to interact with them via communication channels such as telephone, email and Skype meetings, and once-a-year face-to-face meetings. With time, however, we realized that this communication method was not optimal; not because of the communication channels per se but because of strong hierarchical decision making in the company and lack of support from their respective contact persons. Therefore, after a year of our study, we decided not to utilize their support. We felt that we were losing valuable time. We still continued our research on the handover subject; this time, however, using support of other companies to be presented in the next research phase descriptions and in the next sections of this chapter.

### 3.2.2 Initial Study

The Initial Study phase consisted of four steps. These are Literature Study - First Round, Scope Study, Elicitation of Handover Problems and Creation of Handover Taxonomy.

#### 3.2.2.1 Literature Survey, First Round

During the Literature Study - First Round step, we searched for publications dealing with software handover. Our goal was to establish the state of art within the domain. Here, we used three major media such as (1) research articles (2) books and standards, and (3) online publications. In all these media, we used keywords such as pre-delivery maintenance, transition planning, maintenance planning, handover, software handover, transition, software transition, transfer of responsibilities, and transition from development to maintenance.

Regarding the research articles, we searched in renowned scientific databases including IEEE Explorer, ACM, and Springer. Moreover, we used KTH library searching tool Primo, for extracting search results from multiple databases. When looking for books and standards, we used the KTH tool Libris, the tool that did not only search in KTH library but also in all university libraries in the whole Sweden. We also used Google search engine and Google scholar for finding relevant books across the globe.

Despite the enormous effort of searching for appropriate literature, we did not manage to find many publications. We only found eight publications dealing with software handover.

---

\(^2\) On purpose we do not expose the CMM level of the company studied.
These publications were one article written by Thomas Vollman focusing on the management and technical aspects of a transition process [17], three book chapters written by only Thomas Pigoski who continued with Vollman's work by describing his experiences and lessons learned when transitioning systems at the US Department of Defense in USA [9], two papers [23] [22], an MIL-HDBK 347 standard written by the US Department of Defense [19], two industrial standards discussing transition at a general level [12] [13] and finally, a book chapter written by Alan April who used the Vollman and Pigoski's process descriptions and put them into a model [11]. It is important for us to mention that the contents of all these publications strongly overlapped.

3.2.2.2 Scope Study

The results from the literature study showed that despite the significance of software handover, little research had been conducted in this area. We managed to find a handful of research articles dealing with the subject. Therefore, we realized that our work would have to be based on exploring the domain within the industry only.

The authors of the few publications clearly focused on transitioning from development to maintenance. They placed a dividing line distinguishing development and maintenance on a border between the first system version and the remaining versions. This implies that by development, they meant a creation of the initial system version and by maintenance, they meant a creation of its consecutive versions. Hence, their understanding of the transition scope was clear.

When studying various other publications, publications related to software maintenance, we found out that maintenance was differently understood by the software community and that the dividing line between development and maintenance was not always clear-cut. This had made us somewhat confused. Therefore, in the Scope Study phase, we decided to explore and demarcate the scope of the handover process. The study was done by interviewing 18 companies using the questionnaires presented in Appendix B.1. Our goal was to find out how the industry understood the process and how it placed it within software lifecycle. We felt that identifying the scope was a prerequisite step for commencing our research on the handover process.

In a nutshell, the interviews had helped us identify seven different scopes of the handover process. They had helped us extend the scope as defined in the literature studied. The interviews had also made us realize that handover process strongly overlapped with development, pre-delivery and post-deliver maintenance. It was not always clear what was the actual difference between those processes. This phase had resulted in one publication [46] to be presented as Paper 3 in Part III of this thesis.
3.2.2.3 Creation of Handover Taxonomy and Elicitation of Handover Problems

The literature studied revealed that the domain studied was not sufficiently explored. It was only described as a free general text in [17] and its problems were not clearly identified. Therefore, in the next phase, the Creation of Handover Taxonomy phase, we created an initial version of the taxonomy of handover activities. This taxonomy is called EM³: Handover Taxonomy.

When creating the taxonomy, we extracted handover activities from the available literature and clustered them into process components which we believed were coherent and logically consistent. Our goal was to define activities in as generic way as possible so that they might be useful in different handover contexts. The publications providing the platform for creating the taxonomy were [17] [9] [19].

Because most of the publications were published almost two decades ago, they contained activities with outdated terminology. Therefore, before putting their activities into our taxonomy, we had to rephrase many of them. Altogether, we arrived at seven taxonomy components, which we later call practice components. As can be seen in Figure 3.2, the initial set of taxonomy components were Management and Administration, Maintenance Environment, Version and Configuration Management, Training, Deployment, Documentation and Maintainability Management. Regarding the handover problems, we elicited them by interviewing five companies using the questionnaire presented in Appendix B.2.

This phase had resulted in two publications, one dealing with handover problems [47] and the other one suggesting a preliminary taxonomy of handover activities [48]. The preliminary version of the taxonomy is presented in Appendix A.1 and the publications presenting the problems and the taxonomy are presented in Part III of the thesis as Paper 1 and Paper 2, respectively.

![Figure 3.2 Taxonomy of handover activities](image-url)
3.2.3 Small-Scale Exploration

Now, when we had an initial version of EM3: Handover Taxonomy, it was high time we evaluated it within the industry. We did it in the Small-Scale Exploration phase during which we evaluated the taxonomy within three companies. The evaluations were conducted in form of interviews using the questionnaires presented in Appendices B.3 and B.4. The goal was to explore the domain and confirm that the activities in the taxonomy were relevant and to explore knowledge about other EM3: Handover Framework parts, such as roles involved, placement of the activities in software lifecycle and the like. Altogether, this phase consisted of four industrial case studies, where two of them were conducted in one and the same company.

The evaluation of the handover taxonomy was conducted within three companies via three case studies using the same questionnaire (see Appendix B.3. The companies were AMIS Software, and two Pakistani companies, CRM Telecom and VAST Telecom. In the Swedish company, AMIS Software, we focused on an in-house handover process that might or might not involve COTS suppliers. In the first Pakistani company, the CRM Telecom Pakistan, we studied an in-house handover where the system was transitioned from a development team to a separate maintenance team. Here, we interviewed IT Development Manager. In the second Pakistani company, VAST Pakistan, we focused on the handover of a software system from a vendor conducting system development, evolution and maintenance to an acquirer organization using the end product and conducting front-end support. Here, we interviewed Business Support System Executive.

In addition to the three above-listed case studies evaluating the handover taxonomy, we had the opportunity to follow a handover process in VAST Pakistan. Here, we walked the handover process as if it were implemented in the company. When walking the process, we inquired about its main properties by using the questionnaire presented in Appendix B.4. Our goal was to gain understanding of the handover activities in a real-life case scenario.

During all the interviews, we realized that the scope and magnitude of the handover process were huge. Therefore, the effort and time required for making the interviews was immense. To be able to grasp the whole domain, we conducted our interviews in a series of sessions. This was because (1) the large scope of the process was not possible to discuss in only one session, (2) we needed time for maturing with the information acquired during the interviews, and (3) we needed extra sessions to repeat some of the questions in order to clarify the ambiguous and incomplete answers provided in the previous interview sessions. The study had resulted in four publications that are presented in Part III as Paper 4, Paper 5, Paper 6 and Paper 7 [49] [50] [51] [52].

3 The names of the companies are fictitious
3.2.4 Large-Scale Exploration

The small-scale evaluation made us discover that our taxonomy activities were relevant; however, their performance strongly varied depending on the type of handover, the type of the system to be handed over, its criticality and complexity, number of parties involved, the organization’s structure, and the time span during which the handover was performed. The wide time span of its activities indicated strong process variability and parallelism with many other processes. It needed to be expanded with more activities to be relevant in various industrial contexts. Therefore, we strongly wished to explore the domain on a larger scale in various organizational scenarios and contexts.

Our time constraints and resources did not allow us to do a large-scale exploration on our own. It was difficult to get access to the right individuals possessing knowledge about the handover process domain. Only finding such individuals would take time and interviewing them would take much time as well. For this reason, we used the opportunity to delegate the interviews to students studying software engineering at a post-graduate level.

As mentioned in the Small Scale Exploration phase, the interviewing on the taxonomy was very burdensome and time-consuming. Many times, it required a series of interviewing sessions. When doing the large-scale exploration, we simply could not delegate the study of the whole taxonomy to the students. We felt that this might jeopardize the quality of the collected data. For this reason, we divided the seven taxonomy components into two parts that were manageable and conductible within one interview. We also decided that the students would use the same questionnaire as the one that was used during the Small Scale Exploration step (see Appendix B.3).

The two parts of the large scale study are presented in Figure 3.3. As can be seen there, the first one consisted of Management and Administration, Maintenance Environment, Version and Configuration Management and Documentation components, and the second one consisted of Training, Deployment, Maintainability Management and Documentation components. The astute reader may immediately see that the Documentation component was a constituent of the two...
parts. We did so because it was difficult to divide the taxonomy into equal parts. In order to balance the study scope and even out the interviewing burden, we included this component in two parts. The taxonomy the students used for interviews is available in Appendix A.2.

The students were free to choose one of the two parts. As illustrated in Figure 3.3, the majority of the students chose the first part leading to the collection of data from 29 organizations. The remaining students provided data on the second part from 20 organizations. We organized the data collected by students in templates for each activity. A snapshot of data gathering template arranged organization-wise is available in Appendix C.3.

The results of the Large-Scale Exploration steps showed that the activities were relevant. They helped us discover new activities and they provided us with feedback leading to the improvement and expansion of the existing activities available in Appendix A.3. This step had resulted in six papers that are presented in Part III as Paper 8, Paper 9, Paper 10, Paper 11, Paper 12 and Paper 13 [53] [54] [55] [56] [57] [58].

3.2.5 Final Study

The results of the Large-Scale Exploration phase helped us to further improve the taxonomy of the handover activities and the framework components. It also made us realize that the handover process domain resembled software outsourcing domain. Therefore, in the Final Study phase, we decided to explore this domain via a next-round of literature study which we call the Literature Study – Second Round phase. Our goal was to find material that might be suitable for enhancing our taxonomy. Finally, we evaluated whether the taxonomy served its purpose within the industry by conducting participatory observation in the Acting on Handover Taxonomy phase.

3.2.5.1 Literature Study, Second Round

When conducting the second round of the literature study, we searched in the same literature sources; however, we expanded our keyword portfolio with new words such as IT outsourcing, offshore outsourcing, transition in outsourcing, and knowledge transfer in outsourcing. We succeeded to find 25 additional publications related to the handover domain and we came upon an important standard which we missed in the first round of our literature study [18]. The related publications dealt with issues related to software outsourcing. The standard was the ITIL standard and it dealt with transition of changed or new IT service to operation.

To remedy the problem of not including the standard in our initial study, we studied it very carefully in order to find similarities and differences between our work and the standard. We soon found out that the title of the standard was related to our domain, but the focus of the ITIL standard was different. The standard focused on transition of a new service from development to operation. We found the standard to be very comprehensive and focusless. The standard provided very detailed descriptions of a set of processes that were important for decommissioning the old service and replacing it with new service, but it did not identify the activities of these processes that were critical for performing and succeeding with the handover process.
Regarding the related literature, we found four publications discussing transition as part of outsourcing process [40] [41] [42] [43]. However, they discussed transition at a general level. We could not discover any new activities which we could use in our taxonomy. It had however helped us to confirm the already existing activities in our taxonomy. Moreover, we realized that some of outsourcing issues resemble handover issues.

Finally, at the very end of our thesis work, we found a licentiate thesis published on a related topic that was defended in 2011 [59]. What is worthwhile to mention is the fact that the thesis used our research results as a base for defining how software organizations might transition their systems from project organizations to maintenance ones. Our research results utilized by this study were published in [49] [46]. Although the thesis was in Swedish, we google translated it, read it and compared it with our results.

3.2.5.2 Acting on Handover Taxonomy

As a last step in our research presented in this thesis, we evaluated the handover taxonomy via participatory research in an industrial setting. The goal was to implement the taxonomy in industry, and thereby, to evaluate its realizability and usefulness. We got an opportunity to follow and act upon a handover process within one Swedish organization, which was in the process of handing over a software system from a team that was dissolved to a new team that would be created in the future to come.

We followed the participatory research approach during the study where we played the role of active participant [60]. This means that through participating in the process, we tried to understand the handover process studied, we actively observed what did happen and what did not happen, and provided support to the company while implementing the EM³ practices. In this way, we gained a close familiarity with the process and the people performing the process, we gained understanding of the complexity and vulnerability of the process, and finally, we gained confirmation that the EM³ handover practices were fully applicable in an industrial handover context.

The handover process studied took three weeks to perform and the duration of our participatory observation research corresponded to the duration of the handover process, that is, eight hours a day during three consecutive weeks. In this way, we were able to follow the process from its very beginning to its very end. During this time, we conducted four major steps that are typical of a participant observation method [60]. 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 members 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 it developed and the processes it used for managing the system. To be able to obtain quality data during our research phases to come, it was important for us to become part of the company’s team. For this reason, we spent the remaining time of the first day on becoming friends with the company’s employees. We strongly believed that establishing good
relationship would contribute to become accepted as a team member in the company studied.

In the *Acting in the Field* phase, we tried to act just as the company’s “local” member with some minor exceptions [60]. The exceptions were that in addition to starting 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 with informal discussions with the company’s employees.

As an active team member, or as a local, we actively worked on implementing the handover process using *EM³*: *Handover Framework*. Being fully integrated with the transition team of the company studied, 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, and supporting the company’s management in decision making.

In addition to the above listed activities, we participated in brainstorming sessions with the development team lead. The goal of the brainstorming sessions was twofold. First, we obtained detailed knowledge about the process area and the team leader got the opportunity to refresh his knowledge. Second, it helped us to develop documents in a structured way to provide detailed knowledge for the transitionee team.

The third phase, the *Recording Observations* phase ran in parallel with the *Acting in the Field* phase. Here, while observing the process we could follow the implementation of almost all of the *EM³* practices, compare their activities with the company’s handover activities and evaluate their applicability. Wherever it was relevant, we suggested improvements in the light of *EM³ Framework*. This helped the company in covering the gaps in their handover process and this helped us gain important feedback for improving the *EM³: Handover Framework*.

Regarding the activities that could not be implemented in the process, we inquired about their usefulness in the other handover contexts using semi-structured interviews based on questionnaire available in Appendix B.3. We also conducted semi-structured interviews with the purpose of finding reasons behind the handover process studied, its main challenges, and the action that could be taken to remedy the challenges.

Finally, in the *Analyzing Data* phase, we studied each of the *EM³* activities with the purpose of finding out whether it was fully or partially implemented and with the purpose of finding reasons for their non-adherence to the executed handover process.

### 3.3 Experiences

Our journey towards handover framework was not a very smooth one. We faced many difficulties during our research process. These are the following:
1. **Difficulties to find companies**: One of the main hurdles we faced was to find and convince companies for participating in our research. For instance, it took us one month to contact Main Company in Pakistan and to manage to convince them. A similar hurdle applied to the Swedish companies as well.

2. **Difficulties to cooperate with the companies**: To find companies was only the first difficulty to overcome. The next step was to cooperate with them. In our initial research phases, we encountered difficulties to cooperate with the Pakistani company at a long distance. We observed that the company did not provide fully dedicated resources for supporting our research. They became extra careful for sharing their data. All their answers should always be approved by lawyers and via a chain of managers from the lowest to the highest levels. As a result, it took sometimes days to get the desired information. This resulted in a substantial delay of our initial research study results.

3. **Difficulties to find right professionals**: As mentioned earlier, it was difficult to find the right professionals having rich experience of a handover process. This is because the handover process is very comprehensive and broad. It covers almost all the development lifecycle phases and initial maintenance phases. To find individuals having this broad competence proved to be immensely difficult. There are a few people who have a broad knowledge of handover domain and they are normally working at key positions. Despite this, we succeeded to find few individuals working for AMIS and CRM organizations.

4. **Limited access to professionals**: It was always difficult to keep a continuous contact with the professionals involved in our study. The studies we conducted were huge in their nature and it was not always possible to conduct the interviews within one meeting. To find time for a series of meetings demanded great sacrifices from both our and our interviewees’ sides. Also, we sometimes had to contact the professionals that were interviewed by our students in order to dissolve various ambiguities that aroused during data analysis. This resulted in much time spent on chasing the individuals and finding time for common discussions.

### 3.4 Research Approach

In this section, we describe the research approach. First, in Section 3.4.1, we compare the qualitative and quantitative research approaches. Then, in Section 3.4.2, we motivate the choice of our approach.

#### 3.4.1 Quantitative versus Quantitative Research

There are two major research approaches, *quantitative* and *qualitative*. Quantitative approach is objective and heavily relies on statistics and figures. It focuses on investigating things via statistical, mathematical or computational techniques. It is mainly used for studying narrow questions by analyzing data, producing statistics and yielding results that can be generalized. Quantitative methods focus attention on measurements and amounts of the characteristics displayed by the people and the events the researcher studies. [61, p. 1] According to [62, p. 9], a major disadvantage of quantitative research is the fact that it tradeoffs understanding of the contexts of the study for the ability to generalize them across the populations.
studied. Quantitative research focuses on methods like laboratory experiments, simulations, mathematical modeling, statistical analysis and the like [62, p. 8].

The qualitative approach, on the other hand, focuses on investigating things without comparing them in terms of measurements and amounts. In other words, it focuses on exploring issues, understanding phenomena and answering various contextual questions. Instead of using statistical and mathematical techniques, it continuously collects and analyzes data until it reaches the point of data saturation, that is, the point when no more information may be found on the domain studied.

The aim of the qualitative approach is to study a specific domain, and to gain an in-depth understanding of its context and reasons governing the domain of the context studied. Its key benefit is that it allows researchers to see and understand the context within which decisions and actions take place. Qualitative research includes methods like participatory research, case study research, ethnography, grounded theory and like. [62, p. 8]

The major disadvantage of qualitative method research is that it is often difficult to generalize beyond the sample studied to a large population [62, p. 9]. This is because qualitative research is rich in descriptions and various motivations and explanations. It does not approach any specific hypotheses to be proved or questions to be answered [63]. It is rather concerned with understanding the domain from the frame of reference of the subjects involved in the study. Here, the researchers focus on investigating the whats, whys, whos, hows, wheres and whens. Instead of testing theories via statistical methods using a randomized sample, the researchers inductively analyze their data and develop theories using mainly purposive sampling of their subjects, the subjects that are not chosen by chance but by their capability to contribute to the research results. Therefore, the samples are often small and the conclusions are only propositions. For this reason, the quality of the qualitative research is not determined in terms of validity but in terms of credibility, transferability, dependability and confirmability to be described in Section 3.6.

3.4.2 Why qualitative research approach?

It was only via qualitative research such as case studies and participatory research that we could explore knowledge about the handover domain. Our main purpose with conducting research was to generate theory in the handover process domain, that is, to create knowledge and understanding of the handover domain, and to build theory that was useful for creating handover process models. Our theory is the handover framework corresponding to a set of activities, roles, lifecycle roadmap, handover types, contexts, and guidelines that explain the phenomena of a handover process domain.

While generating the theory, we explored (1) what activities were critical for performing handover, (2) what roles are involved in the handover process, (3) where in the lifecycle process the handover activities take place, (4) what types of handover do we have, (5) in what contexts does handover process apply and (6) what are the suggestions to implement the activities. We then evaluated our theory’s usefulness and appropriateness in real-life industrial handover contexts, and created a basis for defining a handover process model.
In our case, we were dealing with the domain that was not well investigated beforehand. There was not much information available in literature and the only option that was left was to explore the handover by communicating with experienced people both at organizational and individual levels. Therefore, in our case, quantitative research methods would not be helpful at all. We had nothing to test the validity of. What we had was the out-aged handover theory that was two decades old and awareness that the domain of the handover domain was strongly unexplored. For this reason, the choice of the qualitative research method was self-obvious and self-explanatory.

3.4.3 Types of Studies

In our research, we mainly conducted two types of studies: case studies and participatory research. As can be seen in Figure 3.1, the majority of our research was covered by the case studies. Only the last study was conducted via participatory research. During all our phases, we evolved the handover framework. After the participatory research, we understood that our framework reached the saturation point. We simply could confirm that our framework was realistic in an industrial handover setting and we could not add anything new to it.

Regarding the first type of study, the case study, we judged it to be an appropriate approach for exploring the domain and for finding empirical evidence that was necessary for evaluating our framework. We used organizations as fora for conducting our case studies, where we focused on what, how, why and when questions using multiple sources of evidence such as interview results and organizational documents. This helped us explore the handover taxonomy in detail, and thereby, augment it with new activities.

Via case studies, we could explore all the events and features that were inherent and relevant in the handover process and understand the contexts in which they took place [64]. By conducting several studies, we could collect a substantial amount of empirical evidence from the industry, and thereby, to contribute to knowledge building. We built the knowledge by confirming that the taxonomy was useful in various industrial contexts and by providing contextual and casual explanations of the taxonomy activities.

Most of our cases were of descriptive character. Often, they were presented in a narrative form based on real-life situations. With this, we were able to study the process, the activities taking place within the process, find reasons for why they were taken, and find out how and when they were implemented and with what results [65, p. 12].

Regarding the choice of the second type of our study, the participatory research type, we conducted it in the Final Study stage of our research. We found this approach useful for experiencing and acting upon a handover process in a real life scenario. The company under the study had neither any handover process model in place nor any guidelines for how to conduct handover. Therefore, our framework was the main driver of the process which strongly contributed to its implementation. Being part of the team, we did not only observe its handover process, we also acted upon it, and we took the opportunity to evaluate every single activity of our taxonomy.
During our participatory research phase, we collected data via (1) observing the employees during work and meetings, (2) informal communication, and (3) active participation. Observations helped us understand the working of the organization. Informal communication helped us understand the reasons behind handover, reasons behind conducting handover activities and the difficulties the organization faced when implementing the handover process. Via active participation, we became an active team member. Here, we did not only contribute by executing the handover process but also by sharing our taxonomy and our experiences gained during our former research phases.

### 3.4.4 Sampling

When selecting our subjects of study (organizations and professionals), we used the nonprobability sampling method which is called *purposive or judgmental sampling* [66]. This was the only sampling method that was appropriate for us to use. We did not enjoy the luxury of having a population of software organizations from which we might randomize a sample of the organizations to be studied. Neither was it possible for us to estimate the population size and randomize its sample. Even if it were possible, we would still face the problems of getting denials from the companies to participate in our study or getting companies that did not have any knowledge and experience in the handover process.

It is important to put substantial effort into choosing the subjects of study. We chose our samples by judging whether the companies were appropriate for our study and whether they had the capability to contribute to our research results. In our opinion, *purposive sampling* was the only method to choose when there was a limited number of subjects who had the expertise in the area being researched on. In all the sampling cases, we always put substantial effort into confirming that the roles involved in our studies had the right expertise and experience.

### 3.5 Research Tools

We used interviews as a data collection tool. A major part of our study consisted of semi-structured interviews used during all our case studies. The questionnaires used are in Appendices B.1 – B.4. By studying them, the reader may easily see that (1) via the questionnaire presented in Appendix B.1, we explored the scope of the handover process, (2) via the questionnaire presented in Appendix B.2, we elicited problems in handover, and (3) via the questionnaires presented in Appendix B.3 and Appendix B.4, we explored the handover process. We even used the questionnaire in Appendix B.3 during informal discussions during our participatory research. We used it for exploring the activities that could not be performed during the instance of the handover process that we actively participated in. Almost all the interviews were voice recorded and transcribed. Those few that could not be recorded were thoroughly documented instead.

We and the students who were commissioned to conduct the study used two ways of conducting interviews. These were (1) face-to-face interviews and (2) telephone interviews. Our first preference was always to conduct face-to-face interviews. However, face-to-face interviewing was not always possible due to different geographical locations or other
commitments of the interviewees. In such situations, we or our students used telephone or Skype interviews instead.

3.6 Validity Threats

Validity evaluates the strength and soundness of the statement and the appropriateness of a research method; whether a method used investigates what it purports to investigate [67, p. 327]. There are certain tests which are used to establish the quality of empirical research. These are internal validity, external validity, dependability, confirmability, and construct validity [65, p. 34]. These tests are mainly accepted within quantitative research studies.

Validity of our qualitative research results corresponds to the strength and correctness of how our framework mirrors the process parts that are included in real world handover processes [67, p. 327]. In the qualitative research context, they should be judged with the following qualitative criteria: [68]

- Credibility corresponding to internal validity.
- Transferability corresponding to external validity.
- Reliability corresponding to dependability.
- Objectivity corresponding to confirmability.

Below, we describe the validity criteria using the terminology from both qualitative and quantitative research wherever relevant and appropriate and motivate how we have fulfilled them.

3.6.1 Internal validity – Credibility

The internal validity is only a concern for explanatory and causal studies where the researcher determines whether event \( x \) led to event \( y \) [65, p. 36]. It is not applicable in the context of exploratory and descriptive studies due to the fact that such studies do not make any causal claims. However, in our context, it may be translated to the qualitative criteria – credibility [68] [69].

The credibility establishes the confidence in the truth of the findings from the subject’s (respondents) perspective [69]. The purpose is to establish whether the researchers may judge whether the research results are credible.

In our case, we mitigated the credibility threat via four steps. First, we attempted to base our research on the former published studies [9] [17] [22] [19]. Although these studies were old and scarce in their number, they were still developed in serious and complex organizations such as, for instance, US Department of Defense. Second, in Small-Scale Exploration phase, we used member check technique for assuring credibility [69]. That is, we shared the results with the interviewees for checking before casting them in the final form. Third, in the Large-Scale Exploration phase, we used triangulation technique [69]. That is, we used multiple source of information, and hence, explored our taxonomy in many diverse industrial contexts. Forth, in the Final Study phase, we eventually observed and executed a
handover process using our taxonomy in the real world scenario. Thereby, we confirmed the usefulness and appropriateness of the taxonomy activities.

### 3.6.2 External validity – Transferability

External validity deals with the problem of whether a study’s findings are generalizable beyond the immediate study [65, p. 37]. In qualitative research, it corresponds to transferability of the results from one context or setting to another [68] [70]. A common complaint about qualitative research is that it is difficult to generalize [65, p. 37]. We are aware of the fact that it is difficult to declare qualitative research findings as universally generalizable. A theory to be tested must replicate the findings in a second or even a third case. Once the researcher made such replications, the results might be accepted as providing strong support for the theory, even though further replications have not yet been performed. [65, p. 37]

In our research, we replicated the majority of our results with the help of the case studies and participatory research study. To enhance the transferability of our results, we studied and provided detailed descriptions of the research contexts. We found out that our handover framework was relevant in all our studies and that, when adapted to a specific context, it was easily transferable among different industrial settings. The only finding that we could not transfer among the studies was the placement of the activities on the software lifecycle phases and the order among the handover activities. This can be easily explained by the intricate, complex and diverse nature of the software processes. We solved this issue by suggesting that the handover activities be placed within activity spaces to be described in Chapter 5. The individuals wishing to transfer our results to their contexts have to make their own judgments of how much of our taxonomy is transferable.

To further strengthen the transferability of our results, we explored the taxonomy in 61 organizations. The organizations were of different sizes, they operated in different business domains and they were located at distinct geographical locations. During our research, we found that our taxonomy was applicable and relevant in all the organizations studied with certain variations with respect to the type of handover and the nature of the system to be handed over.

Although we cannot claim that the results of our study are generalizable, still, due to the large and diverse nature of our sample, the repeated studies and participatory research, we are confident that our results appropriately mirror the current state of the handover practice within the organizations studied.

### 3.6.3 Reliability – Dependability

Reliability is the consistency or repeatability of measures [68] [71]. It determines whether the findings of an inquiry would be consistently repeated if the inquiries are replicated with the same or similar respondents [69]. The objective is to reach the same conclusions by repeating the study and by following the same procedures as followed in the study under consideration. [65, p. 37]
We did not enjoy the luxury of conducting the same case study twice in the same organization at different points in time due to reasons such as time constraints and limited availability of our contact professionals. Even if we had this luxury, it would not be certain that the process criteria would stay the same in the organizations studied.

The dependability was not a direct threat to us. While conducting our studies, we continuously accounted for the organizational contexts, and if deemed appropriate, we extended our taxonomy using the feedback from those contexts [46] [49] [50] [51]. All the changes to the taxonomy due to the dependability criteria are clearly identified and reported in the thesis results.

### 3.6.4 Objectivity – Confirmability

Objectivity means that the researcher should not allow his own biases to affect the study [70]. The purpose is to assure the neutrality of the research. To remedy the threat of confirmability, we repeated our studies and reused our questionnaires in all our study steps and in all the 61 companies studied. We whole-heartedly reported the negative instances with proper reasoning. Moreover, we used open-ended questionnaires so that we might be able to interpret the taxonomy activities in the contexts studied and judge whether they were appropriate enough for being put into our taxonomy. This helped us remove the bias that might be accounted to a specific context or organization.

### 3.6.5 Construct Validity

*Construct validity* is assessment that whether an inquiry is measuring the construct it claimed to be measuring [65, p. 37]. In our research, we were interested in evaluating and exploring the activities that were critical for performing a handover process. We did this via mainly interviews and participatory research. The biggest threat here was that we might end up conducting interviews with the wrong individuals, the individuals that might not have any knowledge of a handover process or they might differently understand it. To overcome this threat, we first made sure that the interviewees had a solid knowledge and experience of the handover process. We also assured that their scope of understanding agreed with ours. Only then went we ahead with conducting interviews.

Regarding the large scale exploration, the risk here was that the students might misinterpret the handover process and its results and that they might choose the wrong interviewees. To minimize this threat, we prepared students for conducting interviews. First, we requested that the students read about handover. We then gave one two-hour lecture on handover during which we assured that students had understood the domain studied. We also presented the questionnaire to the students, the questionnaire presented in Appendix B.3. On top of that, we provided counseling hours to the students to assure that they correctly understood both the handover domain and the questionnaire.

To ensure that all the questions were answered for each handover activity, we created templates to be filled in by the students during interviews. An excerpt of such a template is provided in Appendix C.1. As can be seen there, we listed each question for each activity
and arranged space to be filled in with the answers. We also requested that the students recorded and transcribed the interview results. An example of the answers provided by one student is provided in Appendix C.2. In this way, we ensured the uniformity of the data provided by the students and completeness of their answers.

Finally, to validate that the right individuals were interviewed, we requested that the students followed the same procedures as we did, that is, that they made sure that the right individuals were chosen for the interviews and that their interviewees had the same understanding of the handover scope as the students had. To be able to verify this, we also requested that the student provided contact information about their interviewees.

3.7 Bodies Involved

Altogether, sixty one organizations were involved in our study during the whole research process. They are all listed in Appendix D. As can be seen there, the organizations were of different sizes (large, medium and small), and the roles involved included CEO, CTO, project manager, system architect, consultants, head operations, integration manager, development manager, development team lead and the like. The organizations involved operated in different domains including telecom sector, financial systems, enterprise resource planning (ERP) systems, aviation industry, and more. They were located at 16 different countries. Sweden had the maximum participation of 28 organizations.

Every step of our research implied interaction with professionals working in IT industry. Either we or our students were involved in interacting with those companies. Usually, one company was involved in one specific research method phase. It is important to mention however that some of the companies were involved in more than one phase. The last column in the table available in Appendix D lists the names of the phases in which each respective company was involved in. Some organizations participated in more than one research phases. Due to sensitivity issues, we neither disclose the name of the organizations nor the names of the interviewees.
4. Research Contribution

In this chapter, we summarize our research contribution in the handover domain. Our research work resulted in fourteen research publications which we group into (1) Initial Study publications, (2) Small-Scale Exploration publications, (3) Large-Scale Exploration publications and (4) Final Study publication. These groups reflect the phases of our research work. Below, we present the summaries of our publications.

4.1 Initial Study publications

The first group consists of three papers: Paper 1, Paper 2 and Paper 3 [47] [48] [46]. These papers discuss results of the Initial Study phase of our research. Our goal was to conduct a preliminary analysis of the handover domain, determine the state of art and create a basis and direction for our future research in the handover domain. Hence, this group of papers focuses on (1) discovering the major handover problems in Paper 1, exploring the state of the art in the software handover domain and creating a preliminary taxonomy of EM*: Handover Taxonomy in Paper 2, and on demarcating the handover process in the system lifecycle in Paper 3. In short, the papers provide a basis for the research in the handover domain.

**Paper 1:**

**Core Handover Problems**


In this paper, we elicited problems faced by five IT organizations while handing over their software systems. Our goal was to uncover the core handover problems, identify their impact and identify suggestions for their solutions.

The organizations involved in this study were working in diverse domains (see Appendix D) and they were of different sizes. Two of them were large organizations, two were medium-sized and one was a small organization. Three of the studied organizations implemented a handover process as part of their development lifecycle processes. One organization experienced handover as part of a company acquisition, and the fifth one practiced self-to-self handover.

The problem elicitation was conducted in form of interviews during which we inquired about (1) the handover problems, (2) the impact of these problems, and (3) suggestions for remedying them. The result of these interviews had led us to five core handover problems. These were:

- **Insufficient system knowledge.** The transfer of system knowledge is critical for a successful software handover. The situation becomes even more critical if the handover is taking place between two different organizations. In the absence of appropriate system knowledge, a maintenance team can neither make proper corrections or modifications,
nor can they guide their customers in their daily operation. As a solution, involvement of the maintenance team during development is of utmost importance.

- **Lack of domain knowledge**: Transfer of system knowledge is not enough. The maintenance team must possess the domain knowledge as well. In fact, the domain knowledge was considered as a prerequisite for acquiring system knowledge in the organizations studied. Hence, it is very important to assess the level of domain knowledge possessed by the maintenance team or organization when designating it.

- **Insufficient communication**: Insufficient communication between the participating parties may substantially jeopardize the success of the handover process. In the absence of communication, the maintenance team may start correcting or modifying the system on the basis of wrong assumptions which, in turn, may result in a delay of handover and customer dissatisfaction. As a suggestion, the developer must communicate with the maintainer team on a frequent basis during handover. Agile practices such as daily meetings and pair programming have proven to be helpful for reducing the communication gap in the organizations studied.

- **Difficulties in tracking changes**: The difficulties in tracking changes made to a system during handover may result in inconsistent system versions. As a solution, a common version control repository should be established to be accessed and used by both development and maintenance teams.

- **Inadequate documentation**: Good quality documentation is essential for understanding and supporting the system. However, documentation is given secondary importance due to the pressure to deliver the system on time. Hence, it is not always correct, consistent or complete. The organizations suggested that hiring technical writers for documenting the system might be a good solution.

The handover problems elicited in this study are only the core problems. There may be many more problems that might be triggered from these core problems. This is clearly indicated by the magnitude of the difficulties that may arise during handover. In our future research steps, these core problems provided an important feedback for designing our taxonomy of handover activities.

**Paper 2:**

**Taxonomy of Handover Activities**


This paper presents the preliminary version of taxonomy of handover activities. It is called *Evolution and Maintenance Management Model (EM³) Handover Taxonomy*, or shortly *EM³: Handover Taxonomy*. Our goal was to determine the rudimentary handover activities and establish a basis for creating handover taxonomy.

We developed the taxonomy in two steps. These were (1) *literature survey* and (2) *taxonomy design*. In the *literature survey* step, we explored research articles, reports, standards, book chapters and various online publications dealing with software handover. Due to the fact that not much has been published on the subject, our preliminary version of the taxonomy is based on only a handful of publications [9] [23] [22] [12] [13] [19] [17]. In the *taxonomy*
design step, we extracted the activities from the literature and arranged them under different
categories based on their logical relationship. In this paper, we called each category a
component. Later on, when creating $EM^3$ Handover Framework in Paper 14, we call each
category a component practice.

Our preliminary version of the taxonomy consisted of seven components. These were
Management and Administration, Maintenance Environment, Version and Configuration Management,
Deployment, Training, Documentation and Maintainability Management (Appendix A.1). In total, it
consisted of 53 activities, out of which 32 were main activities and 21 were their sub-
activities. The taxonomy components with their activities are listed in Appendix A.1. They
were the following:

- **Management and Administration** component including activities significant for managing
  the handover process. It contains activities important for identifying the maintenance
  organization, establishing a transition team, creating a transition plan, defining
  transition resource requirements and developing management plans.

- **Maintenance Environment** component containing activities required for setting up
  maintenance environment. It includes activities important for determining and
  installing hardware and software suite, determining/assessing and installing software
  support suite, installing software baseline and data, and finally, for transferring and
  placing modification requests in a repository.

- **Version and Configuration Management** component including activities needed for tracking
  and controlling changes in the software system during handover. Its activities include
  establishing software configuration baseline, putting software under configuration
  management and placing software under version control.

- **Deployment** component containing activities required for installing the system on the
  acquirer site. These are the activities needed for developing installation procedures,
  installation and planning for future releases.

- **Training** component including activities important for providing training to the system
  stakeholders. It includes the provision of training on system and its operation, training
  on maintenance processes, training on system support processes, training on new
  technology to be used in the system, and finally, the provision of onsite training in
  critical cases.

- **Documentation** component including documentation related activities useful from the
  handover perspective. It includes activities for establishing documentation repository,
  defining services of the repository, subjecting repository to software configuration
  management, and for establishing standards for documentation development.

- **Maintainability Management** component including activities for continuously monitoring
  and controlling the maintainability of the system to be handed over. It covers both
  system and data maintainability.

Summing up, this study helped us to create an initial version of the basic set of handover
activities that was used in our further research phases. It was this preliminary version which
was then continuously explored, evolved, improved and evaluated within the industry.
In this paper, we explored and demarcated the scope of a handover process within eighteen organizations. We investigated (1) whether the organizations had conducted a handover process and what terminology they used when referring to it, (2) where in the system lifecycle they placed the process, (3) what roles were involved in the process, and (4) what activities were performed in the processes studied. Our main goal was to find out how the industry understood the handover process and how it placed it in the lifecycle of a software system.

This study was critical for identifying different contexts of the handover process, its span and complexity, and for understanding the relationship between handover with other processes. Only then we would be in a position to prepare ourselves for exploring the handover process in detail. The study was conducted in form of semi-structured interviews that were based on three different software lifecycle roadmaps. As shown on the left-hand side of Figure 4.1, the three roadmaps visualized the software lifecycle from different perspectives. They were designed (1) from the product perspective illustrating product transfer between major and minor revisions, (2) from the process perspective visualizing the transfer between the phases in the whole software lifecycle, and (3) from the development cycle perspective visualizing the phases within a software development cycle. In this way, we ensured that our interviewees would be able to identify at least one placement of handover process.

The study results are mainly (1) the activities taking place within handover process and the roles involved in them, and (2) the placement of the handover process within software lifecycle. Due to space restrictions, we do not present them all in this paper summary. We only focus on the placement results. However, we would like to point out that all the results of this study constituted an important basis for scrutinizing and enhancing our future work. As shown in Figure 4.1, we elicited seven different placements of the handover process. These are:

1. **Context 1**: Handover between major versions only,

2. **Context 2**: Handover between major versions and major to minor versions,

3. **Context 3**: Handover from initial development to evolution,

4. **Context 4**: Handover repeated after every evolution,

5. **Context 5**: Handover of the system to be serviced taking place only after it got ensured that no more evolution would be done on the system.
6. **Context 6**: Handover initiated during deployment and executed in parallel with deployment.

7. **Context 7**: Handover implemented in multiple stages.

By studying the above-listed contexts, we realized that the boundaries of the handover processes are highly flexible and that there may be many more contexts that might be relevant for the handover process. We also understood that many of its activities strongly overlapped with other processes such as development and maintenance. This implies that many of the development and maintenance activities are important constituents of the handover process. Only a minor set of the handover process activities can be classified as pure handover activities. They mainly concern planning, management and implementation of the handover process.

### 4.2 Small-Scale Exploration publications

The second group consists of four papers: Paper 4, Paper 5, Paper 6, and Paper 7 [49] [50] [51] [52]. These papers discuss results of the Small-Scale Exploration phase of our research. Here, we conducted four case studies in which we intensively studied and analyzed three different handover processes in three different companies. Our goal was to evaluate the EM³: Handover Taxonomy in industry and obtain feedback for taxonomy evolution and improvement. The feedback resulted in many descriptive results that constituted input for better understanding of the handover process.
This paper communicates our first evaluation of \textit{EM}³: \textit{Handover Taxonomy}. The evaluation was conducted in a Swedish organization - \textit{AMIS Software}. The context studied was an in-house handover in which software systems were either continued to be maintained by the developers or they were handed over to a separate in-house maintenance team. They might or might not involve COTS suppliers. Our goal was to provide insight into the industrial status of a handover process, and thereby, evaluate the taxonomy and gather feedback for its improvement.

Our results show that the majority of the activities as defined in \textit{EM}³: \textit{Handover Taxonomy} are relevant and that they were used within \textit{AMIS Software}. We have however identified a few discrepancies between the company’s handover processes and our taxonomy activities. These are the following: (1) transition teams are established for only large scale projects, (2) simple systems are not put under version and configuration management, (3) the configurations of the legacy systems are managed manually, and (4) white-box testing is classified as a support activity and not as a training activity.

The study has led to the identification of several important activities. They were all based on the following lessons learned:

- There are other roles than the maintainer, developer and acquirer role participating in handover. These may also include COTS supplier.
- Early designation of a maintainer may not always be optimal. It may change during development due to better understanding of the system and its maintenance needs.
- The constellation of the transition team strongly depends on the transition complexity, which in turn, depends on the system complexity and the number of actors involved.
- Just because the transition studied concerned an in-house transition, we found out which of the activities were of a trivial character, if performed within one and the same organization. For instance, determining maintenance environment does not require much effort because it is the same for both the developer and maintainer.
- At least three baselines need to be managed in the transition process. These are system test, acceptance test and operational baselines.
- It is significant to create a training infrastructure resembling the operational environment that is populated with relevant data.
- The training part of the handover process may be as complex as the whole transition process itself. Many times, it may include training on COTS and it may require setting up complex training environments. It may also require adaptations to different target groups such as management and non-management staff and adaptations to different
needs such as novices and refreshers. Data maintainability and its handover are as important issues as system maintainability.

- There must be a clear policy for how to introduce new releases especially in case of using third party components.
- The complexity of deployment is substantially augmented if the system is deployed on multiple nodes.

**Paper 5: Towards Taxonomizing of Core Software Handover Activities**


In this paper, we have evaluated the preliminary version of *EM*: Handover Taxonomy in a Pakistani organization - CRM Telecom. The context studied was an in-house handover from a development team to a maintenance team. The maintenance team’s role was to conduct corrective maintenance only. The goal with this study was to evaluate the taxonomy and obtain feedback for extending and improving it.

The study enabled us to compare our taxonomy in the real-world handover context, examine its usefulness, improve and expand it with new activities and learn new lessons. It was based on a series of interviews with the development manager of CRM Telecom. Here, we used a semi-structured questionnaire stating what, when, who and how questions about each activity of *EM*: Handover Taxonomy. The questionnaire also inquired about the missing activities in the taxonomy.

We found that *EM*: Handover Taxonomy was realistic in the CRM Telecom context. Overall, CRM Telecom followed all the activities listed in *EM*: Handover Taxonomy to a varying degree depending on whether handover concerned a handover of a newly developed system or a handover of a new system release, which, in turn, strongly impacted the budget and placement of the handover process within software lifecycle. When studying their handover process, we have observed the following differences:

- There exist three baselines under version and configuration management. These were acceptance testing baseline, maintenance baseline and operational baseline.
- Modification requests are not transferred to the maintenance team due to the fact that the team was dedicated to the corrective maintenance only.
- Training was not a time consuming activity at CRM Telecom thanks to continuous intensive interaction of the stakeholders involved and the “train the trainer” policy.

This study helped us to learn many lessons. Some of them are:

- The initial phase of post-delivery and operation is critical and maintainers should be chosen with great care. They must possess overall knowledge of the system and the ability to work under stress.
• The manpower resources are rotated between development and maintenance teams in order to evenly distribute the development and maintenance effort.
• Members of the transition team may include product delivery teams.
• Developers play the role of a temporary internal support team to the maintenance team during handover phase.
• As much as one fourth of the release cycle time may go to the handover phase.
• Communication between all parties involved in handover is a challenging task.
• The maintenance team must have the same environment as the operational team.
• Handover process does not only include the transfer of a system but also the transfer of the associated data which, in turn, may strongly impact the complexity of the handover process.
• Developers and maintainers must regularly synchronize their codes with new corrections.
• The maintenance team does not need to have access to operational data. They may keep their replicas. In this way, one prevents the unwanted changes to the operational database.
• Planning for future releases starts in parallel with the current release development.

Paper 6:
Evaluation of Handover Activities from the Perspective of the System Acquisition and Front-End Support

In this paper, we have evaluated EM³: Handover Taxonomy in an organization VAST Pakistan. The context studied covers the handover of a software system from the vendor conducting system development, evolution and maintenance to VAST Pakistan - an acquirer organization using the end product and conducting front-end support.

The evaluation was conducted as a case study comprising a series of interviews with Business Support System executive who was part of the handover team. Our goal was to find out whether the taxonomy activities were realistic and relevant from the company’s perspective in its role as both a system acquirer and front-end supporter.

Our results show that most of the EM³ handover activities are relevant in the context studied. Some of the activities are however not applicable in the front-end support context. They mainly concern (1) implementation and testing related activities, (2) management of documentation and maintainability, (3) installation of data and transfer of modification requests, and creation of configuration baseline. Overall, we also learnt that the constellation of the transition team varied with time. At the beginning, it included mainly management roles whose task was to mainly plan for handover. With time, it included roles involved in implementing the handover.
Our study has also revealed the need for more activities in the Deployment and Training components. Regarding the Deployment component, our taxonomy lacked important activities concerning (1) the planning for system deployment, (2) installation of operational data, (3) testing of system touch-points, and (4) monitoring of the system operation. The Training component, on the other hand, did not include the system deployment and testing as important training activities. It also included some redundant activities concerning provision of training on technology and system support process.

This study has made us conclude that the choice of handover activities mainly depends on the organization’s or team’s responsibility towards the system. It depends on whether the organization is the acquirer, front-end supporter, back-end supporter, vendor or a combination of all these roles. Organization’s size, location, working culture and maturity may also determine the choice of the taxonomy activities.

**Paper 7:**

Walking a Software Handover Process


This paper is a continuation of the former paper. In the former paper, we have evaluated EM3: Handover Taxonomy. In this paper, we have walked through a handover process using the taxonomy activities. We did it in the same organization which was VAST Pakistan. Here, we walked through their handover process by following the company’s software lifecycle model and by mapping EM3’s activities on them. Our goal was to gain understanding of a handover process and its execution from the software lifecycle perspective and to provide a basis for creating a handover process and guidelines.

VAST Pakistan’s handover process was performed across five phases. These were (1) system requirements specification, (2) system design and development, (3) pre-deployment, (4) system deployment, and (5) system maintenance and support. It was conducted by two companies: (1) VAST Pakistan playing the role of an acquirer and front-end supporter and (2) VAST Pakistan’s vendor.

The first phase, the system requirements phase, was conducted by the company studied. Here, they identified maintenance and transition teams, created different plans important for managing handover and determined maintenance support suite. The next-coming phases, however, were conducted by the VAST Pakistan’s vendor. Due to its acquirer and front-end supporter role, VAST Pakistan was not much involved in this phase. However, the company ensured that the vendor developed installation procedures, established standards for documentation development and created training material.

VAST Pakistan resumed its active participation in the handover process in the pre-deployment phase. Here, the company was involved in testing the transitioned system, assessing data maintainability, installing hardware, software and support suites, planning for deployment and establishing the documentation repository.
VAST Pakistan assisted in the deployment phase including the activities required for installing the software baseline and operational data, establishing VCM baseline, testing the deployed system, and finally, for monitoring the system operation after deployment. Here, they monitored the system for a predetermined time. If they got satisfied with the system operation, then they signed off the acceptance of the system, a procedure that ended the system acquisition process.

The last phase of the handover process is system maintenance and operation. Although system acquisition ended in the deployment phase, there still existed handover activities to be performed in the initial phases of the post-deployment stage. These activities mainly concerned (1) provision of training of the operations team, if required, (2) subjection of documentation repository to SCM, and (3) planning for future releases. It is worthy to mention that training activities spanned the whole handover process. Their choice, timing and intensity strongly depended on whether the acquired system was a newly developed or an enhanced one.

Summing up, this study enabled us to map the handover activities on the real-life software system lifecycle. It has contributed with details on the sequence of activities to be chosen when performing a handover process. However, we would like to emphasize that this sequence provides only one scenario of a handover process. Hence, the feedback achieved in this paper provides a good ground for understanding the process. Still, however, it does not provide any ground for generalizing the sequence of its activities.

4.3 Large-Scale Exploration Publications

The third group consists of six papers: Paper 8, Paper 9, Paper 10, Paper 11, Paper 12, and Paper 13 [56] [58] [55] [54] [53] [57]. These papers discuss the results of the Large-Scale Exploration phase of our research. By the time we finished our small-scale exploration, we understood that the scope of the handover process was very huge. Still, however, due to the strong diversity and complexity of the processes, we decided to conduct a very extensive study that would grasp the whole handover scope and that would be conducted within many companies. Our goal was to elicit as much theory about handover as possible. Here, we investigated the taxonomy under different contexts and obtained feedback from the industry which helped us to (1) examine the usefulness and relevance of the taxonomy, (2) find out whether the taxonomy had met the needs and requirements of the software industry today, and (3) to create a basis for further research in software handover.

We collected data from 36 organizations. Due to the large magnitude of taxonomy domain, we divided our study into two parts called Part 1 and Part 2. Out of the 36 organizations studied, thirteen organizations participated in both studies. Documentation component was included in both Part 1 and Part 2. Hence it contains data from 36 organizations.

In Part 1, we explored three components of EM$^3$: Handover Taxonomy within 29 organizations. These were Management and Administration (MA), Maintenance Environment (ME) and Version and Configuration Management (VCM). Part 2, on the other hand, consisted
of Training (T), Deployment (DP) and Maintainability Management (MM). It was explored in twenty organizations.

Due to the large magnitude of data produced in the Large Scale Exploration phase, it was not possible for us to share all the data in one publication because of maximum number of page restrictions. Hence, we produced six publications to share our knowledge with the research community. Below, we summarize the papers results.

4.3.1 Large-Scale Exploration publications - Part 1

Paper 8: 
Identifying and Tackling Diversity of Management and Administration of a Handover Process
Ahmad Salman Khan and Mira Kajko-Mattsson, 12th International Conference on Product-focused Software Process Improvement, Torre Canne, Italy, 2011, pp. 156-170, ISBN: 978-3-642-21843-9, DOI: 10.1007/978-3-642-21843-9_14, Lecture Notes in Computer Science; Springer

In this paper, we have evaluated the Management and Administration component of EM³: Handover Taxonomy in 29 organizations. The component addresses activities for planning and controlling the transition process, its schedule, budget and resources. The evaluation was done in form of semi-structured interviews focusing on getting the understanding of the EM³ activities and their contexts. Our goal was to find out whether the component was realistic and whether it had met the needs and requirements of the industry.

Our results show that the organizations studied implemented all the management and administration component activities. Those that were not implemented were usually not right for the context. Our results also indicate a strong diversity in the implementation of the activities, their placement within a software lifecycle and the roles involved. Overall, we made the following observations which were important for improving the handover taxonomy.

- There exist three contexts for designating a maintenance organization. These are (1) maintenance stays with the development team, (2) maintenance is performed by a separate team in the same organization, and (3) maintenance is performed by a separate organization.
- The identification of the maintenance team depends on parameters like system size, system criticality, number of manpower resources available for maintenance and maintenance outsourcing cost.
- The choice of the maintainer depends on the context change. Even if about 50% of the organizations studied identified their maintenance teams early in the development phase, they were still not certain whether the identified maintainer was the right one.
- A number of maintenance team members are selected based upon the maintenance workload and priority of maintenance tasks. New maintenance team members are recruited only if the maintenance budget permits
• Ways of establishing transition teams strongly vary depending on who does maintenance. In many cases, the transition team is established only for large projects. In small projects where maintenance stays with development, no transition team is being formed.

• The constellation of the transition team changes with time. Even if the transition teams are created early, they are still not fixed. The constellation of its members change as there is a need for more roles to be involved in the transition with time.

• Either organizations develop an independent transition plan or include transition plan as part of their overall project plan. The structure of a transition plan considerably varies from case to case. It is a simple document in cases when the same development team continues with maintenance. It is a very comprehensive document in cases when transition takes place between separate development and maintenance teams and/or organizations.

• Transition budget is either treated separately or as part of the overall project budget. Usually, however, project managers and finance department representatives are the major players in determining the transition budget.

• Transition schedule is created at the same time when the transition plan is being created by the project management. Initially, however, it only defines deadlines for the main transition activities to be completed. The schedule details are then handled by the development and maintenance teams that setup a time schedule to complete each activity.

• Regarding transition procedures, they either define general procedures to be then followed by all the projects or they define separate procedures for each project. They are realized in form of breakdown structures for each of the major transition task.

• Resource requirements are an important part of the transition plan. They vary with respect to each project. If maintenance is performed within the same organization then the only resource would be manpower. In other cases, it may include hardware and software resources.

• Finally, the roles responsible for MA activities strongly vary. Even if project manager is the key person, still the set of roles responsible for performing MA activities varies depending on the type of transition and system complexity and criticality.

Summing up, we understood that transition process complexity strongly varies depending on who transfers to who. It may range from merely a handful of activities to very monumental processes. Moreover, the choice of the maintainer depends on the software lifecycle the software system is in. In cases when an existing software system has been evolved and maintained, the maintainer is already known in advance. Hence, it is not an issue to identify him. In case of new development, there is a need to identify the future maintainer.
In this paper, we have evaluated the *Version and Configuration Management (VCM)* component of *EM³: Handover Taxonomy* in 29 organizations. The component includes activities required to keep track of changes made to a software system during handover. The activities deal with the establishment of configuration baselines and placement of software under configuration management and version control.

Even if not all the activities were always performed by the organizations studied, all of the interviewees agreed that all the activities as defined in the *EM³: Software Handover* were highly relevant and important for an effective and efficient handover process. However, the complexity of conducting these activities depends on the type of handover, system size and the number of system acquirers.

In intra-organizational handover contexts, development and maintenance teams follow the same standards and mechanism for VCM activities, and therefore, the VCM process is less complex. In case of inter-organizational handover, there exist two variants. If maintenance team takes full responsibility for the evolution and maintenance, then the team usually works according to their own organizational VCM policies. However, if the maintenance team is responsible for only corrective maintenance, then it usually follows the standards for version control as defined by the development team.

The complexity of the VCM process increases with the increased number of acquirers. Sometimes, certain system components need to be altered according to specific customer’s needs. This results in different variants of the same component and different configurations of the system. The information about all these variants has to be handed over to the maintenance team. If not properly managed, it may imply immense problems for the maintainer when reconstructing the right versions and configurations for the right customer.

Various roles take part in implementing VCM activities and therefore, the constellations of the teams involved strongly depends on the nature of handover. In case of intra-organizational transitions, software configuration management team and development team manage all the VCM activities. In case of the inter-organizational transition, the constellation gets more complex in nature. Here, project managers on both sides and also maintainer provide input for deciding on configuration management and version control rules.

Defining and identifying the configuration items is important for tracking and controlling the items during system transition. To track that the right configuration items have been handed over, the organizations have created registers of configurations items to be checked off when handing over the software system.
It is important that the development and maintenance teams have the identical copies of the operational environment. In this way, the maintainers may provide better feedback to the developers on the problems encountered during operation and maintenance, and the developers may better understand the maintainer’s problems and requests to be attended to in the future system releases. In addition, we have learnt the following lessons:

- Maintainers must have access to the developers’ version control management in case they are involved in future system evolution.
- System test baseline is not critical for handover. However, it is a prerequisite for creating next baselines that are, in turn, critical for system handover.
- The point in time when the VCM activities get implemented varies. The implementation of the majority of the activities begins in the development phase and continues with the software development and evolution. Some activities have however fixed points in time, such as for instance, establishment of system test baselines.
- Finally, some organizations that develop and maintain critical systems prefer to configure their systems manually rather than using software configuration management. In their opinion, it is safer to do so.

**Paper 10:**

**Evaluating the Role of Maintenance Environment Activities in Software Handover**


In this paper, we have evaluated the Maintenance Environment (ME) component within twenty nine companies. The component includes activities required for determining and managing software maintenance environment. The activities concern the management of hardware and software suites, support software, data, software baseline, maintenance support suites, and finally, the management of modification requests.

Our study shows that the determination of the hardware/software suites depends on whether maintenance stays with development or not. In case it does, one continues with the suites that have been used in the development phase. In case it does not, then maintainer must determine the suites by considering the development environment, the operational environment, and the feedback from the development organization. Our study also shows that not all the organizations studied use maintenance support suites. As many as almost one third of the organizations studied provide support in a manual manner.

We found that the maintenance and operational environments must be identical. They must have the same versions and configurations of the software and hardware suites. Only then the maintenance team can reproduce problems faced by the customer and provide an appropriate solution. Similarly, the maintenance team must have the same baseline version of the software system operating on the customer site. Moreover, any changes made to the software system should be communicated and synchronized with the maintenance team.
We discovered that it is a prerequisite for providing maintenance services that the maintenance team must have access to the customer operational data. Only then they will be able to investigate the software problems. However, due to data integrity and data security reasons, maintainers have access to only the replicated operational data.

It is significant to form common repository for tracking modification requests. Both development and maintenance teams must have access to the repository during handover stage. We also realized that the transfer of modification requests from development to maintenance team during software handover is a complicated issue. It mainly depends on the services to be provided by the maintenance team. In case the maintenance team is in charge of evolution and maintenance, the modification requests are transferred to the maintenance team. If the maintenance team, on the other hand, only corrects software problems, then the modification requests stay with the development team who incorporates the corrections in the next revision or version of the system.

Maintainers start participating in resolving the modification requests in the acceptance testing phase with the help of the development team. Complete transfer of modification requests occurs in the post-delivery maintenance phase, when the maintenance team is fully familiar with the system and the maintenance process.

Finally, we could not found any clear pattern for placing the ME activities within lifecycle phases. However, the majority of the activities were performed either in deployment or maintenance phases.

4.3.2 Large-Scale Exploration publications - Part 2

Paper 11:
Evaluating a Training Process in a Software Handover Context

In this paper, we have evaluated the Training component within 20 organizations. The component addresses activities for educating and training the stakeholders involved in software handover. It includes five main activities required for training system stakeholders in a handover context. These concern training on (1) system structure and operation, (2) maintenance process, (3) support process, (4) new technology to be used in maintenance and operation, and finally, (5) on the provision of on-site support. The stakeholders are maintenance team, support team, acquirer and development team.

We discovered that the training activities were relevant for the organizations studied. However, the nature of the training activities differs according to the handover context, complexity of the system and the service level agreement for maintenance. Below, we briefly summarize the core findings.

Training on the system and maintenance and support processes is context dependent, and is often conducted on two different formality levels. Either it is conducted on a formal
level by arranging special training sessions in form of traditional classroom courses or workshops or on a rather informal level via informal discussions, meetings, emails and telephone. For instance, in cases when maintenance is conducted by a separate team that may or may not be geographically distributed, then the training is conducted in form of specially organized courses. These courses may be traditional classroom courses or the video conference ones. Regarding the cases when maintenance is conducted by a separate in-house team, the scenario may vary. Training may be conducted both in a formal and informal manner. In cases when maintenance stays with development, then not much training is done. The developers already possess system knowledge, and therefore, they do not require any special training.

The complexity and scope of providing system training varies as well. It may depend on the training needs, the size and complexity of the knowledge to be transferred, or it may simply depend on the scope of the agreement as defined in service level agreements. In some cases, it may even require a specially designed environment, the environment which is similar to the operational one. It may include presentation of the system on different granularity levels in several consecutive rounds. For instance, the maintenance team may first get an overview of the system to then receive a series of detailed trainings about its structure and functionality.

Choice of who trains who also varies. Usually, in companies where maintenance is conducted by a separate in-house team, it is the developers who train maintainers and support personnel first. The maintainers or support personnel then train the acquirer’s end users. Regarding the training of the acquirer, it is mainly directed towards selected members (super users) of the acquirer’s support team. These members, in turn, train the end-users in the acquirer organization.

The ways of creating the training material and making it available varies somewhat. The material is created and/or updated either on a continuous or on an as-needed basis by mainly developers, maintainers and technical writers.

Regarding the training on new technology, the organizations usually send their employees on external courses whenever they experience a technological need that they cannot learn on their own. In other cases, new technology training may be provided internally either in form of a seminar or assignment of resources to individuals for the learning purposes.

Finally, other forms of training may include onsite support or maintainers’ assistance of developers. Onsite support is provided by sending experienced consultants to support the maintainer’s onsite. This usually takes place in critical cases. Regarding the maintainers’ assistance, maintainers assist developers in solving some of the modification requests during acceptance testing and/or deployment. They usually do it if the modification requests are few or if they are of a minor character. If they are many or if they are of a serious character, then the development team must solve them by themselves. Otherwise, there exists the risk of deployment and handover being delayed.
In this paper, we have evaluated the Deployment component containing the activities that are critical for deploying and handing over software systems. We did it within twenty organizations.

The organizations studied perform the majority of the activities in the EM’s Deployment component. Our results however reveal that deployment is a highly diverse process. The complexity of implementing its activities varies according to the size of the deployed system, nature of the deployment and forms of deploying software.

From the handover and maintenance perspective, our results show that the companies studied are well prepared for performing handover. By defining the scope and contents of the release, they get an understanding and indication of the magnitude of the deployed software, and thereby, they acquire a preliminary perception of the magnitude of the handover process and the burden of future maintenance responsibilities. This, in turn, aids them in planning resources for and in directing their future maintenance activities.

We discovered that the scope and contents of the software system to be handed over may not be always fixed upfront. It may be redefined as late as either during acceptance testing or deployment itself. This, in turn, affects the handover and maintenance processes. Their plans should be adjusted to the new scope and contents.

We found out that by listing the organizations affected by the release or release variants, one gets an idea about the complexity and dimension of the future maintenance process. Some releases may not be of interest and use to all the customers. This helps the maintenance organizations identify the customers that will need support and maintenance in the future.

By identifying roles responsible for releases, the organizations studied have one focal point of contact for communicating on the status of the releases to all the parties involved. This makes them well equipped to co-ordinate their work with the work of the roles responsible for the handover and future maintenance. By getting continuous feedback on late deliveries, changed release scope and contents, changed system quality, installation problems, release rollbacks and other things, they may take appropriate actions and re-plan their activities, if necessary, and thereby, be better prepared for performing their duties during and after handover.

By preparing release documentation, the deployment process provides important source of information for the handover and maintenance teams. It is a formal document providing evidence of what has been transferred from development to maintenance which then provides a platform for planning maintenance work, creating maintenance environment and for defining or redefining service level agreements (SLAs).
Deployment readiness test is an important tool in the context of system handover and maintenance. It aids either in confirming the original transition and maintenance plans or in re-planning them, if need arises. Also, by recording incidents taking place during deployment, the roles responsible for handover are able to have insight into the changes made to the release scope and contents and take appropriate actions. There may be no handover at all, or a handover of only a system part. This also aids maintainers in quickly re-planning future maintenance.

By determining deployment forms and by defining system distribution structure, both the handover and maintenance teams get a better understanding of the complexity of the handover process. Also, the complexity of the handover process is strongly dependent on the distribution. The more the system is distributed across the geographical locations, the more parameters must be considered during handover. Finally, the distribution of post-delivery responsibilities greatly influences the complexity of the deployment, handover and maintenance processes. The more international distribution, the more synchronization across all the actors is required.

Knowing the size of the transitioned system greatly impacts the choice of the deployment roles. Small systems require only one or two individuals to handle the whole deployment process. Large systems, on the other hand, require a big release or deployment team. This, in turn, impacts the complexity of both the handover and maintenance processes.

The results of the review and closure of the deployment is an important determinant of whether the actual handover and future maintenance will take place. It is the final milestone for the system handover and an initial milestone for software operation and maintenance. It also provides a basis for planning future releases. By providing their feedback on their experiences both from the deployment and post-deployment phases, the maintainers may influence its scope and contents and make sure that it suits their future plans.

Finally, we could not distinguish any clear pattern of assigning particular roles to particular activities and placing particular activities within particular software development lifecycle phases. Reasons are many. The deployment process is strongly diversified, and thereby, difficult to standardize. Roles assignment basically depends on the handover context, process and its complexity.

**Paper 13:**

**Management of Documentation and Maintainability in the Context of Software Handover**


In this paper, we have studied the activities required for managing documentation and maintainability that need to be considered during software handover. The study was conducted with thirty six practitioners. Our results show that the activities are relevant and that the majority of the companies perform them. However, their performance strongly varies depending on the organizational and handover context at hand. Below, we present our findings.
The documentation repository is an important player in the context of software handover. It serves different purposes. It is used by developers to understand the requirements, by customers to learn to operate the system using user manuals, by management to see the project progress and assess the handover readiness, by system testers to test the system, and by maintainers to learn the system and to conduct maintenance. Due to strongly varying needs, all these above-listed role groups should have different access rights to the documents in the documentation repository.

We found out that a documentation repository exists either at an organizational level and/or at a project level. It is established in two ways: either offline or online. The online repository is especially beneficial for inter-organizational handover. Since the repository is available online, its contents can be easily shared with the maintenance team working at a remote location.

Not all organizations subject their documentation repositories to SCM. Those that do it find it of great help in assuring the quality and consistency of the system documents during handover. Despite this, the documentation is not always consistent with the system. This impacts the ability of the maintainers to conduct maintenance. To remedy the inconsistencies, some organizations notify the document owners about the missing or incomplete documents via the documentation repository.

Templates and guidelines for using the templates constitute the documentation standards. They may be shared by both the developer and maintainer in cases when maintenance stays with development or when maintainer commonly agrees upon them with the developer. If not, then the maintainer may have their own standards that are different to those of the developers. This should be considered when designing the handover process models.

Regarding maintainability, we have observed that when assessing system maintainability, the organizations studied assess system documentation, system architecture, and coding and naming standards. If maintenance stays within the organization, then the organization’s maintainers have the opportunity to assess system maintainability in good time before handover. If maintenance is however transferred to a third party, then the maintenance assessment takes place in the deployment phase. This is definitely too late for the organizations taking over the system to react to the maintainability deficiencies.

Regarding data maintainability, the organizations have pointed out three ways of assessing data maintainability. These are the following:

1. Data maintainability is continuously assessed during development. It mainly includes assessment of data structures and database capability for future updates. It is then tested and assessed by test engineers. In light of the testing results, the data structures may then be adjusted or redesigned, if required and necessary.
2. To assure data consistency, the access to the operational data is restricted in the maintenance phase. Therefore, two identical copies of databases are created; one dedicated to the operational activities and the other one dedicated to the testing, training and simulation activities.
3. Scalability tests ensure that the data structures can hold data under maximum data traffic in the future. There are two major goals of assessing data scalability. These are (1) test the workload on the system and (2) test the ability of the system to handle huge incoming data.

4.3 Final Study publication

The last group consists of Paper 14. The paper discusses the results of Final Study phase. Our goal was to implement the taxonomy in industry, and thereby, to evaluate its industrial realizability and usefulness.

**Paper 14**

**Evaluating and Acting on the Handover Framework in a special Case of Project Parking**

*Ahmad Salman Khan and Mira Kajko-Mattsson, submitted to Journal of Software Maintenance and Evolution*

In this paper, we have evaluated $EM^3$: Handover Framework via participatory research in an industrial handover setting. Our goal was to examine the usefulness of the framework in a real-world scenario. The company involved was a Swedish organization which we, in this thesis, identify with a fictitious name, $E$-Identity.

Due to financial reasons, the organization could not continue its business operation at the moment of our study. They however intended to resume it as soon as their finances became better. To be able to resume its operation in the future, $E$-Identity was in the process of conducting a special case of a self-to-self handover process during which it handed over a software system from a development team going to be dissolved to another development team to be created in the future. Since the transitioner and the transitionee would not be able to communicate with each other, the company believed that the transition process would be of a complex nature.

While conducting our study at $E$-Identity, we first studied the company’s processes that were relevant for the handover process. We then actively participated in the process by working as part of the transition team. We played the role of both an active participant observer. The tasks we were involved in were to participate in the morning startup meetings, provide advice and explanations concerning the handover process, suggest improvements to the process, and perform various handover activities such as editing documentation, reviewing documentation and providing suggestions in the light of $EM^3$: Handover Framework. While performing the tasks, we strictly followed $EM^3$: Handover Framework and continuously evaluated it. Summing up, our involvement in the $E$-Identity’s handover process was a win-win situation, where both parties reached mutual benefit and satisfaction.

Overall, $EM^3$ was fully applicable at $E$-Identity. Depending on the $E$-Identity’s context, all the framework’s practice components were relevant to a varying degree. The practices such as Maintenance Environment, Version and Configuration Management, Deployment and Maintainability Management were simple to implement. This is because the company conducted self-to-self
handover and because all the rules and prerequisites for these practices need not be transferred or managed.

The practices that somewhat differed were Management and Administration, Training, and Documentation. This is because the transitioners and transitionees had no opportunity to communicate during handover in the project parking context. Therefore, the company established a two-staged handover process. The first stage was to be conducted by the transitioners who dealt with freezing the project and all the information about it. The second stage was to be conducted by the transitionees who would resume the project sometime in the future. Just because the transitioners would quit the company, they would not have the opportunity to communicate with the transitionees. Therefore, documentation was the only source of information for the transitionees. The transitionees were expected to train themselves by studying the documentation, and by practically working on the system.

This study has resulted in the confirmation that EM³: Handover Framework is realizable in a real-world scenario. In addition, it helped us to extend the framework with one new practice called Software System Transfer. The goal of this practice is to keep track of the status of software components during handover, and monitor the software system transfer.
5. EM$^3$: Handover Framework

EM$^3$: Handover Framework provides a skeletal structure hosting six different parts that together are to be used for creating handover process models. As shown in Figure 5.1, the central part of EM$^3$: Handover Framework is EM$^3$: Handover Taxonomy presented as a set of component practices including the activities that play a significant role in executing a handover process. It is these practices that may be used for orchestrating handover process models using the other five parts such as (1) Handover Types designating three types of software system handover, (2) Handover Contexts placing handover processes within software lifecycle, (3) Handover Roles identifying the main roles having responsibilities in the handover process, (4) Handover Lifecycle Roadmap designating suitable time spaces in the handover lifecycle phases in which specific EM$^3$ activities take place, and (5) Handover Guidelines providing explanations and advice aiding the organizations in constructing their handover processes.

![Figure 5.1. EM$^3$: Handover Framework](image-url)
In this chapter, we present *EM*: *Handover Process Framework*. We first start with presenting the handover types and contexts in Sections 5.1 and 5.2. We then present the framework’s central part, that is, *EM*: *Handover Taxonomy*. We continue presenting the framework’s remaining parts such as *Handover Lifecycle Roadmap*, and *Handover Roles and Guidelines* in Sections 5.3-5.5. Regarding the guidelines, due to the space required for describing them, we put them in *Appendix E*. However, we briefly discuss them in the Section 5.5. Finally, we round up the chapter with the handover component practices to be described in Section 5.6.

5.1 Handover types

*EM*: *Handover Framework* identifies three major types of software handover. As illustrated in Figure 5.2, these are (1) *self-to-self* handover, (2) *intra-organizational* handover, and (3) *inter-organizational* handover. Except for the first type, handover implies transferring a software system from one party to another. Only in case of the self-to-self handover, maintenance stays with the development team who continues providing maintenance services. This is the simplest case of handover.

In the *intra-organizational* handover, a system is handed over from the development team to a separate maintenance team in the same organization. Finally, the third type is the *inter-organizational* handover, in which the system is handed over to a separate organization. As illustrated in Figure 5.2, there may be many separate organizations involved in maintaining one and the same system. A typical scenario here is when a third party takes over the system which contains COTS components that are to be serviced by the COTS supplier organizations.

![Figure 5.2. Software system handover types](image-url)
5.2 Handover contexts

EM1: Handover Framework distinguishes between seven different handover contexts using three roadmaps showing different ways of illustrating software system lifecycle. The roadmaps are (1) product lifecycle perspective, (2) process lifecycle perspective, and (3) development lifecycle perspective.

5.2.1 Product lifecycle perspective

Figure 5.3 shows a software lifecycle roadmap from the product perspective. This perspective illustrates the product development, evolution and maintenance across its major and minor releases (versions and revisions). Versions correspond to releases of a software system with new or enhanced features whereas revisions correspond to corrections or minor improvements made to a specific version. One version may include multiple revisions.

EM1: Handover Framework identifies two handover contexts on the product lifecycle roadmap. These are:

- **Context 1**: Handover takes place only between major versions. It is illustrated with a bent arrow between major versions in Figure 5.3.
- **Context 2**: Handover includes Context 1, and in addition, it includes handover between major versions and revisions. The addition to Context 1 is illustrated with a thick arrow between the major versions and minor releases in Figure 5.3.

5.2.2 Process lifecycle perspective

Figure 5.4 shows a software lifecycle roadmap from the process perspective [72]. The roadmap has five phases including Initial development, Evolution, Servicing, Phase-out and Closedown. Using this roadmap, EM1: Handover Framework identifies three different handover contexts. As shown in Figures 5.4, these are:

![Figure 5.3. Handover contexts in the product lifecycle roadmap](image_url)
- **Context 3:** Handover takes place from initial development to evolution stage. In this context, system enhancement and evolution become the transitionee’s responsibility. The transitioner is responsible for developing the initial system version only and the transitionee develops the next coming system versions and provides maintenance services.

- **Context 4:** Handover occurs after every evolution iteration. System evolution is considered as part of development where the system is enhanced with new features. After that, it is transitioned to another group who is responsible for its corrective maintenance.

- **Context 5:** Handover happens in the point of time between evolution and servicing. In this case, the system is handed over only after making sure that it will no longer be enhanced with new features. The system will be serviced so that it stays operational till it is time to retire it.

### 5.2.3 Development lifecycle perspective

Figure 5.5 shows a traditional roadmap that is presented by the majority of literature. It focuses on only the development lifecycle perspective where the operation and maintenance, despite their complexity and magnitude, are depicted as one of the phases at the end of the software life cycle. Using this roadmap, EM: Handover Framework identifies two handover contexts. These are:

![Figure 5.4. Handover contexts in the process lifecycle roadmap](image-url)
**Context 6:** All the activities from requirement definition till system testing are part of development. Handover gets initiated during acceptance testing.

**Context 7:** Handover takes place in three major consecutive stages. These are (1) the stage between requirements definition and system design during which both development and maintenance teams discuss and agree on the deliverables to be handed over, (2) the stage between the implementation and integration and system testing where the proposed solution is transitioned to the maintainer for approval, and (3) the stage after acceptance testing, during which, the system is moved on to the operation and maintenance and where the maintainer who takes over the responsibilities for the system.

### 5.3 Handover Lifecycle Roadmap

The time span of a handover process may range from the initial phases of development to the initial phases of operation and maintenance. Many of its activities may take place in any of the phases and still be justified to be performed in them. For this reason, *EM*: Handover Framework does not designate any specific phases in which particular handover activities should take place. As a framework, it instead provides a blueprint of the major phases in which the activities belonging to various handover practices may take place. To go in line with recent trends within software engineering [73], these major phases are represented as activity spaces and put into a *Software Handover Lifecycle Roadmap*. As shown in Figure 5.6, *EM*: Handover Framework includes three activity spaces. These are handover planning, handover implementation and handover closure.

*Handover planning* includes activities significant for preparing the system and transitionee for handover. It spans across development, system testing and initial part of acceptance testing including activities ranging from the designation of the transitionee and transition team, to planning for transition, to initial activities of the acceptance testing such as agree upon the executed handover process.
Handover implementation includes activities important for executing the actual handover. It spans from the acceptance testing phase, to deployment and to the initial part of the maintenance phase. Its activities range from installations of software and hardware platforms to provide training to the actual handover of a software system.

Finally, Handover closure includes activities important for monitoring the system and transitionee during the initial postdelivery phase. It includes activities ranging from the synchronization of code between all sites, acceptance and approval of the system for maintenance, and initial monitoring of the system. The complete list of which activities below to what handover phases is available in the Table 5.1-5.8.

### 5.4 Handover Roles

Many different roles are involved in a handover process and their choice is strongly dependent on the handover type, context, system size, complexity, organizational culture and structure, and other issues. For this reason, EM³: Handover Framework only designates the roles that are generic to almost all contexts, the roles that play an important part in the handover process and have responsibilities in the framework’s component practices. Due to the diverse nature of a handover process, however, the responsibilities may be fluid across several roles. As shown in Figure 5.7, EM³: Handover Framework identifies two groups of roles:

- **Principal Roles**: roles that are central in all software system handover contexts. The principal roles may take over the responsibilities of the practice roles. To them belong:
  - Project Manager responsible for managing and planning the handover process. Project manager plays the bridging role between different stakeholders involved in handover.
  - Developer/Transitioner responsible for transferring the system and knowledge during handover.
  - Maintainer/Transitionee responsible for taking over the system and for providing maintenance services.

- **Practice Roles**: roles that are responsible for managing and performing each constituent handover practice. To them belong the following roles:
  - Transition Team comprising the roles coming from a representative set of system stakeholders that are responsible for planning, communicating, executing, and monitoring the whole handover process.
  - SCM Manager responsible for managing versions and configurations during the whole handover process.
Figure 5.7. EM³: Handover Framework roles

- **Training Group** responsible for preparing training material and for providing training according to the needs of different stakeholders.
- **Documentation Manager** responsible for assuring that the environment required for managing system and process documentation is in place.
- **Deployment Team** responsible for deploying the system on the customer site.
- **QA Team** responsible for controlling the maintainability of the system to be transitioned.

### 5.5 Handover Guidelines

As a framework, *EM³: Handover Framework* cannot be directly reused in a specific handover process. It only provides a skeleton of the basic ingredients that need to be included in a handover process. These ingredients may be selected and orchestrated in different ways depending on the handover type, context, system complexity and other reasons. To aid in the orchestration process, however, *EM³: Handover Framework* includes guidelines constituting statements, explanations, suggestions or other indications of policies, rules or procedures by which a course of a handover process may be determined. The guidelines are practical in nature and provide directions to implement *EM³: Handover Framework* practices. The guidelines are available in *Appendix E*.

### 5.6 Handover Practices

The *EM³: Handover Taxonomy* consists of eight component practices. 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**. As shown in Figure 5.8, the central practices in *EM³: Handover Taxonomy* are the **Management and Administration (MA)** and **Software System Transfer**. These two practices are purely dedicated to the handover process. *EM³: Handover Taxonomy* calls them
the central practices. The remaining $EM^3$ practices correspond to complex and comprehensive processes whose involvement in a handover process may range from a contribution with a small set of activities to a wide range of activities covering fully-fledged individual processes. They support the central practices, and therefore, $EM^3$: Handover Taxonomy calls them supporting practices.

To provide aid in creating a handover process, $EM^3$: Handover Taxonomy focuses on listing only the activities in the supporting practices that have a pivotal role in a handover process. Therefore, it does not fully specify all the supporting practices. This section describes the practices and lists their activities, roles, and guidelines. For each activity, it then suggests the activity spaces in which the activities should take place.

### 5.6.1 Management and Administration

The *Management and Administration* (MA) component includes activities required for planning, controlling and managing the whole handover process and for coordinating the activities belonging to other processes [48] [49] [50] [51] [56]. It is of central importance in planning and implementing a handover process. Neglecting its suggested activities may result in various serious problems. Some of them are:

- Failure in establishing a transition process and a team responsible for it may lead to the failure of the transition process itself or its inefficient and ineffective accomplishment.
- Misunderstanding the complexity and magnitude of a transition process may lead to defining wrong strategies and plans for the transition, and thereby, to wrong allocation of resources and funds for the transition process.
- Failure in identifying the right transitionee in time may lead to the fact that the transitionee is not appropriately prepared for taking over the system, which in turn, may impact the maintenance quality and customer satisfaction.
Table 5.1. Management and Administration practice activities

The activity phases marked with black bars are the recommended phases for the activity at hand. The grey bars indicate that it is still possible to perform the activity but placing the activity in this phase may imply risks. Planning stands for Handover Planning space, Implementation stands for Handover Implementation space and Closure for Handover Closure space.

<table>
<thead>
<tr>
<th>Activities</th>
<th>Development</th>
<th>System Testing</th>
<th>Acceptance Testing</th>
<th>Deployment</th>
<th>Maintenance</th>
<th>Handover Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>MA 1:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td>Determine/</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>redetermine</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>type and</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>complexity</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>of transition</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MA 2:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td>Define a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>strategy</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>for transition</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>process</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MA 3:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td>Designate</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>transition</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MA 4:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td>Establish</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>transition</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>team</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MA 5:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td>Define a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>transition</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>process</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MA 5.1:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td>Identify</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>core</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>transition activities</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MA 5.2:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td>Identify</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>activities</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>that are</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>part of</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>the</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>processes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>that either</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>impact or</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>are</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>impacted</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>by the</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>transition</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MA 6:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td>Agree upon</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>the executed</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>transition</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>process</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MA 7:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td>Create/</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>adjust a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>transition</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>plan</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MA 7.1:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td>Define/</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>adjust</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>parameters</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>guiding the</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>design of</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>the transition</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>plan</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MA 7.2:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td>Create the</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>transition</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>plan using</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>the</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>parameters</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MA 7.3:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td>Define</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>transition</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>resource</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>requirements</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MA 7.3.1:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td>Define</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>manpower</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>requirements</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MA 7.3.1.1:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td>Define</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>maintenance</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>manpower</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>requirements</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MA 7.3.1.2:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td>Define</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>developer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>manpower</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>resources</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MA 7.3.1.3:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td>Define</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>transition team manpower resources, if any</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MA 7.3.1.4:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td>Define</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>other</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>manpower</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>resources, if any</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MA 7.3.2:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td>Define</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>maintenance facility requirements</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MA 7.4:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td>Determine</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>transition budget</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MA 7.5:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td>Create a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>transition</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>schedule</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MA 8:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td>Develop</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>management plans</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>necessary for transition</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MA 9:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td>Determine a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>communication model</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>to be used within transition</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MA 10:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Implementation</td>
</tr>
<tr>
<td>Monitor the</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>transition process</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MA 11:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Closure</td>
</tr>
<tr>
<td>Evaluate</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>the</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>transition</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Lacking an appropriate communication model may lead to misunderstanding and confusion which may lead to difficulties in coordinating the transition process, and thereby, increased cost of the transition process.
- Failing to monitor the transition process may lead to lack of awareness of its status, and thereby, to wrong decision making.
• Not evaluating the transition process may lead to repeating the same mistakes in future transitions.

Overall, overlooking the Management and Administration practice may result in slippage of a handover schedule and increased cost of handover. In the worst case, it may result in failure when handing over the system. This substantially impacts the credibility of the parties involved, and at its worst, loss of business.

As shown in Table 5.1, the Management and Administration practice contains eleven main activities and thirteen sub-activities. The activities span the whole handover process cycle starting with determining the transition type and complexity and ending on evaluating the finalized handover process. On this journey, the organizations must define a transition strategy, designate a transitionee, establish a transition team, define and agree on a transition process, create a transition plan, develop appropriate management plans, determine a communication model, and finally, continuously monitor the transition process.

EM*: Handover Taxonomy suggests that the role of a Transition Team be the role responsible for the MA activities. This role may be assisted by the role of the Project Manager. In cases when the system to be handed over is small, then the responsibilities of the Transition Team are to be taken over by the Project Manager.

EM* suggests that all the activities concerning defining the transition process and planning for it should take place in the development phase. This concerns the activities MA1-MA9. The activity concerning the monitoring of the transition process (Activity MA10) should be continuously conducted throughout the whole transition process. Finally, the evaluation of the transition process (Activity MA11) should take place after the transition gets completed and before the next transition takes place.

5.6.2 Maintenance Environment

It is not enough to deliver a software system to the customer and hand over the responsibilities for maintaining it. One must make sure that the maintainer has the environment that is right for the system to be maintained. Therefore, evaluating and managing software maintenance environment is an important prerequisite for enabling system handover and for commencing maintenance. If the environment is not right from the beginning, then the team may encounter the difficulties such as (1) inability to provide maintenance services due to the unavailability of software/hardware and maintenance support suites, (2) compatibility issues between hardware and software suites installed on the maintenance and acquirer sites, (3) difficulties to learn the maintenance support tools, which in turn, may delay the provision of maintenance services, and (4) difficulties to reproduce problems due to discrepancies between the maintenance, development and customer environments [48] [49] [50] [51] [55].

Setting up a maintenance environment is not a trivial activity. As shown in Table 5.2, the Maintenance Environment practice contains two main activities and twelve sub-activities. The activities mainly concern either (1) defining hardware/software needs and maintenance support suites, and then installing them in the maintenance environment, and (2) assessing
Table 5.2. Maintenance Environment practice activities
The activity phases marked with black bars are the recommended phases for the activity at hand. The grey bars indicate that it is still possible to perform the activity but placing the activity in this phase may imply risks. Planning stands for Handover Planning space, Implementation stands for Handover Implementation space and Closure for Handover Closure space.

<table>
<thead>
<tr>
<th>Activities</th>
<th>Development</th>
<th>System Testing</th>
<th>Acceptance Testing</th>
<th>Deployment</th>
<th>Maintenance</th>
<th>Handover Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>ME 1: Manage Hardware/Software suite needs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td>ME 1.1: Determine hardware/software suite needs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td>ME 1.1.1: Determine hardware and software packages constituting the hardware/software suite</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td>ME 1.1.2: Assure that hardware/software suite needs match the developer’s hardware/software suite</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td>ME 1.1.3: Assure that hardware/software suites needs match the customer’s hardware/software suite</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td>ME 1.2: Install hardware/software suite</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Implementation</td>
</tr>
<tr>
<td>ME 1.3: Grant the transitionee access permission to hardware/software suite</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Implementation</td>
</tr>
<tr>
<td>ME 1.4: Assess current hardware/software suite, if any</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td>ME 1.5: Remedy the deficiencies in hardware/software suite, if any</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Implementation</td>
</tr>
<tr>
<td>ME 2: Manage Maintenance support suites</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td>ME 2.1: Determine maintenance support suite</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td>ME 2.2: Install maintenance support suite</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Implementation</td>
</tr>
<tr>
<td>ME 2.3: Assess maintenance support suites</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td>ME 2.4: Remedy the deficiencies in maintenance support suite, if any</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Implementation</td>
</tr>
</tbody>
</table>

The already installed hardware/software suites and maintenance support suites and remedying the deficiencies in them. The Maintenance Environment practice also advocates granting required access rights to the maintainer to the hardware and software suite so that they are able to perform their duties.

EM*: Handover Taxonomy suggests that the Maintainer be the main role with the assistance of Developer and Software Configuration Manager for conducting the activities in the Maintenance Environment practice.

EM*: Handover Taxonomy suggests that the determination and assessment of the suites should take place in the development phase. Installation of the suites, on the other hand, should take place in parallel with the acceptance testing phase.
5.6.3 Version and Configuration Management

Today, organizations keep track of the changes made to their software systems through Version and Configuration Management (VCM) activities. Although it is imperative to track versions and configurations throughout the whole system lifecycle, VCM activities are especially critical during software handover.

Software handover is tightly integrated with software development and maintenance processes. Hence, it cannot be performed in isolation. During this time, a software system goes through many last-minute changes that are made both by development and maintenance teams. To avoid confusion and to enable a smooth system transfer and its future operation, both development and maintenance teams must have access to the latest version of the system configuration.

The Version and Configuration Management (VCM) practice includes activities required for keeping track of the changes made to a software system. The activities that are pivotal in the transition context deal with the establishment and management of configuration baselines [48] [49] [50] [51] [58] [18].

Failing to perform Version and Configuration Management activities may result in different problems during handover. Some of them are (1) different version identification schemes for configuration items, (2) difficulties in tracking changes in configuration items, and hence, mismatch in the state of versions on different sites, and (3) difficulties in tracking changes to baselines. Such problems lead to malfunctioning of the system after handover or use of valuable resources on figuring out the discrepancies between different versions. This ultimately results in a delay in providing maintenance services to the customer.

As shown in Table 5.3, the practice has two main activities and 15 sub-activities. The primary activities are (1) the management of versions and configurations and (2) the management of baselines. The practice suggests that the rules must be in place for labeling and grouping of configuration items. The changes made to the configuration items should be tracked and the software system should be put under version control and configuration management.

The VC practice considers two types of baselines that are significant from the handover perspective. These are test baselines and post-delivery baselines. The maintainer must play an active role when assisting developer in attending to problems arising during system testing and acceptance testing phases. All the changes made to the baselines must be notified and synchronized with all the system stakeholders. Finally, the system should be accepted and approved for operation and maintenance.

EM3: Handover Taxonomy suggests SCM Manager be the main role that is responsible for implementing the practice. The Version and Configuration Management activities begin in parallel with the development phase and continue throughout the software system lifecycle.
Table 5.3. Version and Configuration Management practice activities

The activity phases marked with black bars are the recommended phases for the activity at hand. The grey bars indicate that it is still possible to perform the activity but placing the activity in this phase may imply risks. Planning stands for Handover Planning space, Implantion stands for Handover Implementation space and Closure for Handover Closure space.

<table>
<thead>
<tr>
<th>Activities</th>
<th>Development</th>
<th>System Testing</th>
<th>Acceptance Testing</th>
<th>Deployment</th>
<th>Maintenance</th>
<th>Handover Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>VC 1: Manage version and configuration</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Implantion</td>
</tr>
<tr>
<td>VC 1.1: Define rules to uniquely identify, name and label the configuration items and their relationships</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td>VC 1.2: Define how the configuration items are to be selected, grouped and classified</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td>VC 1.3: Decide on how to identify and track changes made to customizable configuration items during handover</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td>VC 1.4: Put software under configuration management</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Implantion</td>
</tr>
<tr>
<td>VC 1.5: Place software under version control management</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Implantion</td>
</tr>
<tr>
<td>VC 2: Manage baselines</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Implantion</td>
</tr>
<tr>
<td>VC 2.1: Establish test baselines (system test baseline, acceptance test baseline)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Implantion</td>
</tr>
<tr>
<td>VC 2.1.1: Assist developers in attending to problem reports during acceptance testing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Implantion</td>
</tr>
<tr>
<td>VC 2.1.2: Identify and track customizable configuration items during handover</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Implantion</td>
</tr>
<tr>
<td>VC 2.1.3: Keep track of the changes made to the baselines</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Implantion</td>
</tr>
<tr>
<td>VC 2.1.4: Notify about the changes made to the system to all the stakeholders involved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Implantion</td>
</tr>
<tr>
<td>VC 2.2: Establish post-delivery baselines (operational baseline and maintenance baseline)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Closure</td>
</tr>
<tr>
<td>VC 2.2.1: Check whether the reported problems are not of critical nature</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Closure</td>
</tr>
<tr>
<td>VC 2.2.2: Synchronize system changes made during system handover in all the environments (operational, development and maintenance)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Closure</td>
</tr>
<tr>
<td>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</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Closure</td>
</tr>
<tr>
<td>VC 2.2.4: Accept and approve the system for operation and maintenance</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Closure</td>
</tr>
</tbody>
</table>
5.6.4 Training

Handover is a critical process in the software system lifecycle. To assure its success, all the stakeholders involved should handle and direct it with the right degree of knowledge, skill, experience, and caution. Therefore, they should be adequately trained and possess adequate knowledge of the transitioned system, of the processes to be performed, of the technology to be used, and the like. It is only then they can perform their handover duties in a proper way.

The level of training should be adapted to the level of skills required for each stakeholder group. For instance, management requires overall system knowledge to be able to manage the handover, maintainer needs detailed system knowledge to be able to correct or enhance the system, and support team requires detailed operational system knowledge. In addition, maintainers and support technicians require training on maintenance and support processes.

It is important that a handover process includes the Training practice. Otherwise, the risk is that one may fail with the whole handover process or may substantially obstruct its effectiveness and efficiency. A properly defined Training practice aids in avoiding mistakes such as providing training in an unstructured and ad-hoc manner, mismatching the training topics with the training needs, failing to identify the trainee groups and their needs, and failing to create appropriate training environments and materials.

The EM1 Training practice includes eleven main activities and six sub-activities whose purpose is to train all the stakeholders involved in a handover process. The trainees are system maintenance team, support team, acquirer and in some cases development team. As shown in Table 5.4, the activities in the Training practice focus on designating a role responsible for the training process, training planning, identifying training topics, identifying trainee groups and training needs for each group, creating or updating training material, identifying the role responsible for providing training, and preparing for and providing training [48] [49] [50] [51] [54].

As part of training, the Training practice advocates the involvement of the maintainer (transitionee) in attending to modification requests and white box testing for training purposes. It suggests the provision of onsite support in critical cases as part of the training practice. Finally, the Training practice suggests the development of educational policies providing guidance for developing educational plans.

EM1: Handover Taxonomy suggests that Training Group be the main role responsible for executing the Training practice. Training planning, preparation of material and identification of trainers and trainees should take place in the development phase or, at the latest, in the system testing phase. The provision of training should take place at the latest in the acceptance testing phase.
Table 5.4. Training practice activities
The activity phases marked with black bars are the recommended phases for the activity at hand. The grey bars indicate that it is still possible to perform the activity but placing the activity in this phase may imply risks. Planning stands for Handover Planning space, Implementation stands for Handover Implementation space and Closure for Handover Closure space.

<table>
<thead>
<tr>
<th>Activities</th>
<th>Development</th>
<th>System Testing</th>
<th>Acceptance Testing</th>
<th>Deployment</th>
<th>Maintenance</th>
<th>Handover Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>T 1: Designate the role responsible for managing the training process</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td>T 2: Plan training</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td>T 2.1: Identify training topics to be taught (e.g. system, maintenance process, support process, technology, legal aspects)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td>T 2.2: Identify the trainee groups</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td>T 2.3: Determine training needs of each trainee group with respect to the training topics</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td>T 2.4: Define methods of training</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td>T 3: Create/update training material</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td>T 4: Identify the role responsible for providing the training</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td>T 5: Prepare for training</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td>T 5.1: Adapt the training material to the trainee group and its needs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td>T 5.2: Setup training environment, if required</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td>T 5: Provide training</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Implntion</td>
</tr>
<tr>
<td>T 6: Involve maintainers in attending to modification requests</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Implntion</td>
</tr>
<tr>
<td>T 7: Involve maintainers in white box testing and debugging</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Implntion</td>
</tr>
<tr>
<td>T 8: Provide onsite support, if needed</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Implntion</td>
</tr>
<tr>
<td>T 9: Develop educational policies providing guidance for developing educational plans</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td>T 10: Develop project specific educational plan using policy guidelines</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
</tbody>
</table>

5.6.5 Deployment

System deployment is the stage when a software system gets installed in its operational environment and its users start interacting with it. The interaction usually results in a flux of problem reports and modifications requests which either need to be urgently attended to during handover or have to be handed over to the postdelivery maintenance process.

System deployment is one of the critical stages in the lifecycle of a software system. Here, there is a strong need to determine and synchronize the responsibilities and work of the development and maintenance teams. Otherwise, the teams may get out of step with respect to the tasks of determining what is the scope of the deployed system, which version is under deployment, in which version changes should be made, who should make the
Table 5.5. Deployment practice activities

The activity phases marked with black bars are the recommended phases for the activity at hand. The grey bars indicate that it is still possible to perform the activity but placing the activity in this phase may imply risks. Planning stands for Handover Planning space, Implementation stands for Handover Implementation space and Closure for Handover Closure space.

<table>
<thead>
<tr>
<th>Activities</th>
<th>Development</th>
<th>System Testing</th>
<th>Acceptance Testing</th>
<th>Deployment</th>
<th>Maintenance</th>
<th>Handover Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>DP 1: Define/continuously re-define scope and content of the release</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td>DP 2: Determine type of release (major/minor)</td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DP 3: Create a deployment team</td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DP 4: Develop installation procedures</td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DP 4.1: Develop rollback procedures</td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DP 4.2: Develop installation manuals</td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DP 4.3: List organizations and stakeholders affected by the new release</td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DP 4.4: Prepare release and build documentation</td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DP 4.5: Define/continuously update the access rights to release components</td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DP 5: Installation</td>
<td></td>
<td></td>
<td></td>
<td>Implementation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DP 5.1: Take a backup of the system release to be de-installed</td>
<td></td>
<td></td>
<td></td>
<td>Implementation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DP 5.2: Perform deployment readiness test</td>
<td></td>
<td></td>
<td></td>
<td>Implementation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DP 5.3: Distribute and deliver the system and/or system components at a correct location and time</td>
<td></td>
<td></td>
<td></td>
<td>Implementation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DP 5.4: Install the new system version</td>
<td></td>
<td></td>
<td></td>
<td>Implementation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DP 5.5: Install operational data</td>
<td></td>
<td></td>
<td></td>
<td>Implementation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DP 5.6: Record any incidents, unexpected events, issue or deviations from the release plan</td>
<td></td>
<td></td>
<td></td>
<td>Implementation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DP 5.7: Perform deployment verification tests</td>
<td></td>
<td></td>
<td></td>
<td>Implementation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DP 6: Deployment Closure</td>
<td></td>
<td></td>
<td></td>
<td>Closure</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DP 6.1: Review the system deployment</td>
<td></td>
<td></td>
<td></td>
<td>Closure</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DP 6.2: Close the deployment</td>
<td></td>
<td></td>
<td></td>
<td>Closure</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DP 7: Planning of future releases</td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DP 7.1: Plan updates of future releases</td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DP 7.1.1: Identify features to be deployed in the next release</td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DP 7.2: Determine the impact of the externally acquired components on the planned release and vice versa, if relevant</td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DP 7.2.1: Estimate release size, effort, time and hardware/software infrastructure required</td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DP 7.2.2: Determine the system distribution structure</td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DP 7.3: Determine forms of deployment software</td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
changes, who are the customers of the deployed system, and on which version the customers should get service. Inability to fulfill those tasks strongly impacts the efficiency and effectiveness of the handover process. It may result in customer dissatisfaction, it may delay the maintenance services after deployment and, at its worst, it may result in system delivery failure [48] [49] [50] [51][12] [18].

The EM3 Deployment practice contains seven main activities and twenty sub-activities. As shown in Table 5.5, the activities focus on defining the scope and type of the release, on creating a deployment team, on developing installation procedures and on installing the system. It suggests a formal closure of the system deployment and planning for future releases.

The Deployment activities span across the whole system lifecycle. As shown in Table 5.5, planning for deployment of the next release takes place in parallel with the development of the current release whereas the deployment activities are more concentrated in the time span between acceptance testing, deployment and the initial part of maintenance phases. Regarding the roles, EM3 suggests that the responsibilities for managing and conducting deployment belong to the role of the Deployment Team which, in specific contexts, may be delegated to project manager, release manager, developer, maintainer, acquirer, and configuration manager.

5.6.6 Documentation

The Documentation practice mainly focuses on managing a system documentation repository and controlling mechanisms of its status. The goal of the practice is to store the software system documentation at a central location for assuring that no valuable information gets lost while handing over a software system. Not performing the Documentation activities leads to different issues during software system handover [48] [49] [50] [51] [57]. Some of them are:

- Loss of system knowledge, and thereby, difficulty in performing system maintenance after handover.
- Inaccessibility to the documentation repository due to access right restrictions, and thereby, difficulties in gaining appropriate system knowledge during handover.
- Confusion with respect to what to change in cases with the documentation is outdated or not consistent with the system status.
- Difficulties in understanding the documentation, and thereby, difficulties to understand the system due to lack of standards for documentation writing.

The Documentation practice comprises five main activities and seven sub-activities. As shown in Table 5.6, it includes activities for (1) establishing a system documentation repository and (2) for providing rudimentary services such as access, store, update, remove, grant rights, (3) for subjecting the documentation repository to SCM, (4) for developing documentation standards and mechanisms for controlling the status of the system repository, and finally, (5) it advocates the transfer of the documents to the maintainer during handover.
Table 5.6. Documentation practice activities

The activity phases marked with black bars are the recommended phases for the activity at hand. The grey bars indicate that it is still possible to perform the activity but placing the activity in this phase may imply risks. Planning stands for Handover Planning space, Implementation stands for Handover Implementation space and Closure for Handover Closure space.

<table>
<thead>
<tr>
<th>Activities</th>
<th>Development</th>
<th>System Testing</th>
<th>Acceptance Testing</th>
<th>Deployment</th>
<th>Maintenance</th>
<th>Handover Space</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>D 1:</strong> Establish a system documentation repository</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td><strong>D 2:</strong> Define services to be provided by the system documentation repository</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td><strong>D 2.1:</strong> Identify different types of services to be provided by the system documentation repository</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td><strong>D 2.2:</strong> Determine groups of access rights to the services</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td><strong>D 3:</strong> Subject system documentation repository to SCM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Implementation</td>
</tr>
<tr>
<td><strong>D 4:</strong> Establish documentation standards</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td><strong>D 4.1:</strong> Define organizational policies/rules/guidelines for developing documentation standards</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td><strong>D 4.2:</strong> Share documentation standards with the maintenance team during handover</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Implementation</td>
</tr>
<tr>
<td><strong>D 4.3:</strong> Develop templates standards for documentation according to the defined policies/rules/guidelines</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td><strong>D 4.4:</strong> Create rules for updating the system documentation repository</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td><strong>D 4.5:</strong> Create mechanisms for controlling the status of the system documentation repository</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td><strong>D 5:</strong> Transfer the documents from the documentation repository to maintainer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Implementation</td>
</tr>
</tbody>
</table>

The Documentation activities start in parallel with the development of the system and carry on throughout the system lifecycle. Documentation Manager is the main role responsible for the Documentation practice and he is assisted by Project Manager or Developer.

5.6.7 Maintainability Management

The Maintainability Management practice is concerned with the assessment of system and data maintainability of the system to be handed over. The maintainability is the ease with which one can make changes to the system and its data. The goal of the Maintainability Management practice is to assess the system and data maintainability for future maintenance. After system handover, the system needs to be enhanced with new features and corrected for software defects. Therefore, the system to be handed over should be assessed from the maintainability perspective. Ignoring the system and data maintainability may lead to the difficulties including the following:

- Difficulties in understanding the code due to its complexity and unreadability
- Difficulties in changing the system due to unstructured system architecture and its code
• Data inconsistency issues due to lack of restrictions on access rights
• Lack of system and data scalability due to inflexible data and complex system structures
• Lack of traceability between different system documents.

The above-listed difficulties become a hindrance for providing future maintenance services. The maintainer has to face various consequences due to the poor quality software produced by the developer. It may result in an increased cost, delay or even failure in completing customer requests.

The *Maintainability Management* practice contains two main activities and ten sub-activities. As shown in Table 5.7, the activities are concerned with the assessment of system and data for maintainability. They focus on defining attributes, rules and guidelines, and milestones for maintainability assessment. Finally, the taxonomy suggests assessing the system and data maintainability using the maintainability attributes [48] [49] [50] [51] [57].

The *Maintainability Management* practice begins in parallel with the development phase and continues throughout the whole lifecycle of the system. *Quality Assurance Team* is the main role responsible for the *Maintainability Management* practice. However, in different contexts, the responsibility may be delegated to *Developer, Maintainer, Tester* and *Database Administrator*.

**Table 5.7. Maintainability Management practice activities**

The activity phases marked with black bars are the recommended phases for the activity at hand. The grey bars indicate that it is still possible to perform the activity but placing the activity in this phase may imply risks. Planning stands for *Handover Planning* space, Implantation stands for *Handover Implementation* space and Closure for *Handover Closure* space.
5.6.8 Software System Transfer

The *Software System Transfer* practice manages the transfer of the system to be handed over. It includes transfer of software components, replication of data and modification requests. The goal is to assure that the right system components and the information about their modification needs are being transferred.

Lack of the *Software System Transfer* practice may lead to various problems. For instance, the transfer of unstable components will result in extra burden on the maintainer when correcting them in the future or unawareness of the status of the system and its components may make the maintainer accept the system that is not ready for transfer, and thereby, for future evolution and maintenance.

Overall, ignoring the *Software System Transfer* practice may result in chaos during the handover and initial maintenance phases. Stakeholders do not possess required information about the status of the system, and hence, their decisions to handover the system may correspond to an educated guess work only. They may discover the real system status too late, once the system is handed over for maintenance. At that moment, the maintenance team will have to face the consequences because the system becomes their responsibility, and irrespective of the system state, they have to attend to and accomplish customer change requests and problem reports.

The *Software System Transfer* practice consists of four main activities and eleven sub-activities. As shown in Table 5.8, the activities focus on monitoring the status of software components and identifying stable components, that is, the components under development and testing stages, and making decisions on handing over the system on the basis of the status of its components. To be able to monitor the system changes or the need for system changes, the practice suggests creating a template for managing information about modification requests and placing the modification requests in a modification request repository. The practice then suggests the transfer of the agreed upon software components, data, and modification requests to the transitionee. Finally, it advocates monitoring the system after the handover and signing off the software system handover after approval from all the parties concerned.

The *Software System Transfer* practice activities span across acceptance testing, deployment and maintenance phases. *Transition Team* is the main role responsible for *Software System Transfer* practice. However, their responsibilities may be delegated to the roles like *Project Manager*, *Maintainer* or *Developer*. 
The activity phases marked with black bars are the recommended phases for the activity at hand. The grey bars indicate that it is still possible to perform the activity but placing the activity in this phase may imply risks. Planning stands for Handover Planning space, Implantion stands for Handover Implementation space and Closure for Handover Closure space.

<table>
<thead>
<tr>
<th>Activities</th>
<th>Development</th>
<th>System Testing</th>
<th>Acceptance Testing</th>
<th>Deployment</th>
<th>Maintenance</th>
<th>Handover Space</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ST1:</strong> Monitor status of software components</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Implantion</td>
</tr>
<tr>
<td><strong>ST1.1:</strong> Identify stable software components ready to be used in the system to be handed over</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Implantion</td>
</tr>
<tr>
<td><strong>ST1.2:</strong> Identify software components under testing stage</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Implantion</td>
</tr>
<tr>
<td><strong>ST1.3:</strong> Identify software components under development stage</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Implantion</td>
</tr>
<tr>
<td><strong>ST2:</strong> Make decision on the components to be handed over</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Implantion</td>
</tr>
<tr>
<td><strong>ST3:</strong> Manage modification requests</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Implantion</td>
</tr>
<tr>
<td><strong>ST3.1:</strong> Create a template for managing information about modification requests and their management</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Implantion</td>
</tr>
<tr>
<td><strong>ST3.2:</strong> Place modification requests in a modification request repository</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Implantion</td>
</tr>
<tr>
<td><strong>ST3.3:</strong> Use modification requests to revise the handover decision</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Implantion</td>
</tr>
<tr>
<td><strong>ST4:</strong> System transfer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Implantion</td>
</tr>
<tr>
<td><strong>ST4.1:</strong> Transfer the agreed upon software components</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Implantion</td>
</tr>
<tr>
<td><strong>ST4.2:</strong> Transfer the replica of the operational data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Implantion</td>
</tr>
<tr>
<td><strong>ST4.3:</strong> Transfer modification requests</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Implantion</td>
</tr>
<tr>
<td><strong>ST4.4:</strong> Monitor the system after handover</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Closure</td>
</tr>
<tr>
<td><strong>ST4.5:</strong> Signoff the handover closure</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Closure</td>
</tr>
</tbody>
</table>
6. Epilogue

Software system handover is a strongly neglected research area. Both the academia and industry have paid little attention to formulate process models of software system handover. Today, there exist very few process models that can be used as a reference for transitioning software systems from development to maintenance. These models are either outdated or too general. Hence, they are not very supportive of their mission.

The goal of this thesis was to explore the software system handover domain and establish a basis for creating industrial software system handover process models. The subjects of exploration were different types and contexts of handover, the important handover activities, the major participating roles, and the points in time in the software system lifecycle in which software handover activities take place.

The scarcity of the existing knowledge about software system handover has left us with no other choice than to elicit knowledge within the industry. Moreover, since one of our goals was to aid industry with creating handover process models, it felt natural to explore, evolve, and evaluate the industrial knowledge. Only such knowledge is practical in nature and can be utilized within the industry.

6.1 Contribution

The contribution of this thesis is a framework called EM³: Handover Framework. The reason, why we created a framework and not a process model is the fact that software system handover is one of the most diverse processes whose design is strongly dependent on a handover case. Therefore, it may be difficult to create a generic software system handover process model. We believe that it is better to just create a frame of parts which, when fitted and joined together, may be orchestrated into specific process models of software system handover.

The framework consists of six parts. These are (1) EM³: Handover Taxonomy, (2) Handover Types, (3) Handover Contexts, (4) Handover Roles, (5) Handover Lifecycle roadmap and (6) Handover Guidelines. Its central part however is EM³: Handover Taxonomy including eight component practices which, in turn, include the activities that are conducted in software system handover. The other parts include Handover Roles and Handover Lifecycle Roadmap suggesting the major roles and suitable points in time in the system lifecycle for implementing the activities, and Handover Guidelines providing suggestions and explanations required for defining handover process models. Using these parts, and depending on the handover type and handover context at hand, the organizations can adapt EM³: Handover Framework to formulate their own handover process models.

It is worthwhile to mention that it is not compulsory to implement each and every practice and activity during handover. Depending on the handover context and type, the organizations can focus on choosing the parts that are most suitable for their own purposes. However, two component practices, Management and Administration and Software
System Transfer are the core practices and must always be implemented as part of every software system handover process.

We believe that we have provided a substantial input to the software community with respect to the increased knowledge about software system handover and improved visibility into the current state of art and practice. Table 6.1 provides a high-level comparison of our work with the works of the others. By studying the table, we dare to claim the following:

- **EM**: Handover Framework is the only work that considers three handover types. No other work has clearly done this. Pigoski’s work, however does this indirectly by mentioning two types of handover via (1) cradle-to-grave system management and (2) transition to another party.
- **EM**: Handover Framework is the only work that identifies different handover contexts using different lifecycle roadmaps. In this way, it provides more insight into and understanding of the different ways of placing the handover process within software system lifecycle.
- Only three out of five works identify the roles involved in software system handover. These are Vollman and Pigoski’s works and EM: Handover Framework. However, EM: Handover Framework takes a different approach with respect to what role does what and when. It does not impose any specific responsibilities to specific roles. This would make the framework too prescriptive considering the fact that the roles involved in handover vary too much depending on different contexts, size of software systems to be handed over and number of organizations involved. The framework gives the organizations freedom to be able to adapt themselves to their specific scenarios. Even the roles that are central for a specific component practice are not fixed. Their overall responsibilities or parts of them may be delegated to other roles.
- Only EM: Handover Framework clearly identifies all the practices that play the central role within software system handover. Vollman and Pigoski’s works do it only partially and indirectly; Pigoski’s work by identifying transition phases and Vollman’s work by identifying the categories of Software Support Activity function. These works, however, constituted a basis for defining the EM: Handover Framework practices.
- All except for one work define handover activities. EM: Handover Framework, however, defines a comprehensive set of activities. Pigoski, Vollman and ISO/IEC 15288 only identify rudimentary activities and they do not relate them to any practices whatsoever. ISO IEC 14764 identifies only predelivery and postdelivery maintenance activities but no transition activities. In all these works, it is not clear what happens when and why.
- Only EM: Handover Framework defines a software handover lifecycle and suggests which activities can be placed in which handover lifecycle phases. The framework also suggests a lifecycle span for placing the activities. To the knowledge of the author of this thesis, no other work has done it.
- Finally, only EM: Handover Framework provides guidelines that are important for handover implementation. No such guidelines are available in the other works whatsoever.
Table 6.1 Research contribution.
✓ stands for “exists” -- stands for “not exists", P stands for “partially exists"

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Handover types</td>
<td>✓</td>
<td>--</td>
<td>P</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>Handover contexts</td>
<td>✓</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>Handover roles</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>Handover practices</td>
<td>✓</td>
<td>P</td>
<td>P</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>Handover activities</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>P</td>
<td>✓</td>
</tr>
<tr>
<td>Handover lifecycle</td>
<td>✓</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>Handover guidelines</td>
<td>✓</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
</tbody>
</table>

6.2 Future work

EM³: Handover Framework opens a wide range of future research directions. When creating EM³: Handover Framework, we strived towards creating a generic and easy to understand and utilize solution. We did not focus on adapting to any specific method or technology. The framework was created by analyzing the data collected from organizations which were using different methods for developing and maintaining software systems. Therefore, we dare say that the framework is generic enough so that it may be used in most of the industrial contexts. It should be however further explored from the perspective of different methodological or technological approaches such as traditional, agile, global and open source development and cloud computing. Each of such methods or technologies has its own challenges for transitioning software systems, and therefore, there is a need for conducting more research to tackle the challenges in their respective areas.

EM³: Handover Framework comprises eight component practices important for handover. The practice activities were evolved by ensuring that they were generic, easy to understand and easy to utilize. Each practice, however, on its own constitutes a vast and comprehensive process and research area. There is a need to further explore each practice in more detail under different software system handover contexts. For example, the Training practice should be examined in more detail from the knowledge management perspective [74] [75] [76] [77] [78].

Different types of handover take place under different scenarios. We focused majorly on the transition from one party (transitioner) to another (transitionee). However, to better explore the software system handover domain, more work needs to be done on exploring different scenarios of different handover types. There is also a need to study the scenarios in open source communities and in global software development and acquisition [79] [80] [75] [81] [82] [83].

Finally, handover process is closely related to the pre-delivery maintenance process. The goal of the pre-delivery maintenance process is to prepare the system and future maintenance organization for maintenance whereas the goal of the handover process is to transfer the system and responsibilities for maintaining the system. The transitionee team cannot perform their duties if they do not prepare themselves and their systems for future
maintenance. In this way, both pre-delivery maintenance and handover processes are strongly interrelated and interlinked.

Due to the magnitude and complexity of the predelivery maintenance and its intricate relation to the handover process, we have not been able to explore their relationship in greater detail. We have however done some work by evaluating *EM*: *Predelivery Maintenance* and by identifying its contexts [84] [85]. Hence, we have created a basis for co-relating the two processes, so that they may better contribute to a seamless and successful software system handover.
Bibliography


[Accessed 2013].


Sciences (DSV), Stockholm University and Royal Institute of Technology, 2001.


Part II
Appendices
Appendix A

EM³: Handover Taxonomy versions

Appendix A.1

EM³: Handover Taxonomy, version 1

This appendix presents the first version of EM³: Handover Taxonomy. It was developed by eliciting and classifying activities from literature in the Initial Study phase. This version consists of 53 activities including 34 main activities and 19 sub-activities.

### Management and Administration (MA)

**MA 1.** Identify maintenance organization.
**MA 2.** Establish a transition team.
**MA 3.** Create a transition plan.
**MA 3.1.** Determine transition budget.
**MA 3.2.** Create a transition schedule.
**MA 3.3.** Establish transition procedures.
**MA 3.4.** Define transition resource requirements.
**MA 3.4.1.** Define maintenance manpower requirements.
**MA 3.4.2.** Define maintenance facility requirements.
**MA 3.5.** Define other transition elements (not covered in this study)
**MA 4.** Develop management plans.

### Maintenance Environment (ME)

**ME 1.** Determine hardware/software suite needs.
**ME 2.** Install hardware/software suite.
**ME 3.** Assess current hardware/software suite, if any.
**ME 4.** Remedy the deficiencies in the hardware/software suite, if any.
**ME 5.** Determine/assess maintenance support suite.
**ME 6.** Supplement maintenance support suite with new tools.
**ME 7.** Install support software.
**ME 8.** Install software baseline.
**ME 9.** Install data.
**ME 10.** Transfer modification requests from development to maintenance.
**ME 11.** Place modification requests in a Modification Request repository.

### Version and Configuration Management (VC)

**VC 1.** Establish software configuration baseline.
**VC 2.** Put software under software configuration management.
**VC 3.** Place software under version control.

### Deployment (DP)

**DP 1.** Develop installation procedures.
**DP 2.** Install.
**DP 3.** Plan future releases.
**DP 3.1.** Plan updates of future releases.
**DP 3.2.** Determine the distribution structure.
**DP 3.3.** Determine forms of deploying software.
**DP 3.4.** Determine the structure of release notes.

### Training (T)

**T 1.** Train on system, its structure and operation.
**T 1.1.** Create/update training material on system, its structure and operation.
**T 1.2.** Provide training on system, its structure and operation.
**T 1.3.** Attend to modification requests.
**T 1.4.** Conduct white-box testing and debugging.
**T 2.** Train on maintenance processes.
**T 2.1.** Create/update training material on maintenance processes.
**T 2.2.** Provide training on maintenance processes.
**T 3.** Train on system support processes.
**T 3.1.** Create/update training material on system support process.
**T 3.2.** Provide training material on system support process.
**T 4.** Train on new technology to be used within operation and maintenance.
**T 4.1.** Create/update training material on new technology.
**T 4.2.** Provide training material on new technology.
**T 5.** Provide on-site support in critical cases.

### Documentation (D)

**D 1.** Establish a system documentation repository.
**D 2.** Define services to be provided by the system documentation repository.
**D 3.** Subject system documentation repository to SCM.
**D 4.** Establish standards for documentation development.

### Maintainability Management (MM)

**MM 1.** Assess system maintainability.
**MM 2.** Assess data maintainability.
Appendix A.2

EM^3^: Handover Taxonomy, version 2

This appendix presents the second version of EM^3^: Handover Taxonomy. This version was used for Large-Scale Exploration phase. This version consists of 70 activities including 32 main activities and 38 sub activities.

Management and Administration

**MA 1.** Identify maintenance organization.
**MA 2.** Establish a transition team.
**MA 3.** Create a transition plan.
- **MA 3.1.** Determine transition budget.
- **MA 3.2.** Create a transition schedule.
- **MA 3.3.** Establish transition procedures.
- **MA 3.4.** Define transition resource requirements.
- **MA 3.4.1.** Define maintenance manpower requirements
- **MA 3.4.2.** Define maintenance facility requirements

**MA 4.** Develop management plans

Maintenance Environment

**ME 1.** Determine hardware/software suite needs.
**ME 2.** Install hardware/software suites.
**ME 3.** Assess current hardware/software suites, if any.
**ME 4.** Remedy the deficiencies in the hardware/software, if any.
**ME 5.** Determine/assess maintenance support suite.
**ME 6.** Supplement maintenance support suite with new tools.
**ME 7.** Install support software.
**ME 8.** Install software baseline.
**ME 9.** Install data.
**ME 10.** Transfer modification requests from development to maintenance.
**ME 11.** Place modification requests in a Modification Request repository.

Version and Configuration Management

**VC 1.** Establish software configuration baseline.
- **VC 1.1.** Define how the types of configuration items are to be selected, grouped and classified.
**VC 1.2.** Establish system test baseline.
**VC 1.3.** Establish acceptance test baseline.
**VC 1.4.** Establish operational baseline.
**VC 2.** Put software under software configuration management.
**VC 3.** Place software under version control.
- **VC 3.1.** Define the approaches to identification, uniquely naming and labeling all the assets and the relationship between them.
- **VC 3.2.** Assign unique identifier to configuration items.
Deployment

DP 1. Develop installation procedures.
DP 1.1. Define scope and content of the release.
DP 1.2. List organizations and stakeholders affected by the release.
DP 1.3. Identify team responsible for release.
DP 1.4. Prepare release and build documentation.
DP 1.5. Define the access rights to physical and technology components.
DP 1.6. Develop procedures to back out release unit in case the release fails.

DP 2 Install.
DP 2.1. Baseline contents of a release package
DP 2.2. Perform deployment readiness test.
DP 2.3. Distribute and deliver the system and /or system components at correct location and time.
DP 2.4. Build, install and configure the system.
DP 2.5. Record any incidents, unexpected events, issue or deviations from the plans.
DP 2.6. Perform deployment verification tests.
DP 2.7. Review and close the deployment

DP 3. Plan future releases
DP 3.1. Plan updates for future releases
DP 3.2. Determine the distribution structure.
DP 3.3. Determine forms of deployment software.

Training

T 1. Train on system, its structure and operation.
T 1.1. Create/update training material on system, its structure and operation.
T 1.2. Provide training on system, its structure and operation.
T 1.3. Attend to modification requests.
T 1.4. Conduct white box testing and debugging.
T 2. Train on maintenance processes.
T 2.1. Create/update training material on maintenance process.
T 2.2. Provide training on maintenance process.
T 3. Train on system support processes.
T 3.1 Create/update training material on system support process.
T 3.2. Provide training on system support process.
T 4. Train on new technology to be used within operation and maintenance.
T 4.1. Create/update training material on new technology.
T 4.2. Provide training on new technology.
T 5. Provide on-site support in critical cases.

Documentation

D 1. Establish a system documentation repository.
D 2. Define services to be provided by the system documentation repository.
D 3. Subject system documentation repository to SCM.
D 4. Establish standards for documentation development.

Maintainability Management

M 1. Assess system maintainability.
M 2. Assess data maintainability.
This appendix presents the third version of EM³: Handover Taxonomy. This version was evolved as a result of Large-Scale Exploration study. This version consists of 154 activities including 39 main activities and 115 sub activities.

**Management and Administration activities**

**MA 1:** Determine/re-determine type of transition

**MA 2:** Define a strategy for transition process

**MA 3:** Define a transition process

- **MA 3.1:** Identify core transition activities
- **MA 3.2:** Identify activities that are part of processes that either impact or are impacted by transition

**MA 4:** Designate a maintenance organization

**MA 5:** Agree upon the executed transition process

**MA 6:** Create/adjust a transition plan

- **MA 6.1:** Define/adjust parameters guiding the design of the transition plan
- **MA 6.2:** Create the transition plan using the parameters.
- **MA 6.3:** Define transition resource requirements
  - **MA 6.3.1:** Define manpower requirements
    - **MA 6.3.1.1:** Define maintenance manpower requirements
    - **MA 6.3.1.2:** Define developer manpower resources
    - **MA 6.3.1.3:** Define transition group manpower resources, if any
    - **MA 6.3.1.4:** Other manpower resources, if any
  - **MA 6.3.2:** Define maintenance facility requirements

**MA 7:** Develop management plans

- **MA 7.1:** Determine management plans necessary for transition process

**MA 7.2:** Determine a communication model (see the core transition problems (should include oral communication and documentation)

- **MA 7.2.1:** Determine communication requirements (written & oral communication)
- **MA 7.2.2:** Determine communication channels
- **MA 7.2.3:** Determine change notification process

**MA 8:** Evaluate the transition process

**Maintenance Environment**

**ME 1.1:** Hardware/Software suite needs

- **ME 1.1.1:** Determine hardware and software packages constituting the hardware/software suite.
**ME 1.1.2:** Assure that hardware/software suit needs match the developer’s hardware/software suite

**ME 1.1.3:** Assure that hardware/software suit needs match the customer’s hardware/software suite.

**ME 1.2:** Install hardware/software suites.

**ME 1.2.1:** Grant access permission to the maintenance team members

**ME 1.3:** Assess current hardware/software suites, if any

**ME 1.3.1:** Identify deficiencies in the hardware/software suites, if any

**ME 1.3.2:** Evaluate hardware/software suite needs for the next release.

**ME 1.4:** Remedy the deficiencies in hardware/software suite needs, if any

**ME 1.4.1:** Procure new hardware/software suites, if needed

**ME 1.4.2:** Update the hardware/software suites with the tools needed.

**ME 2. Maintenance support suites**

**ME 2.1:** Determine maintenance support suite

**ME 2.1.1:** Determine software packages constituting the maintenance support suite.

**ME 2.1.2:** Assure that the maintenance support suite matches the developer’s hardware/software suite.

**ME 2.1.3:** Assure that the maintenance support suite matches the customer’s hardware/software suite.

**ME 2.2:** Assess maintenance support suite.

**ME 2.2.1:** Identify deficiencies in the maintenance support suites, if any

**ME 2.2.2:** Evaluate the maintenance support suite needs for the next release.

**ME 2.3:** Install maintenance support suite.

**ME 2.4:** Supplement the maintenance support suite with new tools.

**ME 3: Software Baseline**

**ME 3.1:** Install software maintenance baseline.

**ME 3.1.1:** Assure that the software baseline is identical to the baseline installed on the developer and customer’s site.

**ME 3.1.2:** Synchronize the software maintenance baseline with new modifications, if any.

**ME 4: Operational Data Baseline**

**ME 4.1:** Install the replica of the operational data baseline.

**ME 4.2:** Update the replica of the operational data baseline on a regular or as-needed basis.

**ME 5: Modification Requests**

**ME 5.1:** Create a template for managing information about modification requests and their management.

**ME 5.2:** Place modification requests in a modification request repository.

**ME 5.3:** Grant access rights to the maintenance team to the common modification request repository.

**ME 5.4:** Assign modification requests to the maintainer for resolution during handover.

**ME 5.5:** Keep track of the status of each modification request during handover.
Version and Configuration Management

**VC 1:** Version and configuration management
- **VC 1.1:** Define rules to uniquely identify, name and label the configuration items and their relationships.
- **VC 1.2:** Define how the configuration items are to be selected, grouped and classified.
- **VC 1.3:** Decide on how to identify and track changes made to customizable configuration items during handover.
- **VC 1.4:** Put software under configuration management
- **VC 1.5:** Place software under version control management

**VC 2:** Manage baselines
- **VC 2.1:** Establish test baselines (system test baseline, acceptance test baseline)
  - **VC 2.1.1:** Assist developers in attending to problem reports during acceptance testing.
  - **VC 2.1.2:** Identify and track customizable configuration items during handover.
  - **VC 2.1.3:** Keep a track of the changes made during the testing of the baselines.
  - **VC 2.2.4:** Notify the changes made to the system to all the stakeholders involved.
- **VC 2.2:** Establish post-delivery baselines (operational baseline and maintenance baseline)
  - **VC 2.2.1:** Check whether the reported problems (if any) are not of critical nature.
  - **VC 2.2.2:** Synchronize system changes made during system handover in all the environments (operational, development and maintenance)
  - **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.
  - **VC 2.2.4:** Accept and approve the system for operation.

Training

**T 1:** Designate the role responsible for managing the training process

**T 2:** Plan training
- **T 2.1:** Identify training subjects (e.g. system, maintenance process, support process, technology)
- **T 2.2:** Identify the trainee groups
- **T 2.3:** Determine training needs of each trainee group with respect to the training subjects
- **T 2.4:** Define methods of training

**T 3:** Create/update training material

**T 4:** Identify the role responsible for providing training

**T 5:** Prepare for training
- **T 5.1:** Adapt the training material to the trainee group and its needs
- **T 5.2:** Setup training environment, if required

**T 6:** Provide training

**T 7:** Involve maintainers in attending to modification requests

**T 8:** Involve maintainers in white box testing and debugging

**T 9:** Provide onsite support in critical cases
T 10: Develop policies providing guidance for developing education plans
T 11: Develop project specific plan using policy guidelines

**Deployment**

**DP 1:** Release scope and content
  - **DP 1.1:** Define/continuously re-define scope and content of the release.
  - **DP 1.2:** Determine type of release (major/minor).

**DP 2:** Installation procedures
  - **DP 2.1:** Create a team responsible for deploying the release
  - **DP 2.2:** Develop installation procedures
    - **DP 2.2.1:** Develop rollback procedures.
    - **DP 2.2.2:** Develop installation manuals.
  - **DP 2.3:** List organizations and stakeholders affected by the new release.
  - **DP 2.4:** Prepare release and build documentation
  - **DP 2.5:** Define/continuously update the access rights to system components.

**DP 3:** Installation
  - **DP 3.1:** Take a backup of the system version to be de-installed.
  - **DP 3.3:** Perform deployment readiness test.
  - **DP 3.4:** Distribute and deliver the system and/or system components at the correct location and time.
  - **DP 3.5:** Install the new system version.
  - **DP 3.6:** Install operational data.
  - **DP 3.7:** Record any incidents, unexpected events, issue or deviations from the release plan.
  - **DP 3.8:** Perform deployment verification tests.

**DP 4:** Deployment Closure
  - **DP 4.1:** Review the system deployment.
  - **DP 4.2:** Close the deployment.

**DP 5:** Planning of future releases
  - **DP 5.1:** Plan updates for future releases.
    - **DP 5.1.1:** Identify features to be deployed in the next release.
    - **DP 5.1.2:** Determine the impact of the COTS components on the planned release and vice versa, if relevant.
    - **DP 5.1.3:** Estimate release size, effort, time and infrastructure required.
  - **DP 5.2:** Determine the distribution structure.
  - **DP 5.3:** Determine forms of deployment software.

**Documentation**

**D 1:** Establish a system documentation repository
**D 2:** Define services to be provided by the system documentation repository
  - **D 2.1:** Identify different types of services
  - **D 2.2:** Determine groups of access rights to the services

**D 3:** Identify communication channels for eliciting information from software engineers

**D 4:** Subject system documentation repository to SCM

**D 5:** Establish documentation standards
  - **D 5.1:** Define organizational level policies/rules/guidelines for developing documentation standards
D 5.2: Share documentation standards with maintenance team during handover
D 5.3: Develop templates for documentation according to defined policies/rules/guidelines
D 5.4: Create rules for updating the system documentation repository
D 5.5: Create mechanisms for controlling the status of the system documentation repository
   D 5.5.1: Create mechanisms for controlling the quality of the updated system documentation
   D 5.5.2: Create notification mechanisms
      D 5.5.2.1: Create notification mechanisms for notifying about missed deadlines
      D 5.5.2.2: Create notification mechanisms for notifying about the non-adherence to the standards.
D 6: Transfer the documents from the documentation repository to maintainer

Maintainability Management

MM 1: Assess system maintainability
   MM 1.1: Define system maintainability attributes
   MM 1.2: Define rules and guidelines for adhering to the system maintainability
   MM 1.3: Identify milestones for assessing system maintainability.
   MM 1.4: Assess system maintainability using the system maintainability attributes
   MM 1.5: Assess procedures for managing and controlling system maintainability

MM 2: Assess data maintainability
   MM 2.1: Define data maintainability attributes
   MM 2.2: Define rules and guidelines for adhering to the data maintainability
   MM 2.3: Identify milestones for assessing data maintainability.
   MM 2.4: Assess data maintainability using the data maintainability attributes
   MM 2.5: Assess procedures for managing and controlling data maintainability.
Appendix A.4

EM³: Handover Taxonomy, version 4
(Final version)

This appendix presents the final version of EM³: Handover Taxonomy after evaluating and evolving it in the Final Study phase. This version consists of 139 activities including 44 main activities and 95 sub activities.

**Management and Administration**

**MA 1:** Determine/redetermine type and complexity of transition  
**MA 2:** Define a strategy for transition process  
**MA 3:** Designate a transitionee  
**MA 4:** Establish a transition team  
**MA 5:** Define a transition process  
  **MA 5.1:** Identify core transition activities  
  **MA 5.2:** Identify activities that are part of the processes that either impact or are impacted by the transition  
**MA 6:** Agree upon the executed transition process  
**MA 7:** Create/adjust a transition plan  
  **MA 7.1:** Define/adjust parameters guiding the design of the transition plan  
  **MA 7.2:** Create the transition plan using the parameters  
  **MA 7.3:** Define transition resource requirements  
    **MA 7.3.1:** Define manpower requirements  
      **MA 7.3.1.1:** Define maintenance manpower requirements  
      **MA 7.3.1.2:** Define developer manpower resources  
      **MA 7.3.1.3:** Define transition team manpower resources, if any  
      **MA 7.3.1.4:** Define other manpower resources, if any  
    **MA 7.3.2:** Define maintenance facility requirements  
  **MA 7.4:** Determine transition budget  
  **MA 7.5:** Create a transition schedule  
**MA 8:** Develop management plans necessary for transition  
**MA 9:** Determine a communication model to be used within transition  
**MA 10:** Monitor the transition process  
**MA 11:** Evaluate the transition process

**Maintenance Environment**

**ME 1:** Manage hardware/software suite needs  
  **ME 1.1:** Determine hardware/software suite needs  
    **ME 1.1.1:** Determine hardware and software packages constituting the hardware/software suite  
    **ME 1.1.2:** Assure that hardware/software suite needs match the developer’s hardware/software suite
ME 1.3: Assist that hardware/software suite needs match the customer’s hardware/software suites
ME 1.2: Install hardware/software suite
ME 1.1: Grant the transitionee access permission to hardware/software suite
ME 1.4: Assess current hardware/software suite, if any
ME 1.5: Remedy the deficiencies in hardware/software suite needs, if any

ME 2: Manage maintenance support suite
ME 2.1: Determine maintenance support suites
ME 2.2: Install maintenance support suite
ME 2.3: Assess maintenance support suite
ME 2.4: Remedy the deficiencies in support suite, if any

Version and Configuration Management

VC 1: Manage version and configuration
VC 1.1: Define rules to uniquely identify, name and label the configuration items and their relationships
VC 1.2: Define how the configuration items are to be selected, grouped and classified
VC 1.3: Decide on how to identify and track changes made to customizable configuration items during handover
VC 1.4: Put software under configuration management
VC 1.5: Place software under version control management

VC 2: Manage baselines
VC 2.1: Establish test baselines (system test baseline, acceptance test baseline)
VC 2.1.1: Assist developers in attending to problem reports during acceptance testing
VC 2.1.2: Identify and track customizable configuration items during handover
VC 2.1.3: Keep track of the changes made to the baselines
VC 2.1.4: Notify about the changes made to the system to all the stakeholders involved
VC 2.2: Establish post-delivery baselines (operational baseline and maintenance baseline)
VC 2.2.1: Check whether the reported problems are not of critical nature
VC 2.2.2: Synchronize system changes made during system handover in all the environments (operational, development and maintenance)
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
VC 2.2.4: Accept and approve the system for operation and maintenance
Training

T 1: Designate the role responsible for managing the training process

T 2: Plan training
  T 2.1: Identify training topics to be taught (e.g. system, maintenance process, support process, technology, legal aspects)
  T 2.2: Identify the trainee groups
  T 2.3: Determine training needs of each trainee group with respect to the training topics
  T 2.4: Define methods of training

T 3: Create/update training material

T 4: Identify the role responsible for providing the training

T 5: Prepare for training
  T 5.1: Adapt the training material to the trainee group and its needs
  T 5.2: Setup training environment, if required

T 6: Provide training

T 7: Involve maintainers in attending to modification requests

T 8: Involve maintainers in white box testing and debugging

T 9: Provide onsite support, if needed

T 10: Develop educational policies providing guidance for developing educational plans

T 11: Develop project specific educational plan using policy guidelines

Deployment

DP 1: Define/continuously re-define scope and content of the release

DP 2: Determine type of release (major/minor)

DP 3: Create a deployment team

DP 4: Develop installation procedures
  DP 4.1: Develop rollback procedures
  DP 4.2: Develop installation manuals
  DP 4.3: List organizations and stakeholders affected by the new release
  DP 4.4: Prepare release and build documentation
  DP 4.5: Define/continuously update the access rights to release components

DP 5: Installation
  DP 5.1: Take a backup of the system release to be de-installed
  DP 5.2: Perform deployment readiness test
  DP 5.3: Distribute and deliver the system and /or system components at a correct location and time
  DP 5.4: Install the new system version
  DP 5.5: Install operational data
  DP 5.6: Record any incidents, unexpected events, issue or deviations from the release plan
  DP 5.7: Perform deployment verification tests
DP 6: Deployment Closure
   DP 6.1: Review the system deployment
   DP 6.2: Close the deployment

DP 7: Planning of future releases
   DP 7.1: Plan updates of future releases
      DP 7.1.1: Identify features to be deployed in the next release
      DP 7.1.2: Determine the impact of the externally acquired components on
                  the planned release and vice versa, if relevant
      DP 7.1.3: Estimate release size, effort, time and hardware/software
                  infrastructure required
   DP 7.2: Determine the system distribution structure
   DP 7.3: Determine forms of deployment software

Documentation

D 1: Establish a system documentation repository
D 2: Define services to be provided by the system documentation repository
   D 2.1: Identify different types of services to be provided by the system
           documentation repository
   D 2.2: Determine groups of access rights to the services
D 3: Subject system documentation repository to SCM
D 4: Establish documentation standards
   D 4.1: Define organizational policies/rules/guidelines for developing
           documentation standards
   D 4.2: Share documentation standards with the maintenance team during handover
   D 4.3: Develop templates for documentation according to the defined
           policies/rules/guidelines
   D 4.4: Create rules for updating the system documentation repository
   D 4.5: Create mechanisms for controlling the status of the system documentation
           repository
D 5: Transfer the documents from the documentation repository to maintainer

Maintainability Management

MM 1: Assess system maintainability
   MM 1.1: Define system maintainability attributes
   MM 1.2: Define rules and guidelines for adhering to the system maintainability
   MM 1.3: Identify milestones for assessing system maintainability
   MM 1.4: Assess system maintainability using the system maintainability attributes
   MM 1.5: Assess procedures for managing and controlling system maintainability

MM 2: Assess data maintainability
   MM 2.1: Define data maintainability attributes
   MM 2.2: Define rules and guidelines for adhering to the data maintainability
   MM 2.3: Identify milestones for assessing data maintainability
   MM 2.4: Assess data maintainability using the data maintainability attributes
**Software System Transfer**

**ST 1: Monitor the status of software components**
- **ST 1.1:** Identify stable software components ready to be used in the system to be handed over
- **ST 1.2:** Identify software components under testing stage
- **ST 1.3:** Identify software components under development stage

**ST 2: Make decision on the components to be handed over**

**ST 3: Modification Requests**
- **ST 3.1:** Create a template for managing information about modification requests and their management
- **ST 3.2:** Place modification requests in a modification request repository
- **ST 3.3:** Use modification requests as support for making handover decisions

**ST 4: System transfer**
- **ST 4.1:** Transfer the agreed upon software components
- **ST 4.2:** Transfer the replica of the operational data
- **ST 4.3:** Transfer modification requests
- **ST 4.4:** Monitor the system after handover
- **ST 4.5:** Signoff the handover closure

**MM 2.5:** Assess procedures for managing and controlling data maintainability
### Appendices B

#### Questionnaires

#### Appendix B.1

**Questionnaire used for demarcating the scope of handover process**

This appendix presents the questionnaire used in the *Initial Study* phase for identifying the scope of handover process.

<table>
<thead>
<tr>
<th>No</th>
<th>Questions</th>
<th>No</th>
<th>Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><strong>Respondents and their organizations</strong></td>
<td>(iii)</td>
<td>Could you pointout a transition phase on the <em>Lifecycle Roadmap 3</em>?</td>
</tr>
<tr>
<td>(i)</td>
<td>The interviewee personal data</td>
<td>(i)</td>
<td>What groups/organizations are involved in transition?</td>
</tr>
<tr>
<td>(a)</td>
<td>What is your name?</td>
<td>(a)</td>
<td>Developer?</td>
</tr>
<tr>
<td>(b)</td>
<td>What is your email?</td>
<td>(b)</td>
<td>Maintainer?</td>
</tr>
<tr>
<td>(c)</td>
<td>What is your telephone number?</td>
<td>(c)</td>
<td>Acquirer?</td>
</tr>
<tr>
<td>(ii)</td>
<td>What is name of your organization?</td>
<td>(d)</td>
<td>List anyother groups/ organizations</td>
</tr>
<tr>
<td>(iii)</td>
<td>What is number of employees at organization?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(iv)</td>
<td>What is your role within the organization?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td><strong>Process usage and terminology</strong></td>
<td>(ii)</td>
<td>What activities are included in the transition phase?</td>
</tr>
<tr>
<td>(i)</td>
<td>Does your organization perform transition activity?</td>
<td>(a)</td>
<td>On the developer’s side?</td>
</tr>
<tr>
<td>(ii)</td>
<td>What term does your organization use to refer to transition?</td>
<td>(b)</td>
<td>On the maintainer’s side?</td>
</tr>
<tr>
<td>3</td>
<td><strong>Placement within lifecycle</strong></td>
<td>(c)</td>
<td>On the acquirer’s side?</td>
</tr>
<tr>
<td>(i)</td>
<td>Could you pointout a transition phase on the <em>Lifecycle Roadmap 1</em>?</td>
<td>(d)</td>
<td>List the activities performed by any other groups/organizations?</td>
</tr>
<tr>
<td>(a)</td>
<td>From major version to major version n+1?</td>
<td>(iii)</td>
<td>What roles are involved in the transition phase?</td>
</tr>
<tr>
<td>(b)</td>
<td>From major version to revision n.1?</td>
<td>(a)</td>
<td>Roles on the developer’s side?</td>
</tr>
<tr>
<td>(c)</td>
<td>Other, please specify?</td>
<td>(b)</td>
<td>Roles on the maintainer’s side?</td>
</tr>
<tr>
<td>(ii)</td>
<td>Could you pointout a transition phase on the <em>Lifecycle Roadmap 2</em>?</td>
<td>(c)</td>
<td>Roles on the acquirer’s side?</td>
</tr>
<tr>
<td>(a)</td>
<td>From new development to evolution?</td>
<td>(d)</td>
<td>List other roles in any other groups/organizations?</td>
</tr>
<tr>
<td>(b)</td>
<td>Other, please specify?</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Questionnaire used for identifying handover problems

This appendix presents the questionnaire used during *Initial Study* phase for identifying the handover problems.

<table>
<thead>
<tr>
<th>No</th>
<th>Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Interviewee personal data questions</td>
</tr>
<tr>
<td></td>
<td>(i) What is your name?</td>
</tr>
<tr>
<td></td>
<td>(ii) What is your email?</td>
</tr>
<tr>
<td></td>
<td>(iii) What is your telephone number?</td>
</tr>
<tr>
<td>2</td>
<td>Organization basic information questions</td>
</tr>
<tr>
<td></td>
<td>(i) What is name of your organization?</td>
</tr>
<tr>
<td></td>
<td>(ii) What is number of employees at organization?</td>
</tr>
<tr>
<td></td>
<td>(iii) What is your role within the organization?</td>
</tr>
<tr>
<td>3</td>
<td>Elicitation of Handover problems</td>
</tr>
<tr>
<td></td>
<td>(i) What main problems have you encountered within a handover process?</td>
</tr>
<tr>
<td></td>
<td>(ii) What impact have these problems had within your organization?</td>
</tr>
<tr>
<td></td>
<td>(iii) What suggestions do you have for remedying the problems?</td>
</tr>
</tbody>
</table>
# Appendix B.3

## Questionnaire used in the Small-Scale and Large Scale Exploration studies

This appendix presents the questionnaire used for evaluating and evolving the EM3: Handover Taxonomy activities during Small-Scale Exploration and Large-Scale Exploration phases of the research process.

<table>
<thead>
<tr>
<th>No</th>
<th>Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Interviewee personal data questions</td>
</tr>
<tr>
<td>(i)</td>
<td>What is your name?</td>
</tr>
<tr>
<td>(ii)</td>
<td>What is your email?</td>
</tr>
<tr>
<td>(iii)</td>
<td>What is your telephone number?</td>
</tr>
<tr>
<td>2</td>
<td>Organization basic information questions</td>
</tr>
<tr>
<td>(i)</td>
<td>What is name of your organization?</td>
</tr>
<tr>
<td>(ii)</td>
<td>What is number of employees at your organization?</td>
</tr>
<tr>
<td>(iii)</td>
<td>What is your role within the organization?</td>
</tr>
<tr>
<td>(iv)</td>
<td>What type of products do you develop?</td>
</tr>
<tr>
<td>(v)</td>
<td>What development model is followed in your organization?</td>
</tr>
<tr>
<td>(vi)</td>
<td>(a) Waterfall? (b) Agile (c) other? Please specify</td>
</tr>
<tr>
<td>3</td>
<td>Handover basic information questions</td>
</tr>
<tr>
<td>(i)</td>
<td>Is your organization involved in transition?</td>
</tr>
<tr>
<td>(ii)</td>
<td>What are the reasons for transitioning software system at your organization?</td>
</tr>
<tr>
<td>(iii)</td>
<td>Please point out the type of transition at your organization</td>
</tr>
<tr>
<td>(a)</td>
<td>Developer to developer?</td>
</tr>
<tr>
<td>(b)</td>
<td>Developer to maintainer in the same organization?</td>
</tr>
<tr>
<td>(c)</td>
<td>Developer to maintainer in separate organizations?</td>
</tr>
<tr>
<td>(d)</td>
<td>Other? Please specify</td>
</tr>
<tr>
<td>4</td>
<td>EM3 Handover Taxonomy questions repeated for each activity</td>
</tr>
<tr>
<td>(i)</td>
<td>Is this activity performed?</td>
</tr>
<tr>
<td>(ii)</td>
<td>Who does exactly perform this activity?</td>
</tr>
<tr>
<td>(iii)</td>
<td>When in the lifecycle of the system is it performed?</td>
</tr>
<tr>
<td>(iv)</td>
<td>How is it performed?</td>
</tr>
<tr>
<td>(v)</td>
<td>Is any activity missing in this component?</td>
</tr>
</tbody>
</table>
Appendix B.4

Questionnaire for walking through a handover process
This appendix presents the questionnaire used for exploring the handover process in the Small-Scale Exploration phase.

<table>
<thead>
<tr>
<th>No</th>
<th>Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Interviewee personal data questions</td>
</tr>
<tr>
<td>(i)</td>
<td>What is your name?</td>
</tr>
<tr>
<td>(ii)</td>
<td>What is your email?</td>
</tr>
<tr>
<td>(iii)</td>
<td>What is your telephone number?</td>
</tr>
<tr>
<td>2</td>
<td>Organization basic information questions</td>
</tr>
<tr>
<td>(i)</td>
<td>What is name of your organization?</td>
</tr>
<tr>
<td>(ii)</td>
<td>What is number of employees at organization?</td>
</tr>
<tr>
<td>(iii)</td>
<td>What is your role within the organization?</td>
</tr>
<tr>
<td>3</td>
<td>Handover basic information questions</td>
</tr>
<tr>
<td>(i)</td>
<td>Is your organization involved in transition?</td>
</tr>
<tr>
<td>(ii)</td>
<td>Please point out the type of transition at your organization</td>
</tr>
<tr>
<td>(a)</td>
<td>Developer to developer?</td>
</tr>
<tr>
<td>(b)</td>
<td>Developer to maintainer in the same organization?</td>
</tr>
<tr>
<td>(c)</td>
<td>Developer to maintainer in separate organizations?</td>
</tr>
<tr>
<td>(d)</td>
<td>Any other type of transition? Please specify</td>
</tr>
<tr>
<td>4</td>
<td>Handover walkthrough questions</td>
</tr>
<tr>
<td>(i)</td>
<td>Please provide general background information regarding the project lifecycle and the handover process?</td>
</tr>
<tr>
<td>(ii)</td>
<td>What groups/ organizations are involved in the transition process?</td>
</tr>
<tr>
<td>(iii)</td>
<td>What activities are performed in the handover process?</td>
</tr>
<tr>
<td>(iv)</td>
<td>What is the sequence of performing transition activities?</td>
</tr>
</tbody>
</table>
Appendix C

Templates

Appendix C.1

Taxonomy components template

This appendix presents the templates prepared for students for filling in answers after conducting interviews in the *Large-Scale Exploration* phase.

<table>
<thead>
<tr>
<th>Maintainability Management</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Is this activity performed?</strong></td>
</tr>
<tr>
<td>M 1. Assess system maintainability.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Deployment</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Is this activity performed?</strong></td>
</tr>
<tr>
<td>DP 1. Develop installation procedures.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Version and Configuration Management</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Is this activity performed?</strong></td>
</tr>
<tr>
<td>VC 1. Establish software configuration baseline.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maintenance Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Is this activity performed?</strong></td>
</tr>
<tr>
<td>MT 1. Determine hardware/software suite needs.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Is this activity performed?</strong></td>
</tr>
<tr>
<td>D 1. Establish a system documentation repository.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Management and Administration</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Is this activity performed?</strong></td>
</tr>
<tr>
<td>MA 1. Mentify maintenance organization.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Training</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Is this activity performed?</strong></td>
</tr>
<tr>
<td>T 1. Train on system, its structure and</td>
</tr>
</tbody>
</table>

| Missing Activity 1(if any) Please replace this row's text with activity title. |
Example of MA component answers filled for one organization
This appendix presents snapshot of filled in template of Management and Administration component for one organization in the Large-Scale Exploration phase.

<table>
<thead>
<tr>
<th>Is this activity performed?</th>
<th>Who exactly perform it?</th>
<th>When it is performed?</th>
<th>How it is performed?</th>
</tr>
</thead>
<tbody>
<tr>
<td>MA 1. Identify maintenance organization.</td>
<td>Yes</td>
<td>Product Manager</td>
<td>DP</td>
</tr>
<tr>
<td>MA 2. Establish a transition team.</td>
<td>Yes</td>
<td>Product Manager</td>
<td>DP</td>
</tr>
<tr>
<td>MA 3.1. Determine transition budget.</td>
<td>Yes</td>
<td>Project Manager</td>
<td>PDM</td>
</tr>
<tr>
<td>MA 3.2. Create a transition schedule.</td>
<td>Yes</td>
<td>Project Manager</td>
<td>PDM</td>
</tr>
<tr>
<td>MA 3.3. Establish transition procedures.</td>
<td>Yes</td>
<td>System Architect</td>
<td>PDM</td>
</tr>
<tr>
<td>MA 3.4. Define transition resource requirements.</td>
<td>Yes</td>
<td>Project</td>
<td>PDM</td>
</tr>
<tr>
<td>Activity Description</td>
<td>Yes/No</td>
<td>Owner</td>
<td>Notes</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------------------</td>
<td>--------</td>
<td>------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>MA 3.4.1. Define maintenance manpower requirements</td>
<td>Yes</td>
<td>Project Manager</td>
<td>PDM</td>
</tr>
<tr>
<td>Project Manager evaluate how many days a person can complete the requirements? From this result, they can decide how many members can complete requirements in a specific time.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MA 3.4.2. Define maintenance facility requirements</td>
<td>Yes</td>
<td>System Architect</td>
<td>PDM</td>
</tr>
<tr>
<td>System Architects define what facilities (hardware/software resources ...) needed to finish requirements.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MA 3.5. Define other transition elements (not covered in this study)</td>
<td>Yes</td>
<td>Product Manager</td>
<td>PDM</td>
</tr>
<tr>
<td>Product Managers review all of activities whether accept or not accept the transition plan.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MA 4. Develop management plans</td>
<td>Yes</td>
<td>Project Manager</td>
<td>PDM</td>
</tr>
<tr>
<td>Project Managers create management plans: Create schedule to report the results, Create to do list in case of reaching deadline of particular tasks.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Missing Activity 1(if any) Please replace this row’s text with activity title.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Data organizing template

This appendix presents the snapshot of template with filled in data for one activity of Management and Administration component. The data was collected during the Large-Scale Exploration phase and organized in this way for analysis purposes.

<table>
<thead>
<tr>
<th>ORG ID</th>
<th>Is this activity performed?</th>
<th>Who exactly perform it?</th>
<th>When it is performed?</th>
<th>How it is performed?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Project manager</td>
<td>PDM</td>
<td>The project manager defined the participants of the transition team from the development organization. There was no information how the maintenance organization was organized for the transition process and what was the role of the product owner during his phase.</td>
</tr>
<tr>
<td>2</td>
<td>yes</td>
<td>Developers, Product Development Manager</td>
<td>Deployment</td>
<td>They select the most suitable persons (one or more developer). Usually the most experienced ones and familiar with the software.</td>
</tr>
<tr>
<td>3</td>
<td>Yes</td>
<td>CTO (Chief Technical Officer)</td>
<td>After project initial estimation</td>
<td>Assign task according to role. i.e. Project manager performs project planning and project monitoring and control. Developer performs coding and unit test. QC performs testing.</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Project Manager</td>
<td>Development</td>
<td>When a module/segment/section is completed it is needed to be forwarded to another department. Developers and management take part in establishing transition team.</td>
</tr>
<tr>
<td>5</td>
<td>yes</td>
<td>Development Project Manager</td>
<td>From Development stage they are involved in all activities.</td>
<td>Team leader from maintenance along with Development leader</td>
</tr>
<tr>
<td>6</td>
<td>Yes</td>
<td>Developer</td>
<td>2- System testing</td>
<td>When primary decision has been taken by the management team proceed for further work.</td>
</tr>
<tr>
<td>9</td>
<td>Yes</td>
<td>Product Manager</td>
<td>DP PDM</td>
<td>Product Managers are responsible for:   * Select available members for transition team in case of the same organization   * Require to establish a transition team in case of other organization   * Identify team leader of the transition team   * Identify persons who will communicate between development and transition team.</td>
</tr>
</tbody>
</table>
Appendix D

Bodies Involved
This appendix presents the organizations involved in the whole research process. We arrange organizations according to the research process phase. The phases are listed under the Study Phase column.

<table>
<thead>
<tr>
<th>Org</th>
<th>Emp</th>
<th>Role</th>
<th>Domain</th>
<th>Country</th>
<th>Study Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>80000</td>
<td>Project Manager</td>
<td>Telecom</td>
<td>Sweden</td>
<td>Scope Study</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>Team Leader</td>
<td>IT security</td>
<td>Sweden</td>
<td>Scope Study</td>
</tr>
<tr>
<td>3</td>
<td>900</td>
<td>Project Manager</td>
<td>IT Leasing solutions</td>
<td>Pakistan</td>
<td>Scope Study</td>
</tr>
<tr>
<td>4</td>
<td>50</td>
<td>CEO</td>
<td>IT Security</td>
<td>Iran</td>
<td>Scope Study</td>
</tr>
<tr>
<td>5</td>
<td>150</td>
<td>VP Product Development</td>
<td>B2B solutions</td>
<td>Sweden</td>
<td>Scope Study</td>
</tr>
<tr>
<td>6</td>
<td>20</td>
<td>Developer</td>
<td>IT consultancy</td>
<td>Germany</td>
<td>Scope Study</td>
</tr>
<tr>
<td>7</td>
<td>175</td>
<td>Head Operations</td>
<td>Financial applications</td>
<td>Pakistan</td>
<td>Scope Study</td>
</tr>
<tr>
<td>8</td>
<td>200</td>
<td>SQA Engineer</td>
<td>Software development, BPO</td>
<td>Pakistan</td>
<td>Scope Study</td>
</tr>
<tr>
<td>9</td>
<td>20000</td>
<td>Head of Department</td>
<td>Finacial systems</td>
<td>Sweden</td>
<td>Scope Study</td>
</tr>
<tr>
<td>10</td>
<td>60</td>
<td>CTO</td>
<td>Revenue and billing solutions</td>
<td>Sweden</td>
<td>Scope Study</td>
</tr>
<tr>
<td>11</td>
<td>1200</td>
<td>Consultant</td>
<td>IT procurement consultancy</td>
<td>Sweden</td>
<td>Scope Study</td>
</tr>
<tr>
<td>12</td>
<td>2000</td>
<td>Software Engineer</td>
<td>Financial systems</td>
<td>Germany</td>
<td>Scope Study</td>
</tr>
<tr>
<td>13</td>
<td>450</td>
<td>IT Architect</td>
<td>Application infrastructure</td>
<td>Turkey</td>
<td>Scope Study</td>
</tr>
<tr>
<td>14</td>
<td>400000</td>
<td>Consultant</td>
<td>SAP solutions</td>
<td>Sweden</td>
<td>Scope Study</td>
</tr>
<tr>
<td>15</td>
<td>24</td>
<td>Software Engineer</td>
<td>Healthcare, graphics, CRM</td>
<td>Bangladesh</td>
<td>Scope Study</td>
</tr>
<tr>
<td>16</td>
<td>542</td>
<td>Solution Integrator</td>
<td>Telecom</td>
<td>Sweden</td>
<td>Scope Study</td>
</tr>
<tr>
<td>17</td>
<td>300</td>
<td>Consultant</td>
<td>Supply chain management</td>
<td>Thailand</td>
<td>Scope Study</td>
</tr>
<tr>
<td>18</td>
<td>8</td>
<td>Developer</td>
<td>Mobile Applications</td>
<td>France</td>
<td>Scope Study/ Elicitation of Handover Problems</td>
</tr>
<tr>
<td>19</td>
<td>80000</td>
<td>Project Manager</td>
<td>Telecom</td>
<td>Sweden</td>
<td>Elicitation of Handover Problems</td>
</tr>
<tr>
<td>20</td>
<td>50</td>
<td>Senior System Engineer</td>
<td>IT Security</td>
<td>Sweden</td>
<td>Elicitation of Handover Problems</td>
</tr>
</tbody>
</table>
## Appendix D

### Bodies Involved (continued)

<table>
<thead>
<tr>
<th>Org</th>
<th>Emp</th>
<th>Role</th>
<th>Domain</th>
<th>Country</th>
<th>Study Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>40</td>
<td>Software Engineer</td>
<td>IT Security</td>
<td>Neatherlands</td>
<td>Elicitation of Handover Problems</td>
</tr>
<tr>
<td>22</td>
<td>2000</td>
<td>IT Development Manager</td>
<td>Telecom</td>
<td>Pakistan</td>
<td>Elicitation of Handover Problems/ Small Scale Study</td>
</tr>
<tr>
<td>23</td>
<td>13000</td>
<td>Senior Software Developer</td>
<td>Aviation industry</td>
<td>Sweden</td>
<td>Small Scale Study</td>
</tr>
<tr>
<td>24</td>
<td>10000</td>
<td>Business Support System Executive</td>
<td>Telecom</td>
<td>Pakistan</td>
<td>Small Scale Study</td>
</tr>
<tr>
<td>25</td>
<td>1000</td>
<td>Project Manager</td>
<td>Telecom ,WiMAX and LTE</td>
<td>Russia</td>
<td>Large Scale Exploration P-1</td>
</tr>
<tr>
<td>26</td>
<td>16500</td>
<td>System Architect</td>
<td>Internet web applications</td>
<td>Sweden</td>
<td>Large Scale Exploration P-1</td>
</tr>
<tr>
<td>27</td>
<td>600</td>
<td>Development Manager</td>
<td>Web based system for Internet marketing</td>
<td>Sweden</td>
<td>Large Scale Exploration P-1</td>
</tr>
<tr>
<td>28</td>
<td>250</td>
<td>Software Test Engineer</td>
<td>Web and desktop application</td>
<td>Bangladesh</td>
<td>Large Scale Exploration P-1</td>
</tr>
<tr>
<td>29</td>
<td>1500</td>
<td>Team Lead</td>
<td>Software development , Databases, IT consultancy</td>
<td>Pakistan</td>
<td>Large Scale Exploration P-1</td>
</tr>
<tr>
<td>30</td>
<td>400000</td>
<td>SAP Consultant</td>
<td>Implementing SAP software</td>
<td>Sweden</td>
<td>Large Scale Exploration P-1</td>
</tr>
<tr>
<td>31</td>
<td>15</td>
<td>Software Developer</td>
<td>Open source, ecommerce solutions and web development</td>
<td>Bangladesh</td>
<td>Large Scale Exploration P-1</td>
</tr>
<tr>
<td>32</td>
<td>10000</td>
<td>Software Designer</td>
<td>Telecom platform application</td>
<td>Sweden</td>
<td>Large Scale Exploration P-1</td>
</tr>
<tr>
<td>33</td>
<td>120</td>
<td>Java Teams Leader</td>
<td>Web based applications</td>
<td>UAE</td>
<td>Large Scale Exploration P-1</td>
</tr>
<tr>
<td>34</td>
<td>73</td>
<td>Project Manager</td>
<td>Web applications and mobile applications for smart phones</td>
<td>Bangladesh</td>
<td>Large Scale Exploration P-1</td>
</tr>
<tr>
<td>35</td>
<td>8</td>
<td>Manager</td>
<td>Web based and computer based applications</td>
<td>Iran</td>
<td>Large Scale Exploration P-1</td>
</tr>
<tr>
<td>36</td>
<td>300</td>
<td>Team Lead</td>
<td>Business to business applications</td>
<td>Sweden</td>
<td>Large Scale Exploration P-1</td>
</tr>
<tr>
<td>37</td>
<td>68</td>
<td>Software Engineer</td>
<td>Voice engine in PC &amp; mobile devices</td>
<td>Sweden</td>
<td>Large Scale Exploration P-1</td>
</tr>
<tr>
<td>38</td>
<td>18</td>
<td>Project Manager</td>
<td>Customize software solutions</td>
<td>Iran</td>
<td>Large Scale Exploration P-1</td>
</tr>
<tr>
<td>39</td>
<td>135</td>
<td>Product Manager</td>
<td>Solution provider for Telecommunication systems</td>
<td>Sweden</td>
<td>Large Scale Exploration P-1</td>
</tr>
<tr>
<td>40</td>
<td>50</td>
<td>Web Developer</td>
<td>Development of web-based applications</td>
<td>Nepal</td>
<td>Large Scale Exploration P-1</td>
</tr>
</tbody>
</table>
## Appendix D

### Bodies Involved (continued)

<table>
<thead>
<tr>
<th>Org</th>
<th>Emp</th>
<th>Role</th>
<th>Domain</th>
<th>Country</th>
<th>Study Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>41</td>
<td>800</td>
<td>Developer</td>
<td>Financial Products for banking systems</td>
<td>Iran</td>
<td>Large Scale Exploration P-1</td>
</tr>
<tr>
<td>42</td>
<td>399409</td>
<td>Business Manager</td>
<td>Business Intelligence software, ERP software, B2B applications</td>
<td>Sweden</td>
<td>Large Scale Exploration P-1</td>
</tr>
<tr>
<td>43</td>
<td>200</td>
<td>Software Engineer</td>
<td>Web based application</td>
<td>Pakistan</td>
<td>Large Scale Exploration P-1</td>
</tr>
<tr>
<td>44</td>
<td>87400</td>
<td>SW Designer</td>
<td>Telecommunication system</td>
<td>Sweden</td>
<td>Large Scale Exploration P-1</td>
</tr>
<tr>
<td>45</td>
<td>60</td>
<td>Software Integrator</td>
<td>Telecom software</td>
<td>Sweden</td>
<td>Large Scale Exploration P-1</td>
</tr>
<tr>
<td>46</td>
<td>3000</td>
<td>Data Manager</td>
<td>Portal development, database management</td>
<td>Pakistan</td>
<td>Large Scale Exploration P-1</td>
</tr>
<tr>
<td>47</td>
<td>47000</td>
<td>Testing Analyst</td>
<td>Applications for SAP and legacy systems</td>
<td>Mexico</td>
<td>Large Scale Exploration P-1</td>
</tr>
<tr>
<td>48</td>
<td>45</td>
<td>Deployment Manager</td>
<td>CMS for online video distribution</td>
<td>Sweden</td>
<td>Large Scale Exploration P-1</td>
</tr>
<tr>
<td>49</td>
<td>250</td>
<td>Project Manager</td>
<td>ERP, CRM systems, BI</td>
<td>Sweden</td>
<td>Large Scale Exploration P-1</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>Team Leader</td>
<td>ERP, MIS, accounting management system</td>
<td>Bangladesh</td>
<td>Large Scale Exploration P-1</td>
</tr>
<tr>
<td>51</td>
<td>757</td>
<td>Software Engineer</td>
<td>Web browser</td>
<td>Sweden</td>
<td>Large Scale Exploration P-1</td>
</tr>
<tr>
<td>52</td>
<td>10</td>
<td>Application Consultant</td>
<td>ERP-system</td>
<td>Sweden</td>
<td>Large Scale Exploration P-1</td>
</tr>
<tr>
<td>53</td>
<td>2500</td>
<td>Integration Manager</td>
<td>E services</td>
<td>Sweden</td>
<td>Large Scale Exploration P-1</td>
</tr>
<tr>
<td>54</td>
<td>80</td>
<td>Support</td>
<td>Internet security applications</td>
<td>China</td>
<td>Large Scale Exploration P-2</td>
</tr>
<tr>
<td>55</td>
<td>47000</td>
<td>Support Consultant</td>
<td>Internal software for testing system</td>
<td>Ireland</td>
<td>Large Scale Exploration P-2</td>
</tr>
<tr>
<td>56</td>
<td>50</td>
<td>Project leader</td>
<td>Telecom billing application</td>
<td>UAE</td>
<td>Large Scale Exploration P-2</td>
</tr>
<tr>
<td>57</td>
<td>350</td>
<td>Technology Manager</td>
<td>Desktop and web based applications</td>
<td>Ireland</td>
<td>Large Scale Exploration P-2</td>
</tr>
<tr>
<td>58</td>
<td>700</td>
<td>Software Architect</td>
<td>Provisioning, mediation and charging</td>
<td>Turkey</td>
<td>Large Scale Exploration P-2</td>
</tr>
<tr>
<td>59</td>
<td>230</td>
<td>Team Lead</td>
<td>Web design, E commerce</td>
<td>Bangladesh</td>
<td>Large Scale Exploration P-2</td>
</tr>
<tr>
<td>60</td>
<td>60</td>
<td>Developer</td>
<td>B2B applications</td>
<td>Sweden</td>
<td>Large Scale Exploration P-2</td>
</tr>
<tr>
<td>61</td>
<td>30</td>
<td>Participatory Observation</td>
<td>IT Security</td>
<td>Sweden</td>
<td>Final Study</td>
</tr>
</tbody>
</table>
Appendix E

EM³: Handover Taxonomy Guidelines

Management and Administration

Guideline 1: Determine the type and complexity of the handover process
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 also the transitionees, or it might be an external transition where the transitioners and transitionees are separate teams or organizations. The transition might be of a high complexity implying a handover of a safety critical system among several organizations or it might be as simple as a self-to-self handover from development to maintenance of a newly developed system version. It is critical to investigate the type of transition in advance to foresee and plan for the handover process, and thereby, remedy the problems which might occur during transition.

Guideline 2: Define a strategy for conducting handover
The organization must define a strategy for executing a transition process. The strategy must focus on pointing out motives, main challenges and guidelines for transition implementation. It may include ways of designating a maintenance team, determining the time when maintenance team should start interacting, approaches of transferring system knowledge and like. The transition strategy functions as the main point of reference for designing the transition plan, for determining the core transition process activities, for identifying the main stakeholders involved, and for determining budget requirements.

Guideline 3: Designate the transitionee/maintainer as early as possible
It is important that the transitionee/maintenance team be identified as early as possible. Only then the maintenance team members are prepared for taking over the new system. The transitionee team is responsible for evolving and maintaining a software system after transition. Its organizational membership and formation varies in different contexts. Overall, maintenance either stays with the development where it is delegated to one or several individuals or it is transferred to another team within the same organization, or it is transferred to a totally separate organization. To manage this wide spectrum of the maintenance actors, we refer to them all as transitionee teams. It is worthwhile to mention that transitionee may change with time. The change in transitionee designation happens due to changes in maintenance workload, financial restrictions or due to other reasons.

Guideline 4: Make an agreement on the post-handover responsibilities of the transitionee
The system stakeholders must formally agree on a service level agreement describing the services to be provided by the transitionee after handover. The transitionee’s
responsibilities may range from providing only corrective maintenance to taking full responsibility for system evolution and maintenance. The agreement aids the transitionee in understanding the maintenance workload and in allocating the maintenance resources, accordingly.

**Guideline 5: Diversify the roles involved in the transition team**
The handover process needs to have a role that is answerable and accountable for its planning, management and control. Such a role is the transition team. The transition team should be established as soon as possible, preferably at the conception of the software system to be developed. Its members should include representatives of the different stakeholders involved, such as developer, maintainer, acquirer, and COTS suppliers. Only then one may make sure that they represent different perspectives of a handover process. The members should possess the right capabilities and experience to carry out the transition.

**Guideline 6: Define a common transition process**
It is significant to define a transition process in advance. A transition process should enlist all the major transition activities, and their sequence. Establishing transition process is extremely important bearing in mind the fact that the transition team members may belong to organizations having diverse individualistic, collectivist and organizational cultures. All of them may have their own views and understanding of the transition process. In order not to fail, one must make sure that they have all agreed upon the common transition procedures. It is important that all the stakeholders create a consensus on the executed transition process. This minimizes the risk of ambiguities or contradictions during transition implementation.

**Guideline 7: All Management and Administration transition activities should be implemented**
The core Management and Administration activities are the activities that are essential for implementing a transition process and crucial for succeeding with the process. All of them should be planned for and implemented. In some cases, some of the Management and Administration activities cannot be conducted. Then, their non-implementation must be justified. As an example, formation of a transition team is a core transition activity. Failure to implement it may result in jeopardizing the whole transition process. However, in case of simple systems undergoing a self-to-self transition, the role of the transition team may be conducted by a project manager.

**Guideline 8: Always plan for transition**
All software engineering activities, whether complex or not, should have a plan for achieving their objectives. Transition is no exception here. It should have a plan worked out beforehand for the successful accomplishment of handover from developer to maintainer. The EM transition plan maps out transition budget, transition schedule and it defines maintenance resource requirements.
**Guideline 9: Assure that the handover process has been assigned proper resources**

A transition plan should define transition resource requirements. These requirements include (1) definition of manpower requirements and (2) definition of maintenance facility requirements.

Regarding the first requirement, the manpower resources, it should include personnel from (1) development team, (2) maintenance team, (3) transition team, and (4) other roles, if required. Maintenance team has the main responsibility of taking over the system and providing maintenance services. Especially the initial period after software system handover is always very critical. The maintenance team members have to deal with a huge flux of change requests. Hence, the maintenance team members must include a right number of personnel to handle customer requests. Their number depends upon the size and complexity of the transitioned system. Moreover, they must be experienced and they must have ability to work under stress.

Regarding the development team, they have the main responsibility to transfer the knowledge of the transitioned system. They must work with the maintenance team for making the transition successful. When actually transitioning the system, the development team is under immense pressure to meet the delivery deadlines. Depending on the type of transition and nature of the transitioned system, a reasonable number of developers must be available to sort out the problems faced by the maintenance team during handover. In case of self-transition, the development team continues with maintenance.

Regarding the transition team, its composition should be reassessed for including more members before the actualization of software system handover. Developer, maintainer and transition team are the main roles, however, more roles can be included depending on the nature of the system. For example, COTS suppliers should be included as a role if the system is using the COTS components.

Regarding the second requirement, that is facility requirements, maintenance team cannot perform their duties without adequate resources. Hence, a transition team should determine the appropriate hardware and software facilities needed for maintaining the system. Examples of them are hardware and software suite, system software baseline and support suites.

**Guideline 10: Transition budget should be determined in advance**

The cost of transition varies depending on the system size, complexity and the number of the parties involved. If the budget is not determined in advance, then the risk is that too few resources will be assigned to it, and thereby, the overall transition realization may fail. For this reason, the determination of the budget is very important and critical for the success of a transition process.
**Guideline 11: Assure that the transition members commit to the transition schedule**

All processes should have a schedule and so does the transition process too. To create it, however, is not always easy. First, scheduling in general is a very time-consuming activity. Second, it is always a challenge to balance the needs of a process and the availability of the crucial roles involved in the process. In the context of transition, it is an extremely challenging task. This is because multiple parties participate in transition and these parties may be geographically distributed. Moreover, they are highly experienced professionals involved in many critical tasks in their respective organizations. Transition is only one of the many tasks they are engaged in. Hence, their availability, although crucial for the transition, may be significantly limited. Therefore, it is always difficult to develop a transition schedule so that all the important transition team members can participate in and contribute to the transition project in a timely manner. To facilitate it, the transition schedule should be created as early as possible and all the crucial roles in the transition team should be fully committed to it.

**Guideline 12: Management plans of the related process should be coherent with the transition plan**

Handover is not a standalone process. It intersects with many other process areas, where each such area should be planned and managed as well. For this reason, the management plans of these other process areas should be in place before the implementation of a handover process starts and they should be coherent with the transition plan. The plans of relevance include development plans, maintenance plans, software configuration management plan, training program plan, test plan, quality control plan, deployment plans and many other plans that may be relevant for enabling and/or for facilitating the transition process.

**Guideline 13: Determine a communication model to be used within transition**

Various parties get involved in the handover process. They may belong to different organizations having different cultures, processes and they may use different terminologies. Moreover, they may be located at different parts of the world. Hence, it is important that the parties involved in handover must agree beforehand on the common terminology to be used during communication and determine the channels of communications such as video conferencing, emails or telephone conversations and other channels. Otherwise, the communication may become unsuccessful and it may result in misunderstanding and misinterpretation which ultimately may result in conflicts and handover delay.

**Guideline 14: Transition process should be monitored**

Many hidden challenges and problems may arise during the execution of any process within any passage of time. The same is the case with the handover process. It is difficult to foresee all the scenarios that may arise during the handover process implementation. Therefore, it is very important to continuously monitor the handover process execution,
reassess the situation, and reconsider the already agreed plans, assigned budget and resources, and all the other decisions.

**Guideline 15: Do not forget to evaluate the transition process postmortem**

The stakeholders involved in the handover implementation should evaluate the transition process. This helps in sharing the experiences gained, in figuring out the activities that were not well planned and in pointing out the handover problems which were not anticipated beforehand. The evaluation helps in improving the handover process for future transitions. To assure that the handover processes are evaluated postmortem, one should plan for postmortem evaluation in the transition plan.

**Maintenance Environment**

**Guideline 1: The hardware and software suite should be comparative across all the sites**

Hardware includes workstations, processors, storage devices, web servers, database servers, backup servers, networking apparatus and other facilities like telecommunication hardware, printers and other. The software suite includes operating system, integrated development environment and other software packages important for providing maintenance services.

It is significant to determine the hardware and software suite needs. In the case when the development team continues with maintenance, it is a trivial activity. The development team uses the existing suites used during development to provide maintenance services. In the case when the maintenance is transferred to another team, then the maintenance team requires the same setup of the hardware and software environment as used during the development. In both cases, the maintainer must consult with the customer to ensure that the maintenance environment is compatible with the operational environment. In the case of deficiencies, the maintenance team should acquire new suites. It usually concerns the acquisition of the latest versions of compliers, servers and licenses for software packages.

**Guideline 2: Create a manual instructing on how to install hardware and software**

The hardware and software installation manual includes tasks like setting up and configuring hardware and software suites and granting access permissions to maintenance team members. It is a good practice to create an installation manual providing the guidelines for the installation on the maintenance site. This aids maintenance team to create the same environment setup as on the developer’s site.

**Guideline 3: Appropriate support suites should be acquired and continuously reassessed**

It becomes the maintainer’s responsibility to resolve the problems faced by the customer after system handover. To effectively and efficiently address the customer problems and modification requests, the maintainer needs to determine and install support suite software. The support suites mainly include compilers, software support for tracking bugs and tools for managing problems reported by the customers. In some cases, the support suites also include virtual machine software, where a single machine can exhibit the working of
multiple machines. Virtual machines help the maintenance team to reproduce the problems reported by the customer with the limited number of available machines at the support site.

For systems that are already under maintenance, the existing support suites should be reassessed and complimented with new tools, if required. New suites get introduced (1) in cases when one knows that they aid in making problem reporting and management more effective and efficient, or, (2) in cases when the development team changes the tools supporting the development environment. In the latter case, the maintenance support suite is supplemented with new tools that are compatible with the new development environment.

**Version and Configuration Management**

**Guideline 1: Commonly agree upon naming and labeling rules of the configuration items (CIs) to be handed over**

To avoid ambiguities and conflicts during handover with respect to the system configuration and its items, it is important that the organizations involved (transitioner and transitionee) commonly agree upon the naming and labeling rules of the configuration items. Each CI should be identified, named, and described with the information on product number, version dependencies, hardware and operating systems, and the version of the associated documentation items to be transitioned. The documentation items, that are per se, configuration items, mainly correspond to requirement specifications, design and test documents and user manuals.

**Guideline 2: Group configuration items, if relevant and possible**

For better management, the configuration items (CI) should be classified under different categories. For instance, the categories may include (1) hardware, (2) software (3) operating system, and (4) documentation. There are two ways of grouping configuration items. These are (1) grouping of all configuration items under a single version, and (2) creating different variants of configuration items according to the needs of customers and then grouping them together into versions that are customized for particular customers. The organizations that create variants of the same configuration items are the organizations that have multiple customers with strongly varying system needs. All the information about the CI groups and variants should be handed over to the transitionee.

**Guideline 3: Track the changeable configuration items**

During handover, it is a good practice to identify the changeable items and list them in a configuration item register. The register may be a simple file such as an Excel file with names and version numbers of all the configuration items or it may be an advanced CI management tool. Irrespective of how advanced it is, the tool can aid the transitioners and transitionees in tracking and controlling the items during system transition. Especially important is it to keep an eye on the components that are candidates for frequent changes. This ensures that the configuration of the software packages being handed over from the transitioner has the same status as the configuration being received by the transitioner.
**Guideline 4: Use configuration management system for controlling the status of configuration items**

A software system may consist of hundreds of CIs. It is not always possible to track their status manually. Therefore, it is a good practice to upload configuration items in a configuration management system. When putting software under configuration management, it is significant to prepare and execute configuration management plans and establish the structure of the software system to be handed over. It is important to handle dependencies between software and hardware and, wherever relevant, create multiple configurations of the same system for different acquirers.

**Guideline 5: Synchronize the status of the system versions on the transitionee and transitioner’s sites**

The software system to be handed over should be put under version control. It is significant to maintain different branches of code during handover. For instance, three branches of code can be formed after the acceptance testing phase. These include (1) main branch to be used as a central master copy of the system, (2) development branch to be used as a basis for further development, and (3) maintenance branch to be used by the maintainers for creating minor releases. The main branch should be systematically updated after each minor release by the maintainers and its status should be regularly synchronized with the development branch by the development team after every new system build has been created.

**Guideline 6: Involve the transitionee in acceptance testing baseline**

The acceptance test baseline is installed on the premises of the acquirer and the developers and maintainers attend to all the problems encountered in it. It is the time when the acquirer starts interacting with the system and when the system undergoes many changes made both by the developer and maintainer. The acceptance testing environment is also known as the staged environment in some organizations.

It is a good practice for the maintainer to interact with the acquirer and try to resolve their problems during the acceptance testing. In this way, the maintainers gain knowledge about the system, and test the system from the maintenance perspective. The maintainer’s participation helps in reducing the developer’s workload and the involvement of the two groups makes the acceptance testing more efficient. Not only the developers but also the maintainers keep track of the changes made to the system during the acceptance testing phase. This, in turn, substantially shortens the lead time of the handover process.

**Guideline 7: Assure that the operational baseline agrees with the development and maintenance baselines**

Post-delivery baselines include operation baseline and maintenance baseline. Operation baseline is installed on the acquirer site. It is always the acquirer who gives the final approval for deploying the system after acceptance tests. It is common knowledge that it is not always possible to have a defect free system at the end of the acceptance testing phase. However, if the existing defects are not of critical nature and they do not negatively affect the system operation, then the acquirer may accept the system for deployment. At this moment, post-deployment maintenance starts. In its early phase, the changes made to the
software system are still performed by both the maintainers and developers, to be then gradually taken over by only the maintainers.

Maintenance baseline is installed on the maintainer’s site. Development and maintenance teams must have identical copies of the operational environment and they should always communicate with each other on all the changes made to these copies.

**Training**

**Guideline 1: A role should be designated for managing the training process**

Training is a complicated process and its complexity gets augmented in the context of software system handover. Therefore, it is important to designate a role exclusively responsible for managing the training process. The role should develop educational policies, monitor the training activities and play the role of a bridge between different trainer or trainee roles participating in the training process.

**Guideline 2: Do proper planning for training provision**

Training provision is not a spontaneous activity and it requires proper planning. Training planning includes identification of training groups and identification of training topics to be taught. Moreover, planning includes determination of training needs of each group and identifying the method of training.

**Guideline 3: Identify the training topics that are important for handover**

The training topics range from basic system introduction to training on the complex process like maintenance and support. The training topics can be grouped into four major categories. These are (1) training on software systems, their structure and operation, (2) training on maintenance process, (3) training on support process, and (4) training on new technologies. There might be more training topics, for instance, training on deployment process.

There are different stakeholders of the system and each of them has his own training needs. These include management, maintainer, acquirer and support team. The management needs a only basic introduction to the system. The acquirer and user need training on operation of the system. The maintenance team requires detailed training on system design, architecture, and maintenance process. Finally, the support team requires training on the support and problem reporting process.

**Guideline 4: Identify the trainee groups**

It is significant to identify all the trainee groups and their training needs. The level of the required training varies from role to role and person to person depending on the expertise level required. The experienced employees require general refresher training while the freshly inducted employees require detailed training.

In case of large size systems where multiple teams or organizations are involved in managing systems, it is difficult, painstaking and time consuming to provide training to everyone. Therefore, a good practice is to select a few individuals called super users and
provide thorough training to them. The super users then play the role of the educator within their respective teams or organizations.

**Guideline 5: Identify the training methods**

There are various methods which are used for training. The training provision can be a complex task requiring special training environment or it can be simple enough requiring only a system presentation. Depending on the system complexity and pedagogical requirements, system training may be conducted in several sub-stages including provision of general orientation, training on OS and DBMS and training on the system to be handed over. The training can be conducted on either formal or informal levels. The formal training is conducted by arranging special training sessions and the informal training is conducted via informal discussions, emails and telephones. In case of geographical distribution, the development team may transfer knowledge through video conferencing.

**Guideline 6: Create training material that is significant for handover**

Training material is an essential ingredient for training provision during software handover. The training material can be categorized under four groups. These are (1) training material on system, its structure and operation, (2) training material on maintenance process, (3) training material on system support processes, and (4) training material on new technology. The training material includes user manuals, PowerPoint presentations, functional specifications, and wiki material. The preparation of the training material should be a continuous activity. As soon as changes have been made to the system or the process, the related training material must get updated. The training material should be available either online or in a repository.

**Guideline 7: Identify the role responsible for training**

The role responsible for training varies depending on the handover context at hand. There are many training topics and groups who require training. It is neither possible nor practical in nature that one role conducts training in all the topics and for all the groups. Developer has the best knowledge of the system and he is the right role for providing training on system, its structure and operation. If the system is using COTS components, then the COTS suppliers also get involved in providing training. Training on support process may be conducted by the developer or maintainer. The role for training on maintenance process varies. In case of an in-house transition, it may be conducted by the developer. However, in case of inter-organizational handover, it is a good practice to form a special educational group comprising the roles coming from developer, maintenance and acquirer organizations.

**Guideline 8: Train the trainer policy**

Train the trainer is also an efficient way for transferring knowledge. In this case, developer trains a single point of contact (SPOC) person in each department. SPOC works as a trainer for the remaining members in his department. For training of end users, it is suggested that only selected members of the acquirer’s support team get trained. These members, in turn, can train the end-users in the acquirer organization.
**Guideline 9: Do prepare for providing training**

Training preparations include (1) setting up training environment if required and (2) updates or adaptions of the existing training material according to the trainee group needs. The training environment is only established for training on complex systems. In this case, a training infrastructure similar to the operational environment is established. It includes installation of software, hardware and dummy data.

**Guideline 10: Determine the forms of providing training**

Forms of providing training vary and depend on the handover context. Training provision is dependent on the scope of the agreement as defined in the Service Level Agreements (SLAs). In case of a handover of a newly developed system, both formal and informal training is required. Formal training can take the form of classroom, workshop, presentations or various documents, such as for instance, release notes. A good way of providing hands on experience to the maintainer is to provide simulation-based training. Here, the trainer injects defects into the system that are then reproduced by the trainees.

Regarding the informal training, the developers may closely interact with the maintainer through discussion sessions and casual corridor meetings when sharing their knowledge and experiences. In case of a new release of the already installed system, the developer can transfer knowledge via informal communication in form of discussions, emails, and telephone conversations.

**Guideline 11: Involve maintainers in attending modification requests as part of training**

Involving maintainers in attending to modification requests is another practical way of gaining firsthand knowledge of the system. Both the development and maintenance teams should work together to attend to modification requests during handover. The duty of attending to modification requests is then gradually transferred from the development to the maintenance team at the end of handover. This activity helps maintainers in various ways. For instance maintainers get hands on experience with the system and learn the system. Moreover, they can assess the quality of the transitioned system and get a chance to interact with the acquirer and create a bond of trust.

**Guideline 12: Involve maintainers in conducting white box testing and debugging as part of training**

White-box testing and debugging may be regarded as an important training driver. When attending to software problems during system and acceptance testing, maintainers can conduct various forms of debugging and white-box testing aiding them in understanding both the system and the problems being investigated. It is a good training practice to create a specific environment for conducting white-box testing for the training purposes. The changes made in this environment, if any, however, are never uploaded to the operational environment. Only specific problems are white-box tested and debugged by the maintainers for the educational purposes. These problems are usually of a simpler nature. The more severe problems are tested by the developers instead.
Guideline 13: Provide on-site support, if necessary
In case of inter-organization handover, the development organization sends its experienced developers to the maintenance organization to help the maintainers to support the system in the early maintenance phase. This is conducted only in very critical handover situations. The presence of developers on the maintenance site helps the maintainers to smoothly take over the maintenance responsibilities. In less critical situations, the developers provide consulting services by email, telephone, or by arranging video conferences.

Deployment

Guideline 1: Define the scope and contents of release
It is significant to define the scope and contents of the release before installing the system. This helps in creating a common understanding about the functionality of the system to be released and handed over between development, maintenance, installation teams, and the acquirer of the system. All the stakeholders must clearly know what new functionality is included in the new release. In the context of a handover process, defining the scope and contents provides an indication of the future maintenance responsibilities and burden.

Guideline 2: Type of release should be determined in advance
It is important to identify the type of the released system. There are three main types. These are (1) first release of a newly developed system, a (2) major release or (3) minor release of the already deployed system. Determining the type of release helps in identifying the magnitude and complexity of the handover process. The handover process is more complex, requires more resources, budget and time in cases of newly developed systems or major releases. Less effort is required in case of minor releases.

Guideline 3: Create a deployment team in advance
A deployment team responsible for the release should be identified in advance. It is important because this team installs the system and interacts with all the other teams involved in handover. They need to interact with the maintenance and development teams in case the problems arise during deployment. Especially, the maintenance team must know about the system performance during and after deployment. After all, it is the maintenance team which is going to manage the system after handover. Early identification of the deployment team facilitates both maintenance and deployment teams to synchronize their work. Only in this way, the maintenance team can handle problems occurring during deployment, and using their feedback, it can prepare to perform their future maintenance duties.

Guideline 4: Develop procedures to back out release units in case they fail
Back-out procedures should be developed and agreed upon between all the parties involved in handover. In this way, all the stakeholders may effectively synchronize their activities on an emergency basis during handover. This is important because organizations cannot keep the system shut down for longer periods of time while installing new releases. It is not feasible economically and the system should be available to the customer in the least amount of time. Hence, if the system release fails or it starts malfunctioning, it or its parts should be back out to the last stable version.
Guideline 5: Maintainer should have a list of organizations and stakeholders affected by the new release
One release may be developed for one or several customer organizations. These organizations will be, in some way, impacted by the release. The case may also be such that the functionality in some specific release may not be of interest and use to some customers. Hence, not all the customer organizations and/or stakeholders have equal interest in the new release.

When taking over the system, maintainers must know which organizations or stakeholders will be impacted by the new releases. This will help them to get an idea of their engagement, and thereby, aid them in planning resources for and direct their future support and maintenance activities. Therefore, a concrete list should be created containing the names of the organizations and stakeholders that are affected by the release.

Guideline 6: Maintainer should have access to release and build documentation
The release documentation is a useful source of information for the maintenance team. It first provides the maintainer the list of all the features included in the release. This helps the maintainer to focus on extending their future operation on the new release parts. In addition, the release and build documentation provides information about the system environment. The maintenance environment must be identical to the deployment environment at the customer site. Only in this way, the maintenance team can reproduce the problems arising at the customer’s site, identify problem reasons, and provide solutions. The release documentation helps the maintenance team to gain knowledge of the system installation at the customer site and use this knowledge for creating an identical environment at the maintenance site.

Guideline 7: Provide sufficient system access rights to the maintainer
Access rights should be defined for all the system stakeholders. There are many stakeholders interacting with the system for different purposes. For example, users interact to get information or update their own information. They are not concerned about the architecture or code of the system. Hence, they do not need rights to access or change the software modules or data. The maintainers, on the other hand, are responsible for attending to modification requests and problem reports after handover. Hence, they must have access to software code.

Guideline 8: Establish a baseline of a release package
It is critical to baseline the contents of the release package before deployment. Baseline is the frozen version of the system ready for deployment. It means that once a baseline is created, it is not altered by anyone in the team. Content baseline is significant because the same version of the baseline is also handed over to maintenance. If customers report a problem within the latest release, the problem is then reproduced by maintenance team in the maintenance environment. The maintenance environment contains the same baselined version as it is in the operational environment. If the contents of the release is not baselined, then the risk is that the operational and maintenance environments have different versions. In such a case, it becomes difficult to reproduce and identify the problem and provide appropriate solution.
Guideline 9: Perform deployment readiness test
The deployment readiness test must be performed to check the status of the facilities required for installing the system. The facilities include hardware infrastructure, hardware equipment and software packages. Hardware infrastructure includes facilities such as local area network and electricity installations. Hardware equipment mainly includes computers and software includes the operating system. Deployment readiness test ensures that the required apparatus is in place before the deployment begins. After successful deployment tests, all the concerned stakeholders are informed that the deployment site is ready for deployment. This helps people involved in deployment and handover to synchronize their activities and make a schedule for the deployment.

Guideline 10: Distribute and deliver the system and/or system components at a correct location and time
The system must be delivered to a correct location and time for installation. It is significant because there are multiple parties involved in deployment and handover including people from the acquirer, development, maintenance, and support teams. The people involved are highly experienced and their expertise is required on a daily basis in their respective organizations. They have very busy schedules and they cannot afford to sit and wait for the system arrival. Hence, the system should be available for deployment according to the time agreed upon by all the stakeholders involved.

Guideline 11: Share incidents, unexpected events, issues or deviations from the plans
All the incidents, unexpected events and issues should be recorded and shared with project management. This helps management to know about the status of the release. They can analyze the issues and alter their installation procedures so that the issues do not arise in future system deployments. Moreover, this information is useful for the maintenance team because they must have the latest information about the status of the system. This helps them to identify the problems and provide relevant solutions after handover.

Guideline 12: Perform deployment verification tests
Once the system is installed, the deployment verification tests should be performed to check the operation and performance of the system. The deployment verification tests are helpful to compare the deployment outcomes against the deployment plans. This activity enables the stakeholders to check that the system is installed according to the plan and that it is performing according to the expectations. The deployment verification test results are shared with all the stakeholders including the maintenance team that is responsible for the system after handover.

Guideline 13: Maintainer should actively participate in deployment review and closure activity
The deployment is reviewed and closed down after a successful deployment verification testing and after getting feedback from all its stakeholders. This is the time when the customer approves the deployment and the maintainer gradually takes over the responsibilities for managing the system. It is significant that the maintainer actively participates in the deployment review and closure activity. If the maintainer has concerns regarding the quality or stability of the system, he should clearly convey it to other
Guideline 14: Determine the distribution structure before handover
It is important to determine the distribution structure of the software system to be deployed. The software may be designed to be installed on a single standalone machine or it may be designed to be installed on a group of machines. It may also be possible that the software to be deployed is designed to install on multiple machines located at geographically distinct locations. The maintenance team must have knowledge of the system distribution structure in order to prepare themselves for all software problems scenarios which may occur after system handover.

Guideline 15: Determine forms of deploying the software system
Forms of deploying the software system should be determined for future releases. The software may be stored in a compact disk or a hard drive, or it can be placed and accessed online. Determining forms of deployment is a significant activity because formation and working of the release team depends on the form of the deployed software system. In case of online availability, the release team can remotely access and install it. In other more advance deployment cases, they need to physically visit the customer site for installation purposes.

Guideline 16: Determine the impact of the externally acquired components on the future release
A software system comprises several software components. The may be developed in-house or acquired from an external third party. Evolution and maintenance of the externally acquired components is the duty of the third party. Transitioner and transitionee should determine the impact of using the external component while planning for future releases. It may happen that the external components are outdated and not compatible with the future releases. Hence, during the handover, the third party should also be consulted about the status of the external components. A formal agreement should be signed regarding the level of maintenance services to be provided by the third party.

Documentaion

Guideline 1: Establish a common system documentation repository
The system documentation repository is the central place for storing all the project related documentation. The repository works as the first point of reference to gain system and domain knowledge during handover. The repository can either be created at an organizational level or each project can has its own documentation repository. The repository should contain all kinds of documents such as requirements specification, design documents, work breakdown structure, operational instructions, handover and various discussion documents.

Guideline 2: Stakeholder should have adequate access rights to use documentation repository
There are various stakeholders accessing documentation repository to obtain the desired information. Different stakeholders must have different access rights to the documentation
repository. For example, maintenance team need to evolve and maintain the system, hence, they must have full access rights to all the system documents. Support team, on the other hand, must have good understanding of the system. Hence, they need to have access to only the technical documentation. Finally, the customers need access to only user manuals aiding them in system operation.

**Guideline 3: Documentation repository should be subjected to software configuration management**

There are many stakeholders interacting with the documentation repository and they make changes to the documentation throughout the system lifecycle. Particularly, during the system handover, many changes are made to the system and documentation. Hence, it is important to subject the documentation repository to SCM in order to assure the quality and consistency of the documents. This greatly aids in preventing accidental file loss, backtracking to previous versions of the file, managing different releases of the files, and in maintaining a history of document changes.

**Guideline 4: Documentation standards should be established**

Documentation is the key source for gaining information about the transitioned system. The documentation should be developed in a way that it is structured, well organized and understandable by all the stakeholders. If unstructured and disorganized, it may result in creating confusion among the stakeholders and lead to handover inefficiency or even failure. For this reason, stakeholders must agree in advance on establishing standards for documentation development. It is a good practice to develop a template for each document and agree on the terminology to be used when writing documentation.

**Guideline 5: A mechanism for notifying about missing / incomplete documentation should be in place**

Normally, developers/transitioners are always under immense pressure to meet the delivery deadline, and hence, they give secondary importance to documentation. As a result, the transitionee gets outdated documentation which is not very useful during both handover and maintenance. The documentation should be updated on a continuous basis. It is a good practice to create a notification mechanism to remind the transitioner about the missed deadlines for updating or completing the documentation.

**Maintainability Management**

**Guideline 1: Maintainability attributes should be defined**

It is important to define maintainability attributes so that the system can be tested against them for confirming that it is maintainable. Two types of maintainability attributes should be considered. These are system maintainability attributes and data maintainability attributes. System maintainability attributes should encompass different granularity levels ranging from maintainability of system requirements, design documents, code, testing documentation, to user manuals and operational instructions. Regarding data maintainability, it should include attributes like data scalability, database capability for future updates and data consistency, to mention a few.
Guideline 2: Define rules and guidelines for adhering to the maintainability attributes
It is not enough to define maintainability attributes. The organizations must define rules and guidelines for how to follow them. For example, (1) when planning the system, the stakeholders must envision future requirements and pay attention to them, (2) when designing the system the designers must follow the layered architecture approach, and (3) when coding the system the developers must follow the organizational coding standards.

Guideline 3: Identify milestones for assessing maintainability
The system must be checked for maintainability at different points in time in the system lifecycle. It is a good practice to check for maintainability as early as possible. If maintainability is not built in from the very beginning then it becomes more difficult, time consuming and costly to retrofit it later. If the system is transferred to a third party, then the maintenance assessment often takes place in the deployment phase. This is definitely too late for the organizations taking over the system to react to the maintainability deficiencies.

Software System Transfer

Guideline 1: Monitor the status of software components to be handed over
It is significant to monitor the status of software components expected to be handed over. Monitoring includes identification of stable components, components under testing, and components under development. It is only then the transitioner and transitionee have a clear picture of the software system and they can make a decision on the components to be handed over. It is the responsibility of the transitioner to only transfer stable components and the transitionee should always check for the quality and maintainability of the transferred components.

Guideline 2: Monitor the modification requests to revise the handover decision
It is not always possible to deliver a defect free system during handover. The customer submits the modification requests and problem reports on regular basis. It is not possible for the transitioner to satisfy the entire customer requests before handing over the system. Therefore, it is significant to create a template for managing information about modification requests and their management and place these modification requests in a modification request repository. The transitioner and transitionee should analyze the severity and complexity of customer’s change requests and revise the handover decision. If the change requests are not of critical nature, the system and change requests can be transferred to transitionee. Otherwise, the change requests must be deferred till the next release.

Guideline 3: Transitioner should support transitionee in the initial post-handover period
Transferring the agreed upon software components, operational data and unattended modification requests is not enough to conclude the handover. The initial period after the system transfer is always very crucial. This is the time when the transitionee starts interacting with the system in the real world setting and starts providing maintenance services. It is not a good idea to leave the transitionee alone at that point. The transitioner
should interact with the transitionee on a regular basis and help him in resolving issues. Once the transitionee feels confident and satisfied with the system and his ability to tackle the problems, only then both parties should sign off the closure of the handover process.
EM³ Appendix F

Evolution and Maintenance Management Model, abbreviated as EM³, is a process model whose aim is to include the definitions of all process models that are part of the evolution and maintenance domains. EM³ is still under development. As indicated by Figure F1, it includes the following component models:

- **Evolution and Maintenance Management Model: Handover Framework**, a process model to be used for transferring the system for future maintenance to maintenance organization [47] [48] [46] [49] [50] [51] [52] [56] [58] [55] [54] [53] [57].
- **Evolution and Maintenance Management Model: Predelivery Maintenance**, a process model to be used for preparing software systems and maintenance organizations for future maintenance [86].
- **Evolution and Maintenance Management Model: Retirement (EM3: Retirement)**, a process model to be used for retiring software systems [87].
- **Evolution and Maintenance Management Model: Release Management (EM3: Release Management)**, a process model to be used for managing releases [88].
- **Corrective Maintenance Management Model: Scheduled Problem Management (CM3: Problem Management)**, a process model to be used for managing scheduled software problems within corrective maintenance [89].
- **Corrective Maintenance Management Model: Emergency Problem Management (CM3: Emergency Problem Management)**, a process model to be used for managing emergency software problems within corrective maintenance [89].
- **Corrective Maintenance Management Model: Maintainers’ Education and Training (CM3: Maintainers’ Education and Training)**, a process model to be used for educating and training maintenance professionals [89].
- **Corrective Maintenance Management Model: SLA Management (CM3: SLA Management)**, a process model to be used for managing service level agreements [89].

Figure F1. List of EM3 processes that have been developed so far
Part III
Publications