Dissertation

VTT PUBLICATIONS 759

Jukka Kääriäinen

Towards an Application Lifecycle Management Framework

VTT PUBLICATIONS 759

Towards an Application Lifecycle Management Framework

Academic Dissertation to be presented, with the assent of the Faculty of Science, University of Oulu, for the public discussion in the Auditorium L10, Linnanmaa, on April 1st, 2011, at 12 o'clock noon.

ISBN 978-951-38-7728-6 (soft back ed.)
ISSN 1235-0621 (soft back ed.)

ISBN 978-951-38-7729-3 (URL: http://www.vtt.fi/publications/index.jsp)
ISSN 1455-0849 (URL: http://www.vtt.fi/publications/index.jsp)

JULKAISIJA – UTGIVARE – PUBLISHER

VTT, Vuorimiehentie 3, PL 1000, 02044 VTT
puh. vaihde 020 722 111, faksi 020 722 4374

VTT, Bergsmansvägen 3, PB 1000, 02044 VTT
tel. växel 020 722 111, fax 020 722 4374

VTT Technical Research Centre of Finland, Vuorimiehentie 3, P.O. Box 1000, FI-02044 VTT, Finland
phone internat. +358 20 722 111, fax + 358 20 722 4374

Edita Prima Oy, Helsinki 2011

Jukka Kääriäinen. Towards an Application Lifecycle Management Framework. Espoo 2011. VTT Publications 759. 103 p. + app. 81 p.

Keywords application lifecycle management, global software development, software configuration management

Abstract

One recent effort to support the development and management of products is the concept of Lifecycle Management. Lifecycle Management approaches promise more systematic and efficient ways to support the development and management of complex products. Product Lifecycle Management (PLM) means the activity of managing a company's products across their lifecycles in the most effective way. The concept of Application Lifecycle Management (ALM), on the other hand, indicates the coordination of activities and the management of artefacts (e.g., requirements, source code, test cases) during the software (SW) product's lifecycle. The definition and understanding of both these concepts have been driven by tool vendors. This thesis focuses on ALM and, especially, the development phase of the SW lifecycle.

There are surprisingly few scientific efforts to define what ALM constitutes and scientifically reported experiences of the practical development and deployment of ALM solutions in an industrial context. ALM solutions tend to be complex, integrating different tools and practices that are used to produce and manage artefacts during the SW development lifecycle, and there is therefore an apparent need to support the development of such complex solutions for industrial contexts.

This thesis presents an effort towards an ALM framework that can be used to document and analyse an organisation's ALM solution and find improvement ideas for it. This effort began in 2006 and iteratively constructed and demonstrated a proposal for an ALM framework during a series of case studies. The current version of the ALM framework contains six principal elements of ALM, a description of the relations between the elements and an ALM element mapping to the Global Software Development (GSD) patterns in order to reveal how ALM may support GSD.

The evolving framework has been demonstrated in four industrial case studies and gradually refined based on the experiences gained from the studies. First, three case studies were carried out in the automation industry and a fourth case study in the telecommunications industry. This thesis presents the four case studies to the reader and explains the whole research process from the initial literature study, via four phases of constructing and demonstrating the evolving ALM framework, to a proposal for an ALM framework. Furthermore, the series of case studies revealed several experiences related to the application and improvement of an ALM solution in an industrial context. These experiences are also presented and discussed in this thesis.

Preface

This research was carried out at VTT as part of the European research projects called TWINS (Optimizing HW-SW Co-Design Flow for Software Intensive System Development), EVOLVE (Evolutionary Validation and Verification) and MERLIN (Embedded Systems Engineering in Collaboration) during 2006–2011.

I wish to thank my supervisors, Professor Samuli Saukkonen and Doctor Minna Pikkarainen, as well as the reviewers of the thesis, Professor Filippo Lanubile and Professor Juan Garbajosa. I would also like thank Mr. Antti Välimäki who provided valuable support and cooperation during the research work. Furthermore, I would like to thank the people in VTT and case companies who made this work possible.

I would like to give my gratitude to my family for all their support and understanding during my research.

Oulu, March 2011

Jukka Kääriäinen

List of original publications

Paper I	Kääriäinen, J. and Välimäki, A. *Impact of Application Lifecycle Management – a Case Study*, International Conference on Interoperability of Enterprise, Software and Applications (I-ESA). Berlin, German. March 25th–28th 2008. Mertins, K. et al. (Eds.): Enterprise Interoperability III – New Challenges and Industrial Approaches. Springer (2008), pp. 55–67.
Paper II	Kääriäinen, J. and Välimäki, A. *Get a Grip on your Distributed Software Development with Application Lifecycle Management*, International Journal of Computer Applications in Technology (IJCAT), InderScience Publishers, Vol. 40, No. 3 (2011), pp. 181–190.
Paper III	Kääriäinen, J. and Välimäki, A. *Applying Application Lifecycle Management for the Development of Complex Systems: Experiences from Automation Industry*, 16th European Conference, EuroSPI 2009, Alcala (Madrid), Spain, 2–4 Sept. 2009, Communications in Computer and Information Science. Volume 42. R.V. O'Connor et al. (Eds.): EuroSPI 2009, Springer, Berlin–Heidelberg (2009), pp. 149–160.
Paper IV	Välimäki, A., Kääriäinen, J. and Koskimies, K. *Global Software Development Patterns for Project Management*. 16th European Conference, EuroSPI 2009, Alcala (Madrid), Spain, 2–4 Sept. 2009, Communications in Computer and Information Science. Volume 42. R.V. O'Connor et al. (Eds.): EuroSPI 2009, Springer, Berlin–Heidelberg (2009), pp. 137–148.
Paper V	Heinonen, S., Kääriäinen, J. and Takalo, J. *Challenges in Collaboration: Tool Chain Enables Transparency Beyond Partner Borders*.

International Conference on Interoperability of Enterprise, Software and Applications (I-ESA), Madeira, Concalves R. et al. (Eds.): Enterprise Interoperability II, New Challenges and Approaches. Springer Science+Business Media (2007), pp. 529–540.

Paper VI	Kääriäinen, J., Eskeli, J., Teppola, S., Välimäki, A., Tuuttila, P. and Piippola, M. *Extending Global Tool Integration Environment Towards Lifecycle Management*, International Workshop on Information System in Distributed Environment (ISDE 2009), Vilamoura, Algarve-Portugal, November 2009, Lecture Notes in Computer Science (LNCS): 5872, Meersman, R. et al. (Eds.) On the Move to Meaningful Internet Systems: OTM 2009 Workshops, Springer (2009), pp. 238–247.

Contents

Abstract 3

Preface 5

List of original publications 6

List of symbols 11

1. Introduction 13
   - 1.1 Scope of research 16
   - 1.2 Research questions 17
   - 1.3 Contributions 18
   - 1.4 Structure of dissertation 18

2. Background 20
   - 2.1 Product Lifecycle Management 20
   - 2.2 Application Lifecycle Management 21
     - 2.2.1 The concept of Application Lifecycle Management 21
     - 2.2.2 Requirements management and traceability 23
     - 2.2.3 Configuration management 24
     - 2.2.4 Tool integration 25
     - 2.2.5 ALM tools 26
   - 2.3 Global Software Development 28

3. Research design 30
   - 3.1 Research approach 30
   - 3.2 Research methods 33
   - 3.3 Research process 35
     - 3.3.1 Case study design 35
       - 3.3.1.1 Cooperation with case companies 36
       - 3.3.1.2 Research planning 37
     - 3.3.2 Collecting evidence 37
     - 3.3.3 Analysis of collected data 39
     - 3.3.4 Reporting 41
   - 3.4 Research context 42

3.4.1 Case organisation 1: ALM cases ... 42
3.4.2 Case organisation 2: ToolChain case ... 43
3.5 ALM framework construction ... 44
3.5.1 Literature study and Phase 1 (autumn 2006–autumn 2007) ... 45
3.5.2 Phase 2 (spring 2008) ... 46
3.5.3 Phase 3 (autumn 2008–spring 2009) ... 47
3.5.4 Phase 4 (spring 2009–autumn 2009) ... 47
3.6 Summary ... 48

4. Towards an Application Lifecycle Management Framework ... 49
4.1 The need for an ALM framework ... 49
4.2 ALM framework ... 50
4.2.1 The principal elements of ALM ... 51
4.2.1.1 Creation and management of lifecycle artefacts element ... 52
4.2.1.2 Traceability of lifecycle artefacts element ... 53
4.2.1.3 Reporting of lifecycle artefacts element ... 54
4.2.1.4 Communication element ... 56
4.2.1.5 Process support element ... 56
4.2.1.6 Tool integration element ... 58
4.2.2 Description of relations between ALM elements ... 59
4.2.3 Mapping between ALM elements and GSD patterns ... 61

5. Contribution of the thesis ... 64
5.1 PAPER I: Impact of Application Lifecycle Management – A Case Study ... 65
5.2 PAPER II: Get a Grip on your Distributed Software Development with Application Lifecycle Management ... 67
5.3 PAPER III: Applying Application Lifecycle Management for the Development of Complex Systems: Experiences from the Automation Industry ... 70
5.4 PAPER IV: Global Software Development Patterns for Project Management (co-author) ... 73
5.5 PAPER V: Challenges in Collaboration: Tool Chain Enables Transparency Beyond Partner Borders (co-author) ... 74
5.6 PAPER VI: Extending Tool Integration Environment Towards Application Lifecycle Management ... 75
5.7 Summary ... 77

6. Discussion ... 79
6.1 Comparing the ALM framework with related research ... 79
6.2 Discussions on case experiences ... 83
6.3 Implications for research and practice ... 85
6.4 Evaluation of research validity and reliability ... 87
6.4.1 Construct validity ... 87
6.4.2 External validity ... 88
6.4.3 Reliability ... 88

7. Conclusions ... 90
7.1 Answers to research questions ... 90

7.2 Limitations and future research 91

References 94

Appendices

Papers I–VI

Papers I–VI are not included in the PDF version.
Please order the printed version to get the complete publication (http://www.vtt.fi/publications/index.jsp).

List of abbreviations

ALF	Application Lifecycle Framework
ALM	Application Lifecycle Management
ALME	Application Lifecycle Management Environment
CAD	Computer Aided Design
CASE	Computer Aided Software Engineering
CDE	Collaborative Development Environments
CMMI	Capability Maturity Model Integration
CVS	Concurrent Versioning System
CM	Configuration Management
DSP	Digital Signal Processing
EDM	Engineering Document Management
ERP	Enterprise Resource Planning
GSD	Global Software Development
HW	Hardware
IDE	Integrated Development Environment
IPSE	Integrated Project Support Environment

IS	Information Systems
ISEE	Integrated Software Engineering Environment
ISF	Integrated Software Factory
PBI	Product Backlog Item
PDM	Product Data Management
PLM	Product Lifecycle Mangement
RM	Requirements Management
SBI	Sprint Backlog Item
SCM	Software Configuration Management
SEE	Software Engineering Environment
SME	Small and Medium Enterprises
SPI	Software Process Improvement
STD	State Transition Diagram
SW	Software
TFS	Team Foundation Server
TFT	Thin-Film Transistor
VSS	Visual SourceSafe
VSTS	Visual Studio Team System
VTT	Technical Research Centre of Finland

1. Introduction

The ability to produce quality products on time and at competitive costs is important to any industrial organisation. Nowadays, products and their development are becoming more complex. Software is just one part of the complex product, which also comprises hardware (Krikhaar et al., 2009; Broy, 2006; Crnkovic et al., 2003; Stevens et al., 1998). With the term *complex products,* the author denotes systems, such as, televisions, automation systems, cars, base stations, etc. An example of a complex product is the automation system produced by the automation industry, which comprises several interconnected subsystems realised with hardware (HW) and/or software (SW). Another example of a complex product comes from the field of telecommunications. The development of a radio base station involves several design parties including HW development, HW-related SW development and application SW development with solutions for the air interface and the base station itself (Blyler, 2002; Ronkainen et al., 2002). The growing role of SW has been recognised in complex products, for instance, the leading role of SW in automotive development and innovation (Pretschner et al., 2004), the rising costs of SW development in the development of industrial robots (Crnkovic et al., 2003) and the increasing role of SW in base stations (Kääriäinen et al., 2004). Furthermore, globalisation forces companies to operate in global development environments (Damian and Moitra, 2006), complicating communication, coordination and control, etc. (Carmel and Tjia, 2005). Nowadays, companies seek systematic and more efficient ways to support the development and management of complex products in a global development environment.

One response to these challenges is the rise of the concepts Product Lifecycle Management (PLM) and Application Lifecycle Management (ALM). According to literature, better lifecycle management provides benefits for companies, such as shorter time to market and improved product quality (Sääksvuori and Immonen, 2004; Stark, 2005; Brandao and Wynn, 2008). The product lifecycle covers phases that relate to the development of the product before it is delivered as well as phases relating to its use, support and retirement (Stark, 2005). The product lifecycle is therefore the period from *cradle to grave* (Stark, 2005). This is the same for software. ‘Every software product has a lifetime – it starts its life as a response to a user’s need or as a new product concept and ends up being obsolete’ (Leon, 2000). The software product’s lifecycle therefore covers activities related to the development of the product and its operation and maintenance (Leon, 2000).

In literature and among tool vendors, the term *Product Lifecycle Management* (PLM) has been widely discussed (for instance, in Sääksvuori and Immonen, 2004, and Stark, 2005). Stark (2005) provides the definition: ‘PLM is the activity of managing a company’s products all the way across their lifecycles in the most effective way’. A PLM solution can comprise various tools and databases used to create and manage product-related data such as requirements management (RM), configuration management (CM), computer aided design (CAD), enterprise resource planning (ERP), computer-aided software engineering (CASE), etc. One challenge of using several databases is that their functionality and managed information often overlap (Svensson, 2003; Crnkovic et al., 2003). This means that the data are duplicated in several databases, making traceability and maintenance of the data more complicated. Integration is needed to enable these databases to work together (Stark, 2005).

In parallel with the concept of PLM, the concept of *Application Lifecycle Management* (ALM) has emerged to mean from a SW engineering point of view:

‘The coordination of development lifecycle activities, including requirements, modelling, development, build and testing, through: 1) enforcement of processes that span these activities; 2) management of relationships between development artefacts used or produced by these activities; and 3) reporting on progress of the development effort as a whole.’ (Schwaber, 2006)

A common characterisation is that ALM does not focus on any specific lifecycle activity but helps to keep all the lifecycle activities synchronised: ALM is a thread that ties the development lifecycle together (Rossberg, 2008; Schwaber, 2006). In this thesis, the term *artefact* is used to denote any item produced or used during the development of the software, for instance, a requirement, task, code, test case, etc. The term *application* is used in this thesis to denote any type of software that may also be part of a larger system, for instance, reporting software in automation systems. ALM as a concept has mostly been discussed in professional publications, e.g., in Doyle (2007), Doyle and Lloyd (2007), Schwaber (2006) and Shaw (2007). In many scientific articles, the term ALM has only been treated in a cursory way or discussed from the point of view of ALM tools, without further analysis of the ALM concept, (e.g., Dearle, 2007; Heindl et al., 2007; Moore et al., 2007; Medina-Dominguez et al., 2007). Weiß et al. (2009) and Göthe et al. (2008) argue that the whole concept of ALM is somewhat unclear and that definitions are driven by the existing or planned marketing strategies of tool vendors. Rossberg (2008) states that people often equate ALM with operations and maintenance without the development phase. The author of this thesis also found that the whole concept of ALM is scattered. During the first industrial case study (Paper I), a need emerged for an improved understanding of the elements of ALM. This understanding is needed to create an ALM framework that can be used to document and analyse the current state of ALM solutions in an organisation and to detect ALM elements that may need to be improved. It is a misconception that just implementing a new information management system will automatically solve lifecycle management problems (Stark, 2005). To support the improvements to the ALM solution in an organisation, its current ALM solution first needs to be documented and analysed.

As product development is a global activity in one case study company presented in this research, a need also emerged to study how ALM solutions apply to a Global Software Development (GSD) context (Paper I). This need has also been detected in a recent systematic review by Šmite et al. (2010). They argue that there is a need for empirical evaluation of methods and tools for global software engineering in an industrial context.

This thesis presents interpretive case studies that have resulted in a proposal for an ALM framework. The framework presents elements of ALM, relations between the elements and an analysis of how ALM may be applied to a GSD context. The research shows how the framework is used to document and analyse an organisation's ALM solutions. With the concept of the *ALM solution*, the author denotes tool-based solutions with related practices directed at supporting the coordination of SW development activities and the management of SW development artefacts. The organisation's ALM solution therefore contains ALM tools and related practices tailored based on the needs of the organisation. The ALM framework has been used to support practical ALM improvement efforts in one case study company. Furthermore, it has been used to analyse the tool integration environment in order to find out how the environment should be improved to meet the needs of ALM.

1.1 Scope of research

This thesis constructs a proposal for an ALM framework. The framework is constructed and demonstrated iteratively in industrial case studies. The research presented in this thesis focuses on the development phase of the product. Other phases are defined outside the scope of this research. In the future, research should be extended towards other lifecycle phases to cover also the operation and maintenance phase.

The roots of the ALM tools lie in the history of Configuration Management (CM) (Weatherall, 2007). Software Configuration Management (SCM) solutions usually form the foundations of ALM infrastructures, providing storage, versioning and traceability between all the lifecycle artefacts (Schwaber, 2005). The author of this thesis has several years' experience of configuration management. This affects the fact that the ALM elements presented in this research have many similarities to SCM activities and SCM terminology. ALM is treated as a supporting discipline for software development by providing the coordination for SW development activities and the management of lifecycle artefacts (e.g., requirements, tasks, source code, test cases) during SW development.

Even though ALM as a concept is not well defined, many concepts related to ALM have been under active research for a while, for instance, tool integration and traceability. Since ALM as a concept has not been under active research, the research has been oriented in a horizontal way, i.e., analysing ALM as a whole, instead of with a vertical orientation, i.e., analysing each ALM-related concept in-depth. The vertical in-depth research of each ALM-related concept is outside the scope of this research. The key background concepts related to ALM are introduced in Section 2 however. A detailed vertical study requires a significant effort, and it should be considered a topic for future research in order to construct a more elaborated version of the framework. A horizontal approach allowed the construction of an ALM framework proposal to understand what constitutes ALM and to support ALM solution documentation and analysis in a case study company in practice.

The research forms an ALM framework proposal that can be considered as the very first, and essential, step for more definitive support for the documentation and analysis of an organisation's ALM solution (i.e., current state analysis of the organisation's ALM solution). The current state analysis has been recognised as an essential step of process improvement. See, for instance, the use of the process improvement framework for the improvement of configuration management in Taramaa (1998). The purpose of the ALM framework and the improvement models therefore overlaps. The elaboration of the framework, the further validation of the framework and the study of its relation to the related standards, improvement models and concepts (e.g., tool categories) are outside the scope of this research but should be the subject of future research.

1.2 Research questions

The concept of ALM is not well defined (Weiß et al., 2009, and Göthe et al., 2008) and a need emerged to support the improvement of ALM solutions in industry (Paper I). The starting point for ALM improvement in an organisation is the ability to document and analyse the current state of the solution. The documentation and analysis of the ALM solution are difficult, as only a few scientific publications relate to the concept of ALM. This thesis therefore aims to contribute to a better understanding of the concept of ALM. It also aims to motivate discussion on industrial-based challenges and practical ALM experiences in scientific forums. Based on the discussion, the research question of this thesis is as follows:

RQ1: *How can the documentation and analysis of the ALM solution be facilitated in an organisation operating in a Global Software Development environment?*

As the question is broad, this research aims to provide a first step towards a definitive answer to the research question, i.e., a proposal for an ALM framework. In order to support the documentation and analysis, there is a need to understand what constitutes ALM, i.e., what the main elements of ALM and the relations between them are. Furthermore, the global development environment as a context creates challenges for ALM. There is therefore a need to understand how the ALM solution may support an organisation operating in such a context. The main question is therefore divided into the following sub-questions:

- RQ1.1: *What are the main elements of Application Lifecycle Management?*
- RQ1.2: *What are the relations between the elements of Application Lifecycle Management?*
- RQ1.3: *How can Application Lifecycle Management be applied to a Global Software Development context?*

1.3 Contributions

The contribution of the thesis is presented in detail in Papers I–VI included in this thesis. The papers are introduced in Chapter 5.

The main contributions of this thesis are as follows:

Scientific contribution:

1. A proposal for an ALM framework based on literature and case studies that increase the understanding of ALM as a concept
2. The demonstration of the evolving ALM framework for documenting and analysing ALM solutions in four case studies.

Practical contribution:

1. Contribution to the understanding of ALM and support for ALM solution improvement work for a case study company that operates in a global development environment
2. Industrial experiences from the gradual improvement of practical ALM solutions in a case study company
3. Contribution related to the analysis of a global tool integration environment (ToolChain) in order to support its further development towards application lifecycle management.

1.4 Structure of dissertation

This thesis consists of seven chapters. Chapter 1 includes an introduction to the research topic, the motivation for the research, the scope of the research, the research questions and research contributions.

Chapter 2 presents a background to this thesis, dividing the treatment into product lifecycle management, application lifecycle management and global software development.

Chapter 3 explains the research design including a research approach, research methods and a research process. The chapter then presents the case contexts and explains how the ALM framework was constructed.

Chapter 4 introduces the current version of the proposed ALM framework. The ALM framework contains the ALM framework elements, a description of the relations between the elements and the mapping between the ALM framework elements and the GSD patterns.

Chapter 5 introduces the key contribution of this thesis by presenting the papers related to this research.

Chapter 6 discusses the results of the research. First, the chapter discusses how the proposed ALM framework and case experiences reflect the existing research and experience reports. The chapter then discusses the implications and evaluates the research process.

Chapter 7 concludes the research. It includes the answers to the research questions, the limitations of the research and a discussion of future research angles.

2. Background

This chapter sets out the background to the thesis. The chapter is divided into sections discussing the concepts of Product Lifecycle Management (PLM), Application Lifecycle Management (ALM) and Global Software Development (GSD).

2.1 Product Lifecycle Management

Product Lifecycle Management (PLM) ‘is the activity of managing a product throughout its lifecycle – from cradle to grave’ (Stark, 2005). PLM arose from the need to bring together previously disparate and fragmented activities, systems and processes to overcome problems such as information becoming lost, customer requirements being misinterpreted and decisions not being coordinated (Stark, 2005). Atos Origin (2003) gives a short introduction to the history of PLM. The article states that PLM has its roots in the design and engineering activities of a company. When Computer Aided Design (CAD) was taken into use in companies, the systems were needed to help manage vast amounts of CAD data. In these systems, namely Engineering Data Management (EDM) and Product Data Management (PDM), the focus was on file management and version control. These systems were extended with a better user interface, more links to CAD systems and better handling of product structures. In the late 1990s and early 2000s, the focus was on closer links with Enterprise Resource Planning (ERP) systems and collaboration with partners (e.g., subcontracts). Web-based systems have recently made the connection to partners’, suppliers’ and companies’ own dispersed units easier. Among vendors, this gradual evolution has led to the use of the term Product Lifecycle Management (PLM). PLM is a systematic method that attempts to control the product data by controlling and steering the process of creating, handling, distributing, and recording product-related data (Sääksvuori and Immonen, 2004). PLM forms an interlinked solution that can comprise various disparate systems used to create and manage product-related data, such as requirements management (RM), configuration management (CM), enterprise resource planning (ERP), computer-aided software engineering (CASE), etc. Abramovici (2007) expects that in the future, PLM will offer better support for integration of multi-disciplinary products, not just for mechanical and electrical products. Abramovici (2007) argues that PLM research is still at a very early stage, and past approaches to PLM have mostly been driven by software providers and large user companies.

According to Sääksvuori and Immonen (2004), PLM offers benefits such as shorter time to market and improved product quality. PLM also offers better opportunities to improve communication during product development and transfer of files between tools (Sääksvuori and Immonen, 2004).

2.2 Application Lifecycle Management

This section discusses the background to Application Lifecycle Management. First, it looks at the concept of ALM. Next, it presents the issues that relate to ALM: requirements management and traceability, configuration management and tool integration. Finally, ALM is discussed from a tools point of view.

2.2.1 The concept of Application Lifecycle Management

In the past few years, the concept of Application Lifecycle Management has emerged to indicate the coordination of activities and the management of artefacts (e.g., requirements, source code, test cases) during the lifecycle of software products. ALM as a concept is quite new (Weiß et al., 2009) and has mostly been discussed in professional articles, e.g., Doyle (2007), Doyle and Lloyd (2007), Schwaber (2006), Shaw (2007) and Chappell (2008). In the articles, the ALM concept has been discussed from various viewpoints, for instance:

1) model-driven development (Carrillo and McKorkle, 2008)
2) complex systems development (Doyle, 2007; Doyle and Lloyd, 2007)
3) technology and ALM tools (Goth, 2009; Kravchik, 2009; Shroff et al., 2005; Moore et al., 2007; Medina-Dominguez et al., 2007), or
4) only treated in a cursory way (Dearle, 2007; Heindl et al., 2007).

According to Schwaber (2006), companies are aware of ALM as a concept but do not understand what it actually means. The ALM tool providers have their own definitions of ALM that reflect their backgrounds and marketing strategies (Pirklbauer et al., 2009). It is nonetheless difficult to find articles that discuss the concept: what constitutes ALM? This may affect the interpretation that the whole concept of ALM is unclear and driven by tool vendors (Weiß et al., 2009, and Göthe et al., 2008). Pirklbauer et al. (2009) argue that a wide range of tools are labelled as ALM tools due to the loosely defined scope of ALM. Similarly, PLM approaches have mainly been driven by tool providers and large user companies (Abramovici, 2007). Rossberg (2008) found that people often equated ALM with operations and maintenance, without development phases. In the literature study phase of this research, the author also found that the whole concept of ALM is scattered. In the first industrial study conducted at the automation company (presented in Paper I), a need emerged to better understand the elements of ALM and create an ALM framework that could be used to document and analyse the current state of ALM solutions in a target organisation and detect ALM elements that may need to be improved.

Most of the discussions relating to ALM as a concept are from professional publications: books and professional articles. There is a well-known professional article related to ALM from Schwaber (2006). She defines the three pillars of ALM to be traceability, process automation and reporting. An important viewpoint on ALM is that it does not focus on any specific lifecycle activity, but keeps all the activities synchronised (Schwaber, 2006). ALM is therefore a thread that ties the development lifecycle together from business needs to operations (Rossberg, 2008). The support for project management has also been recognised. According to Doyle (2007), a proper ALM tool provides strong support for project management by, for instance, providing an objective means to monitor project activities and generate real-time reports from project data. Schwaber (2006) states that ALM does not necessarily require tools. Traditionally, lifecycle activities have been handled partly by manual operations.

Even though there is no general agreement on the definition of ALM, many concepts that have been defined as important to ALM have already been studied for a long time. Doyle and Lloyd (2007), for instance, state that requirements management is important for ALM, and Schwaber (2006) introduces the idea that concepts such as traceability, process automation, reporting and tool integration relate to ALM. Furthermore, the basis of ALM comes from Software Configuration Management (SCM). According to Weatherall (2007), the ALM tools have their roots in Configuration Management and Integrated Development Environments (IDEs). Murta et al. (2010) and Schwaber (2005) present SCM tools as the usual foundations of ALM infrastructures.

Next, the issues that relate to ALM, such as requirements management, traceability, configuration management and tool integration are discussed based on the literature.

2.2.2 Requirements management and traceability

Requirements management and traceability were under active research in the 90s by, for example, Gotel (1995), Gotel and Finkelstein (1994), Sommerville and Sawyer (1997), Kotonya and Sommerville (1998), Ramesh and Dhar (1992), Ramesh and Jarke (2001), and Toranzo and Castro (1999). Sommerville and Sawyer (1997) present guidelines for different requirements engineering process phases. In their book, requirements management is a support process for requirements engineering. They define the principal concerns of requirements management as managing changes to agreed requirements, managing relationships between requirements and managing dependencies between the requirements document and other documents. Kotonya and Sommerville (1998) stress that requirements identification and storage are an essential pre-requisite of requirements management.

Requirements traceability (RT) refers to the ability to describe and follow the life of a requirement in a forward and backward direction (Gotel, 1995). A comprehensive study on reference models for requirements traceability is presented by Ramesh and Jarke (2001). More generally, traceability deals with the relations between any lifecycle artefacts. Gills (2005) presents a summary of the survey on traceability models in industrial projects. The survey summarises industrial experiences, item types and traceability models used in industry. Gills (2005) also presents a traceability model, based on the survey data, with the most typical items and relations. His study shows that each organisation has its own needs and terminology for traceability. This leads to the need for traceability tailoring (Dömges and Pohl, 1998). Espinoza and Garbajosa (2008) have approached the problem of project-specific traceability requirements with traceability metamodels that include a basic set of items (concepts, structures) for project-specific extensions.

2.2.3 Configuration management

Configuration Management and, more specifically from a SW management point of view, Software Configuration Management (SCM) is a discipline that provides the processes and technologies to identify and control (configuration) items (Moreira, 2004). SCM as a discipline has been in existence for several decades (Estublier et al., 2005). It is an important concept for ALM, as SCM tools can be seen as the foundations of ALM infrastructures (Schwaber, 2005). The generic CM process contains the basic CM activities (configuration identification, configuration control (i.e., change management), configuration status accounting (i.e. reporting) and configuration audit) and CM planning (Buckley, 1996). SWEBOK (2004) extends the basic CM activities with *release management and delivery* activity, which refers to the distribution of a software configuration item outside the development activity. Among the CM activities, the configuration identification activity provides a basis for other CM activities (SWEBOK, 2004). Buckley (1996) has presented configuration management activities as a chronological process. In the process, previous steps form the basis of successive steps. Identification activities, for instance, are used to establish and maintain a definitive basis for control (i.e., Change Management) and status accounting (i.e., reporting). *Configuration management planning* provides mechanisms for planning and documenting the CM solution for a project. IEEE Std-828 (2005) assists in the planning of software CM by providing pre-structured templates for documenting the responsibilities and practices. The author has studied the factors affecting the realisation of software configuration management in Kääriäinen (2006) and configuration management from a complex systems point of view in Kääriäinen et al. (2004).

Estublier (2000) divides the basic functionality of the SCM tools into three main classes: repository for components, help for engineers' usual activities and process control and support. Estublier et al. (2005) present *SCM as one of the software engineering domains in which process support has proven to be most successful*. Schwaber (2005) present that customisable process templates in process-centric SCM tools provide ability to implement different processes for different projects. Software configuration management tools are typically file-based, meaning that the granularity of the management is the file (version). For instance, the requirements document may contain several requirements as paragraphs. SCM tool can treat this document as a single aggregate object that contains all the requirements (Macfarlane and Reilly, 1995). If a requirements specification is prepared as one document, it is possible to manage different versions of it, but it is not possible to control the requirement items separately (Crnkovic at al., 1999). This means that the assignment and maintenance of traceability information are more difficult compared with the fine-grained management of requirements in which each requirement is treated as its own object. Nowadays, requirements management tools manage atomic requirements in a requirements database and, therefore, allow the management of relationships between them. The challenge, however, lies in mechanisms to handle the traceability of lifecycle artefacts that reside in other databases, such as, test cases, test data, design elements, source code, etc. This has been a deficiency of configuration management tools (Kolawa, 2006), and ALM tools need to fix this.

2.2.4 Tool integration

Tool integration has been under active research for a while. Wicks and Dewar (2007) present a study related to tool integration and, based on the results, propose a new research agenda for tool integration. As tool integration has mainly been studied from a technological viewpoint, they propose a more business-oriented approach to future tool integration research. Wasserman (1989) defines tool integration as follows:

'tool integration is intended to produce complete environments that support the entire software development lifecycle.'

From the ALM point of view, tool integration should therefore produce integrated environments that support the entire SW development lifecycle. Pederson (2006) defines an integrated environment to mean that users can easily move from one function to another without having to work with multiple, disconnected tools and manually integrate data between these tools. Tool integration facilitates a productive development environment by allowing the user to launch tools and transfer information easily between different tools. The integration can be achieved in different ways (for instance, integration types in Crnkovic et al., (2003) and Sääksvuori and Immonen (2004)). With regard to the development of complex systems, the research related to the integration of the management of HW and SW development artefacts has been reported by, for instance, El-khoury et al. (2005), Crnkovic et al. (2003) and Krikhaar et al. (2009). From the ALM point of view, the contribution presented by Booch and Brown (2003) is interesting. They introduced the vision of a 'frictionless surface' provided by Collaborative Development Environments (CDE). In this vision, there are a number of points of friction in the daily life of the developer that hinder effective operation, and CDEs may remove these. These friction points relate to issues such as insufficient work product collaboration and problems maintaining effective group communication, including knowledge and experience, project status and project memory. The concept of CDE means (Booch and Brown, 2003):

‘virtual space wherein all the stakeholders of a project – even if distributed by time and distance – may negotiate, brainstorm, discuss, share knowledge, and generally labor together to carry out some tasks, most often to create an executable deliverable and its supporting artefacts.’

2.2.5 ALM tools

Schwaber (2006) and Shaw (2007) have analysed the main vendors’ approaches to ALM and divided them into:

- <u>Single vendor platform</u>: Vendors define their own ALM interoperability frameworks and expect practitioners and other vendors to build integrations for that platform.
- <u>Multi-vendor platform</u>: The framework is developed in an open-source community, and practitioners can help drive requirements, influence the direction of the framework and even participate in the development project.
- <u>Single repository</u>: This approach expects vendors to build a complete set of ALM tools using a single repository to support traceability and cross-discipline reporting.

Even though the above list stresses the importance of the multi-vendor platform, each approach has its benefits. According to Doyle and Lloyd (2007), for instance, a central repository makes it easier to produce a relevant set of related metrics that spans the various development phases of the lifecycle. Shaw (2007), on the other hand, argues that the only approach that makes sense for ALM practitioners is the multi-vendor platform.

Examples of commercial ALM tool providers are Polarion (Polarion ALM), Microsoft (Visual Studio with its ALM core – Team Foundation Server, TFS) and Rally Software (Rally). ALM tools basically provide development environments that integrate various lifecycle applications such as requirements management, project management, configuration management and design and development tools. Pesola et al. (2008) state that one limitation of existing ALM tools is that many of them lock a company into one vendor and limit the choice of tools for different project phases. In some cases, the company may need to replace a number of development tools because a particular ALM tool does not integrate with all the existing tools. This may require a large amount of effort and money.

Multi-vendor platforms can be seen as solutions that combine specialised independent systems (Murta et al., 2010). The limitations of today's technology make it difficult to implement these platforms (Murta et al., 2010). Efforts towards multi-vendor platforms include Eclipse and ALF projects (Eclipse, 2010). Eclipse Platform offers good support to extend its functionality with plug-ins (Eclipse, 2010). The Application Lifecycle Framework (ALF) is a project that aims to provide a logical definition of the overall interoperability business process (Eclipse, 2010). This technology aims to handle the exchange of information from one tool to another, the business logic governing the sequencing of tools in support of the application lifecycle process, and the routing of significant events as the tools interact. This sounded like a very promising framework to support truly open tool-independent ALM. The project failed to gain the support of significant vendors, and it was terminated in 2008 (Kravchik, 2009).

The term Software Engineering Environment (SEE) is interesting from the point of view of ALM tools. ISO/IEC 15940 (2006) defines SEE as:

'provides automated services for the engineering of software systems and related domains (e.g., project management, process management, etc.).'

'includes the platform, system software, utilities and CASE tools installed.'

ALM tools can therefore be seen as tools that belong to this category. According to Ruiz-González and Canfora (2004), SEE has also been known by other names: IPSE (Integrated Project Support Environment), ISEE (Integrated Software Engineering Environment), CASE tools coalition, Federated CASE tools and ISF (Integrated Software Factory).

Comprehensive, well-integrated ALM tools are not just aimed at traditional plan-based product development. Goth (2009) states that the market for ALM tools for agile development has recently been booming. Even agile-based development companies working in global development environments need to consider ALM tools to increase visibility during development. In some cases, the companies are therefore replacing existing proprietary, office-based solutions with integrated ALM tools (Goth, 2009).

2.3 Global Software Development

Nowadays, product development is often distributed over multiple sites (Damian and Moitra, 2006). Active research has been conducted into global software development for some time and has been published as, for instance, a special issue in *IEEE Software* (Damian and Moitra, 2006). Practical industrial experiences of global SW development have been reported by, for example, Lane and Ågerfalk (2009), Ebert and De Neve (2001), Battin et al. (2001), and Herbsleb and Grinter (1999). GSD has been studied from an agile methods point of view, for instance, by Jalali and Wohlin (2010), Sutherland et al. (2007) and Ramesh et al. (2006). Portillo-Rodríguez et al. (2010) present their survey on tools to support global software development. They define desirable features for GSD tools and classify tools according to ISO/IEC 12207 standard processes.

There are many challenges to global working, such as, communication, coordination and control breakdown (Carmel and Tjia, 2005). The benefits of global development are reported, such as, reduced development costs and leveraging time-zone effectiveness (Conchúir et al., 2006). The benefits of global development are not self-evident (Conchúir et al., 2006). Conchúir et al. (2006) present a study focusing on the potential benefits of GSD, as stated in peer-reviewed literature. Their study shows that readers should be cautious about the GSD benefits claimed in literature. The benefits are not self-evident and there are many associated risks. Recently, Šmite et al. (2010) presented a systematic review of the empirical evidence on global software engineering. Their conclusion is interesting and argues that the ‘global software engineering field is still in an immature state with a lack of empirical evaluation of methods, techniques and tools in an industrial context.’

ALM’s promise to support distributed development has been discussed by Doyle and Lloyd (2007). Doyle and Lloyd (2007) argue that geographically distributed development introduces communication and coordination challenges for ALM. They maintain that one possibility is to use a central secure repository with acceptable network performance and implemented work procedures to provide real-time information about changes and task assignments to support work in a distributed development environment. Other articles also claim that modern information technology offers means to overcome geographical barriers caused by globalisation. Experiences reported by, for instance, Battin et al. (2001), Ebert and De Neve (2001), and Herbsleb and Grinter (1999) contain descriptions that refer to the usage of central repositories and distributed configuration management solutions to support global software development. Jiménez and Piattini (2009) have studied problems and solutions in distributed software development using a systematic review. Their results indicate the importance of solutions such as communication mechanisms, and the use of version control and knowledge sharing by maintaining, for instance, a product/process repository.

In parallel with the research presented in this thesis, Mr. Antti Välimäki has studied the best practices for project management in a global development environment (Välimäki and Koskimies, 2006; Välimäki and Kääriäinen, 2007; Välimäki and Kääriäinen, 2008; Paper IV). In this study, Mr. Välimäki described a proposal for GSD process patterns (for patterns as a description technique, see, for example, Coplien and Harrison (2005)) that indicates solutions for GSD challenges. The proposed patterns are presented in Paper IV. These patterns have been collected in parallel with ALM research from the same organisation as the ALM case data presented in this thesis. This approach allowed for an in-depth understanding of the ALM framework and the GSD patterns while analysing how the elements of the ALM framework relate to the GSD patterns (Paper IV and Paper VI). The patterns represent the project management viewpoint of the GSD. The analysis is therefore also limited to this perspective.

3. Research design

This section presents the design for this research, including the research approach (Section 3.1) and methods (Section 3.2) adopted. The research process (Section 3.3.) introduces the steps of the case study method into the research. The research context (Section 3.4) introduces the background to the teams and infrastructure in the case organisations. Finally, the iterative ALM framework construction and demonstration are presented as chronological research phases (Section 3.5).

The ALM framework construction and demonstration cover four cases. The first, three cases (Papers I, II and III) are referred to from here on as the 'ALM cases'. The fourth case (Paper VI) is referred to as the 'ToolChain case'. The ALM cases were carried out in an automation company referred to as 'case organisation 1'. The ToolChain case was carried out in a telecommunications company and is referred to as 'case organisation 2'.

3.1 Research approach

The objective of this research was to describe the elements of ALM and the relations between the elements that can be used to document, analyse and provide improvement ideas for the organisation's practical ALM solution. This was done iteratively by constructing an ALM framework and demonstrating its usage in cases for which the aim was to document and analyse practical ALM solutions in an industrial context. The ALM framework was refined iteratively based on experiences gained from the cases. Therefore,

the cases have increased the author's understanding of the concept of ALM by providing a practical viewpoint from the real users of the ALM solutions:

- *which elements are adequate in an ALM framework*
- *how the ALM framework elements are positioned compared with each other, and*
- *iteratively during the case studies helped to find out how to document, in practice, an ALM solution and its related experiences as a current state document.*

This knowledge can be used when documenting and analysing organisations' ALM solutions in future cases.

The research carried out in the Papers I–VI can be examined through the Information Systems (IS) research framework by Braa and Vidgen (1999). Braa and Vidgen (1999) present the IS research framework for the organisational laboratory (Figure 1). They state that positivism and interpretivism are the basic approaches in research. Positivism is about 'making reliable predictions and explanations, while interpretivism is concerned with making a reading of a situation in order to gain understanding' (Braa and Vidgen, 1999). They further argue that in both of these approaches 'the researcher is making an intervention, despite aspirations to being an objective outsider'. Similarly, Walsham (2006) argues (from the viewpoint of interpretive research) that 'continued involvement in the field situation, regardless of the starting position, can push the researcher towards a more involved stance'. He refers to a longitudinal case study presented by Walsham and Sahay (1999) in which they had a moral imperative to provide direct advice for field personnel in return for the time and effort they put into the research. Braa and Vidgen (1999) present the IS research framework as constituting three dynamics: positivism, interpretivism and intervention. The framework represents the intended research outcomes (understanding, change, prediction) and approaches that result in the outcomes (interpretation, intervention, reduction). In their framework, positivism is positioned in the lower left-hand corner with the outcomes of predictions and explanations. The interpretive approach in the lower right-hand corner relates to understanding and, on the top, a change with an interventionary approach.

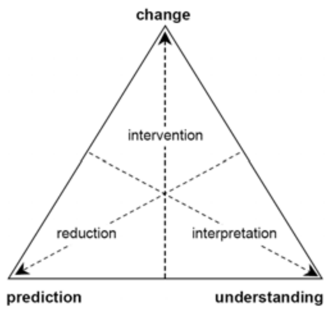


Figure 1. IS research framework (Braa and Vidgen, 1999).

Runeson and Höst (2009) present four types of research as follows:

- Exploratory – finding out what is happening, seeking new insights and generating ideas and hypotheses for new research
- Descriptive – portraying a situation or phenomenon
- Explanatory – seeking an explanation of a situation or a problem, mostly but not necessary in the form of a causal relationship
- Improving – trying to improve a certain aspect of the studied phenomenon.

From a research methods point of view, Braa and Vidgen (1999) argue that understanding is typically achieved with case studies. Similarly, Runeson and Höst (2009) state that an interpretive case study relates to understanding, i.e., descriptive and exploratory research types. Interpretive studies start out from initial theoretical frameworks of previous knowledge (Walsham, 1995). The work then continues as an iterative process of data collection and analysis with initial theories being expanded, revised or abandoned. An important aspect of interpretive research is that ‘interpretive researchers are not saying to the reader that they are reporting facts; instead they are reporting their interpretations of other people’s interpretations’ (Walsham, 1995).

The research presented in this thesis has an interpretive stance. The construction and the demonstration of the ALM framework have been carried out iteratively starting from the ALM framework version constructed based on the literature and refining it based on experiences gained from industrial case studies while applying the ALM framework version to document and analyse practical ALM solutions. The data collected during the case studies primarily represent the interpretations of the industrial respondents and interviewees. Case studies have therefore been used in this research to increase the understanding of ALM in an industrial context from the viewpoint of real users of ALM solutions. These have contributed to a better understanding of ALM as a concept and resulted in refinements to the ALM framework. To support the analysis of the case data, it was also necessary to observe and demonstrate practical ALM solutions directly.

In the ALM cases, the author acted as an outside observer while also providing the case results presented in Papers I–III for case organisation 1 and therefore affected the ALM development by helping the organisation to understand and improve its ALM solutions. The author has therefore been involved in the ALM improvement work of a case organisation. The intervention was used as an approach to provide benefits for the case organisation because it opened for case studies for several years. A similar moral imperative occurred in the longitudinal case study presented by Walsham and Sahay (1999).

3.2 Research methods

The previous section provides rationale for the selection of a case study as a research method to collect experiences iteratively from case organisations. Even though the case study is widely used as a research method, it is not trouble-free. Yin (2003) argues that perhaps the greatest concern has been the lack of rigour of case study research. Yin (2003) continues to say that there is a lack of methodological text providing investigators with specific procedures to be followed. Runeson and Höst (2009) have recently presented guidelines for case study research in software engineering. They present a practical checklist that can be used during the case study process to ensure the quality of the study. The article provides a valuable insight into important aspects of the case study that can be used to examine the research in this thesis.

In this research, the ALM framework has been constructed and demonstrated iteratively during a series of case studies (four) from 2006 until 2009. The first three case studies form a longitudinal case study in which two SW teams from one organisation were studied. A longitudinal case means studying the same case at two or more different points in time (Yin, 2003). It therefore provided an opportunity to follow the way the ALM solutions of case organisation 1 evolved over time.

Yin (2003) defines a case study as ‘an empirical inquiry that:

- investigates a contemporary phenomenon within its real-life context, especially when
- the boundaries between phenomenon and context are not clearly evident.'

A case study method is therefore characterised as the researcher wanting to cover contextual conditions with a potentially high relation to the phenomenon under investigation (Yin, 2003). A set of weaknesses of the case study method is presented as four issues:

1. Access to a suitable organisation may be difficult (Collis and Hussey, 2003).
2. There may be challenges to decide on the delimitations of the study (Collis and Hussey, 2003).
3. The unit of analysis has a history and a future that will influence the understanding of the present (Collis and Hussey, 2003).
4. Generalisability of the results is difficult and studies may take too long, with results that are massive, unreadable documents (Yin, 2003).

In this research, these issues were treated as follows. Issue 1 was treated through research cooperation between research and industrial organisations in the European research project (TWINS, 2010). The representatives of case organisations 1 and 2 were willing to open their organisations to research as part of the research project. The delimitations of the studies (issue 2) were agreed with case organisation 1 at the research planning phase of the ALM cases. In the ToolChain case, the boundaries were also agreed beforehand (e.g., tool set-up, demonstration project/group, aims of the project, product under analysis and data collection procedures). A history perspective (issue 3) is important in all the cases. The representatives of case organisation 1 compared the features of the new ALM solution with the old solution during discussions. Therefore, respondents might be highlighted in their answers tool features that were missing from the old solution. In the ToolChain case, the third version of the ToolChain implementation was analysed against the ALM framework and demonstrated in the telecommunications company. The ToolChain was initially intended to visualise traceability information and integrate different tools used in the development lifecycle and therefore lacked many ALM features. With regard to issue 4, Yin (2003) answers the question of generalisability: 'case studies, like experiments, are generalisable to theoretical propositions and not to populations or universes'. The time period and documentation are not a problem in this study. The research related to the European research project lasted several years and thus provided a good long-term environment in which to carry out the study. The documentation was kept as simple as possible, and the main aim of the research was to construct an ALM framework that could be used to document and analyse the organisation's ALM solution.

The literature essentially distinguishes four main phases of the case study: case study design, collecting evidence, analysing the collected data and reporting (Yin, 2003; Collis and Hussey, 2003; Runeson and Höst, 2009). Each of the phases is discussed next.

3.3 Research process

This section discusses the research process, which comprises the steps of the case study described at the end of the previous section. Walsham (2006) argues that it is important for interpretive researchers to maintain good access to appropriate organisations for the fieldwork. Open and constant communication that enabled successful long-term cooperation with the representative of the case study company is also highly valued by the author. Cooperation with the case companies will be therefore discussed in a separate subsection in the case study design section (Section 3.3.1.1). The process of data collection and analysis is often parallel (Miles and Huberman, 1994) or incremental to allow for the collection of complementary data (Runeson and Höst, 2009). This was also perceived in this research. For instance, working sessions with the development manager of case organisation 1 and demonstrations of Microsoft's Team Foundation Server (TFS) at VTT represented activities that contributed to both and were needed to obtain complementary data and a deeper understanding of the interpretations of respondents and interviewees.

3.3.1 Case study design

This section is divided into sections that introduce cooperation with case companies and research planning.

3.3.1.1 Cooperation with case companies

Runeson and Höst (2009) argue that even though trust is of foremost importance in the case study, there should be explicit measures to prevent problems. They further state that agreements should preferably be handled through a contract between the researcher and the individual participant. The ALM cases and ToolChain case were carried out under non-disclosure agreements between the industrial party and VTT and thus all the data and results intended for public dissemination were reviewed and agreed with the industrial parties.

According to Walsham (2006), for the fieldwork, the interpretive researcher needs to consider a style of involvement and a way to gain and maintain access to the organisation. In the ALM cases, the author cooperated closely with the development manager of case organisation 1, who had his own research interests and was therefore encouraged to carry out and disseminate the case studies properly. The development manager of case organisation 1 had the opportunity to check and ask for clarifications on issues related to case data that remained unclear. This was important as the case data contain interpretations by the respondents and interviewees. The author contributed to the ALM improvement work by providing ALM documentation and improvement ideas for case organisation 1. In case organisation 1, the decisions for improvement work were made in project meetings or retrospective meetings. The cooperation with the development manager of case organisation 1 covered research planning, collection and analysis of case data as well as reporting and publication. The intensity of cooperation reflects, for instance, the total number of workshops and working sessions arranged between VTT and case organisation 1 during the period January 2009 to June 2009. These sessions were mainly arranged using remote collaboration tools such as screen sharing and conference phone. There were 27 sessions and workshops during this period. The project managers of the case projects participated actively in ALM improvement work in the case study company (see Paper II) and were therefore willing to cooperate with the author and development manager of case organisation 1 to provide their interpretations of the ALM.

In the ToolChain case, the author was an outside observer, but the case was facilitated by the fact that the contact person in case organisation 2 was working in the same research project over several years with the authors of Paper VI. The authors of Paper VI therefore represent four organisations that share the same view and objectives for the ToolChain case. So-called ‘Finn meetings’ were used to plan and monitor case activities. The Finn meetings were arranged on a regular basis for the Finnish partners of the European research project. All the Finnish partners (three industrial partners and VTT) participated in the ToolChain case.

3.3.1.2 Research planning

Research planning was carried out in cooperation with industrial partners of the cases. In the ALM cases, the research planning was carried out with the development manager of case organisation 1. There were excellent bases for close research cooperation as both the development manager of case organisation 1 and the author of this thesis are studying for a PhD. At the beginning of the research, the objectives of the overall research were defined from the research and practice point of view for both PhD students. This was guided by the literature study and the needs of case organisation 1. This ensured industrial relevance. Suitable SW teams were found for the case studies. Here, the help and cooperation of the representative of the case study company is the most important. The selection was carried out based on availability (Runeson and Höst, 2009) rather than by selecting teams from a number of teams in case organisation 1. The delimitations were agreed with case organisation 1. It was noted that as SW teams that produce SW for a complex multidisciplinary product do not operate in a vacuum, it is also necessary to gather data about the interfaces for system development. The questionnaire and interviews were selected as basic methods for data collection. These were later complemented with a TFS demonstration at VTT, tool demonstrations in case organisation 1 and discussions with the development manager. In the ToolChain case, the research planning was also carried out in cooperation with industrial partners in Finn meetings. The setting was more isolated than in the ALM cases and there was therefore no need to consider interfaces to other operations in case organisation 2. The aims of the demonstration project, product under analysis and data collection procedures were agreed beforehand.

3.3.2 Collecting evidence

Triangulation is important in empirical research (Runeson and Höst, 2009; Yin, 2003). This is particularly obvious when relying primarily on qualitative data.

Data triangulation (Yin, 2003) was used in both the ALM cases and the Tool-Chain case.

In the ALM cases, the sources of data were a questionnaire for project managers and project members, interviews of project managers (three times), remote working sessions with the development manager, demonstrations of management systems (tools) in a case study company and a practical demonstration of TFS at VTT. The questionnaire was organised according to the initial version of the ALM framework. A motivation letter and questionnaire were planned and reviewed in cooperation with the development manager of case organisation 1 to ensure that the questions were expressed in such a way that they were interpreted in the same way by the researchers and respondents (see, for example, Runeson and Höst, 2009). There were a total of about 20 project personnel in the two SW teams. The questionnaire was sent to 14 persons with experience of the previous solution and the new ALM solution. They were therefore able to estimate the impacts of the new ALM solution on their daily work. The questionnaire was sent by e-mail with the motivation letter. The response rate was 50%. The respondents were asked about the current practices related to ALM, opinions on how the introduced ALM solution had affected daily work compared with previous solutions and opinions about things that were important to efficient application lifecycle management. They were also asked how the distributed development environment affected ALM.

The interviews were semi-structured (Runeson and Höst, 2009) with predefined questions, and they served as guidance or themes on what to discuss and provided a checklist for the interviewer to make sure that all the issues were covered. The first interview round was organised according to the initial version of the ALM framework and carried out during the field visit at the main site of case organisation 1. The second interview round was organised according to the second version of the ALM framework and was carried out using a remote connection (conference phone, screen sharing). The third interview round was carried out by the development manager of case organisation 1 by checking what had changed (and why) compared with the ALM current state analysis from the previous round.

During the first field visit, the author also familiarised himself with the products of the case study company. The ALM system and the other databases used in the case projects were also demonstrated to the author. The tool demonstrations showed the kind of data that was stored in the management systems and the kinds of features that were actually used. The TFS demonstration at VTT allowed us to understand the features and capabilities of the TFS and thus contributed to the case data. This complementing case data enabled the author to see and understand the actual use and concepts of the information management systems. Yin (2003) specifies this type of evidence as a physical artefact. Discussions with the development manager were used to analyse the case data and to collect complementary information that was missing from the case data. The development manager had the opportunity to ask the project managers directly for clarifications and additional information about ALM.

In the ToolChain case, we used a questionnaire and workshop to collect experience information from the case study company and the ToolChain implementation itself as a source of information for the ToolChain's ALM and GSD analysis. The questionnaire was organised according to the ToolChain tools and ALM framework elements. There was also an open question on any comments relating to the solution. After the questionnaire, a workshop was organised that presented and discussed the results of the questionnaire. This session clarified and documented issues that remained unclear after the questionnaire. The industrial (from three organisations) and VTT participants in the workshop session represented specialists in SW development, testing and software process improvement (SPI).

3.3.3 Analysis of collected data

The raw data from the case studies in the ALM cases contained questionnaire responses, interview and demonstration notes, working session notes and answers (notes, e-mails) from project managers and the representative of case organisation 1. As the discussions during the interviews were quite free, the raw data in the interview notes needed to be organised under relevant topics for analysis. The raw data from the questionnaire and interviews were coded (Runeson and Höst, 2009) and organised according to the ALM framework. Each answer or comment was associated with the corresponding ALM framework element. The analysis of rich case data turned out to be challenging. The raw data that referred to the tool- and organisation-specific terminology were challenging and demanded knowledge from different disciplines, e.g., requirements management and configuration management. In configuration management, for instance, the term *baseline* (Leon, 2000) is used to indicate the configuration of the software at a discrete point in time. In different tools, the term baseline can refer to, for instance, a label (TFS, VSS) or tag (CVS). Another example is *changeset* as a concept. Changesets relate to the change-oriented model of configuration management (Zeller, 1997). In the change-oriented model, related changes, which may involve several components (files), can be grouped into changesets to ensure that they can be applied as a single entity (Zeller, 1997). The TFS uses changesets, and it is possible to assign work items, e.g., bug fixing tasks, to the changeset that implements the task. This allows traceability from the bug to the related changeset to retrieve the file versions associated with the changeset (that fixes a bug). This traceability information can be used to, for example, review the old change (retrieve file versions related to the bug fix, associated comments, etc.). The case data also contained many company-specific concepts and references to product development processes. Similarly, Gills (2005) found in his traceability survey that different companies use different concepts originating from local culture, standards and methodologies. The analysis was carried out in cooperation with the development manager of the case organisation 1 to understand organisation-specific terminology and meanings. The analysis was further supported from a tools point of view with the ALM tool demonstration. VTT acquired the Microsoft Visual Studio Team System (VSTS). The author was particularly interested in its ALM core, the Team Foundation Server (TFS). An MSc student installed the system and replicated a similar process configuration of the system as in case organisation 1. The student used VSTS at VTT to demonstrate process template tailoring and to program a test data database and reporting application (Vitikka, 2009; Pollari and Kanstrén, 2009). The ALM features of TFS were analysed. This helped us to understand the tool-specific claims and answers from the case data. With a better understanding of the terminology, backgrounds and tools, the author was able to reduce the raw data (Miles and Huberman, 1994), for instance, by simplifying and abstracting the data by finding similar answers or understanding the technical solutions that enable certain ALM features. Paper II shows a summary of the tables of the analysed interview and questionnaire data. The data were organised and described according to the solution description, advantages, challenges/improvements and distribution. As the tables were also used in publication, the form of the table was agreed with the representative of case organisation 1.

The analysis of the ToolChain case data was more straightforward. The author compared the ToolChain against the ALM elements of the ALM framework and documented the solution according to the framework, i.e., which features were included and how, and which features were missing. The questionnaire responses were presented, discussed and complemented in a workshop session, as presented in the previous section. The advantage of this session was two-fold: first, it could be used to collect additional information and second, it could be used to analyse and comment on the data.

Furthermore, the GSD patterns described in Paper IV were compared against the ALM framework elements (Paper IV and Paper VI) to discover which GSD patterns could be used to analyse the GSD features of the ALM solutions. The mapping was done in a virtual workshop session (conference phone, screen sharing) with Mr. Antti Välimäki who described a proposal for GSD process patterns in his research (Paper IV). In this session, each ALM element was checked against each GSD pattern. The aim was to determine which GSD pattern descriptions contained issues that also existed in some ALM elements. The result of this mapping is presented in Paper VI.

3.3.4 Reporting

To improve the quality of the study, Yin (2003) presents it as a good procedure to have the draft report reviewed by the participants and informants in the case. The results of the cases presented in this research have been reviewed by the contributors prior to reporting. The results and conclusions of the ALM cases have been reviewed by the development manager and project managers of case organisation 1. The results and conclusions of the ToolChain case have been reviewed by the demonstration group members and workshop participants.

Walsham (1995) presents issues that need to be reported in an interpretive case study. If they are compared with the public reporting of the case studies presented in this thesis, it can be noted that some issues could not be revealed. These include the details of the research sites (company name, geographical locations, details about the subsystems) and detailed hierarchical or professional information about the people who were chosen for this study. This information was declared classified.

According to Runeson and Höst (2009), there may be a different audience for the research and it may lead to different kinds of reports for different audiences. The cases presented in this thesis are reported for a professional and academic audience:

- Academic forums: all the papers (Papers I–VI) included in this thesis have undergone a full paper review. Paper IV received best paper award at the EuroSPI 2009 conference.

- Professional forums: MARTES-TWINS (ITEA) public European joint seminar (January 2008), 6th Annual VTT Software Engineering Seminar (November 2008), ITEA2 Symposium 2009 (October 2009) and 7th Annual VTT Software Engineering Seminar (February 2010).

3.4 Research context

This section presents the contexts of the cases. Both organisations produce complex multidiscipline products, meaning that the products comprise SW and HW subsystems.

3.4.1 Case organisation 1: ALM cases

Case organisation 1 is a company that operates in the field of the automation industry. The company is a global supplier of technology and services for a number of industrial fields, such as, power generation, oil and gas, recycling, and pulp and paper. The company has worldwide operations in, for instance, engineering, production, procurement and service business. The company designs, develops and delivers automation and information management application networks and systems, intelligent field control solutions, and support and maintenance services.

Product development is organised according to product lines. As it is no longer competitive to develop multiple products one at a time, the case study company has adopted a product platform approach. The product is therefore based on a product platform on which customer-specific features are configured. The company produces complex automation systems for which the SW is part of the system.

The company operates in a multi-site environment. The challenges of the global development environment therefore need to be resolved. Previously, the company's ALM solution for distributed development comprised several somewhat isolated databases to manage project-related data such as version control, document management and fault management systems. The geographic distribution together with increasing complexity and efficiency demands forced the company to seek solutions that were more integrated to coordinate distinct project phases and provide a centralised project database for all project-related data.

In practice, at the first stage, this involved the deployment of a commercial ALM tool configured with the Scrum process template.

The improvement in ALM focuses on two SW teams (referred to in this thesis as SW Team 1 and SW Team 2), each having several SW projects running in parallel. Depending on the workload, each team has about ten members, and the number of projects may vary from three to six. The projects are currently geographically distributed over several sites (two countries). Each project typically has fewer than ten project members as the reported appropriate size for agile projects. SW Team 1 produces a SW product that constitutes one part of the common automation product platform, whereas SW Team 2 produces SW products for specific industry segments. The teams have adopted the agile development method Scrum.

3.4.2 Case organisation 2: ToolChain case

The case study company operates in the field of telecommunications. In this organisation, the ToolChain was not used in actual product development but was used to measure and analyse the performance of the third-party portable embedded platform. The demonstration group in case organisation 2 was also motivated to test its analysis tool (Tuuttila and Kanstrén, 2008) as part of the ToolChain and to gain experience of the Eclipse-based ToolChain in its organisational context.

The platform under analysis was a mobile Linux computer for industrial purposes with a TFT touch screen and a multitude of interfaces (Espotel, 2010). The demonstration project consisted of three persons from case organisation 2. Two persons focused on the usage of the ToolChain as an integrated development environment for test data collection, management and analysis. The third person focused on multivariate analysis of instrumented test data. The demonstration project used the ToolChain configured to support test data collection, management and analysis. Platform monitoring was carried out using a testing device from case organisation 2. The selected ToolChain configuration (Eskeli, 2009) contained tools for requirements management, project management, test management, test data management, test analysis and configuration management. The following list enumerates the tools used in the demonstration project:

- Requirements Management: *Open Source Requirements Management Tool (OSRMT) for managing requirements and features*
- Project Management: *Trac is originally aimed at issue management. In this demonstration, it was used for project management (management of tasks)*
- Test Management: *TestLink is used for documenting and managing test cases*
- Test Data Management: *Probe database (ProbeDB) (Vitikka, 2009) is a tool for importing, processing, managing and exporting instrumented test data*
- Test Analysis: *The MultiVariate Analysis* (*MVA) solution (Tuuttila and Kanstrén, 2008) is intended for visualising multidimensional data. In this demonstration, it is used for the visualisation of instrumented test data*
- Configuration Management (CM): *Subversion is used for configuration management activities (storing source code, versioning, baselining).*

The demonstration project carried out one sequential analysis iteration that started from the requirements and task definition, followed by a test case definition. The SW was implemented for the instrumentation of the embedded platform. The test cases were executed and the instrumented data were collected and stored, and after setting visualisation and analysis parameters, they were transferred to the visualisation and analysis tool. The results of the performance analysis were given to a third-party company. The results can be used in future platform optimisation projects.

3.5 ALM framework construction

The main contribution of this research is the proposed ALM framework that can be used to document and analyse the organisation's ALM solutions. The ALM improvement work in the case studies was supported by the ALM framework that was used for documenting and analysing the ALM solutions of case organisation 1, finding improvement ideas for ALM solutions in the organisation and analysing the tool integration environment in the ToolChain case. Figure 2 presents the history of the ALM framework development. The demonstration of the

ALM framework and the collection of information for the framework refinements were carried out in case studies.

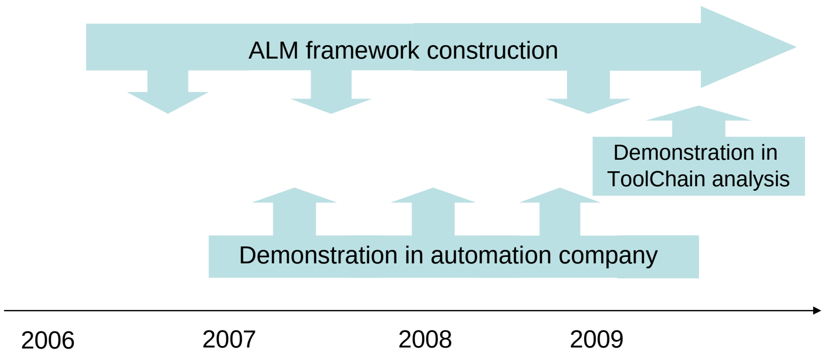

ALM framework construction
Demonstration in ToolChain analysis
Demonstration in automation company
2006
2007
2008
2009

Figure 2. ALM framework construction and demonstration.

The phases of the ALM framework construction and demonstration are shown in Figure 3. The figure shows how the research has been structured from iterative phases in which the ALM framework was constructed and demonstrated. During these phases, the author increased the understanding of ALM as a concept for himself. These phases will be explained next to reveal how the research process proceeded in chronological order.

3.5.1 Literature study and Phase 1 (autumn 2006–autumn 2007)

The first study related to ALM was carried out in 2006 and disseminated in Paper V. For each ALM case paper, the author also carried out complementary literature studies. Industrial cooperation relating to ALM started with a literature study at the end of 2006, discussions with case organisation 1, and a first case study in 2007. The first version of the ALM framework was based on Schwaber (2006) and adapted with professional and scientific publications and the ALM vendors' websites. In the first case study, the company's ALM solution version one was analysed and documented using the first version of the ALM framework. We used a two-step approach for the ALM-related data collection (questionnaire, interviews). The teams had two project managers running parallel projects. These two project managers were interviewed using semi-structured interviews while visiting the case study company's main site. The ALM system and other databases, which were also used in a case project, were demonstrated to the author. The results of the first case study were reviewed by the project managers and development manager of case organisation 1. This phase also resulted in refinements to the ALM framework. The results of the first research phase were presented in Paper I. The second ALM framework was introduced in Paper II.

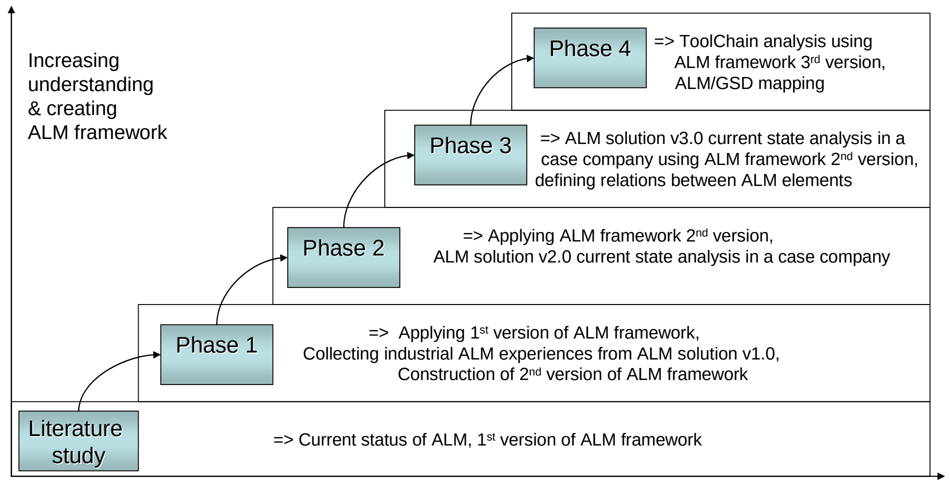

Increasing understanding & creating ALM framework
Phase 4
=> ToolChain analysis using ALM framework 3rd version, ALM/GSD mapping
Phase 3
=> ALM solution v3.0 current state analysis in a case company using ALM framework 2nd version, defining relations between ALM elements
Phase 2
=> Applying ALM framework 2nd version, ALM solution v2.0 current state analysis in a case company
Phase 1
=> Applying 1st version of ALM framework, Collecting industrial ALM experiences from ALM solution v1.0, Construction of 2nd version of ALM framework
Literature study
=> Current status of ALM, 1st version of ALM framework

Figure 3. Phases of the ALM framework construction and demonstration.

3.5.2 Phase 2 (spring 2008)

On second phase, beginning of 2008, the project managers were interviewed again using a semi-structured interview. The interview was structured using the second version of the ALM framework. In this phase, Microsoft’s Team Foundation Server ALM suite was also demonstrated at VTT. Data gained from the first and second phases were then analysed against the second version of the ALM framework and reported. In order to analyse how ALM could support the distributed development environment, the study also analysed the company’s ALM solution against the 3C model (Anderl et al., 2008). At this point, the results of the ALM case were reviewed by the project managers and development manager of the case study company. The results of the second research phase were presented in Paper II.

3.5.3 Phase 3 (autumn 2008–spring 2009)

The third research phase produced the current state analysis of ALM solution version three in the company and further elaborated the ALM framework by defining relations between framework elements. The motivation for this case study was the prior knowledge that both SW teams had successfully implemented improvements related to the requirements management of their ALM solutions and that these solutions were different. It was therefore interesting to find out why the teams ended up with different solutions and whether it was possible to describe these different solutions using the ALM framework. The data for this study were collected by updating the company's previous ALM description based on the information received from the complementary interviews of the project managers (i.e., what had changed and why the teams ended up with different solutions). In Phase 3, the ALM framework was extended by defining relations between the ALM elements. These relations were compiled from comparisons of practical ALM solutions and the ALM framework elements. This resulted in the third ALM framework version. Version three of the ALM framework description, the analysis results and the conclusions drawn were reviewed and complemented by the project managers and development manager of the case study company. The results of the third research phase were presented in Paper III.

3.5.4 Phase 4 (spring 2009–autumn 2009)

The fourth research phase studied the relations of the ALM framework and the GSD patterns and demonstrated these to analyse the global, Eclipse-based, tool integration environment named ToolChain (Paper V) in order to support its development towards lifecycle management in a global development environment. In this study, we compared ALM framework elements and GSD patterns and identified relations between the elements and patterns (Paper IV and VI). This information can be used to point out issues with the ALM framework elements that need to be solved in order to provide ALM support for the management of global SW development. The analysis of ToolChain was carried out against the ALM framework and ALM-related GSD patterns. ToolChain was also tested in a telecommunications company where the aim was to use the ALM framework as a frame to collect and analyse practical experiences while the case study company used ToolChain to support test data collection, storage, processing and analysis. The solution was introduced to the members of the demonstration group in the case organisation. One person from the company was responsible for setting up the ToolChain to its IT environment, and two of the persons tested the solution in the company. The experience data from the use of ToolChain were collected via a questionnaire. Responses were presented, discussed and complemented in a workshop session in cooperation with industry (three companies) and VTT, representing specialists in SW development and testing, and software process improvement (SPI). This study is presented in Paper VI.

3.6 Summary

This research has adopted an interpretive approach to collecting and analysing the interpretations of real users of ALM solutions. Data collection, analysis and documentation have been supported by the ALM framework that was iteratively constructed and demonstrated during this research. The main aim has been to define and demonstrate the ALM framework that can be used to document and analyse the organisation's ALM solution. The ALM framework has been constructed and demonstrated in four phases using the case study as a research method:

- Phase 1: the ALM-related, initial literature study and construction of the first ALM framework version, and using it in the first ALM case documentation (Papers I and V). Construction and description of the second ALM framework version (Paper II).
- Phase 2: applying the second ALM framework version in the second ALM case documentation. Analysing the case results against the 3C model. (Paper II).
- Phase 3: using the second ALM framework in the third ALM case documentation and constructing the third ALM framework version (Paper III).
- Phase 4: using the third ALM framework version to analyse the ToolChain (Paper V) tool integration environment from the ALM and GSD pattern (Paper IV) point of view and proposing features that extend the ToolChain towards application lifecycle management (Paper VI).

4. Towards an Application Lifecycle Management Framework

This section presents the proposed ALM framework that is the key contribution of the thesis. First, the section introduces the concept of a framework. Next, it introduces the proposed ALM framework, which comprises the elements of ALM, the relations of the ALM elements and the way the ALM elements relate to the GSD patterns.

4.1 The need for an ALM framework

A framework concept was selected in this research as the form of presentation for Application Lifecycle Management. Miles and Huberman (1994) define that ‘a conceptual framework explains, either graphically or in narrative form, the main things to be studied – the key factors, constructs or variables – and the presumed relationship among them’. According to Fisher (2007):

‘*In a conceptual framework, you put the concepts together as in a jigsaw puzzle. You work out how all the concepts fit together and relate to one another. The first stage of theorising identifies and clarifies concepts; the second stage concentrates on the connections and relationships between the concepts*.’

The concept of a framework has been used in several studies to describe the proposed solution to a defined research problem. Pikkarainen (2008) and Kanstrén (2010), for instance, have dressed their contribution to their research problems in the form of frameworks. In the research presented in this thesis, the purpose of the framework was twofold. First, the ALM framework was used as an evolving insight into the concept of ALM to help data collection, analysis and documentation during the case studies. Second, the main result of the research process was a proposal for the ALM framework that was constructed and demonstrated iteratively during the series of case studies. The ALM framework provides the means for scientists and practitioners to understand the complexity of ALM from an ALM solutions perspective, i.e., what the principal elements of ALM are. In addition, according to our case experiences, the framework can be used to document, analyse and find improvement ideas for an organisation's ALM solution.

The need for the ALM framework arose from the lack of clear means to document and analyse the organisation's ALM solution (Paper I). The whole concept of ALM was also found to be unclear and driven by tool vendors (also reported in Weiß et al., 2009, and Göthe et al., 2008). The construction and demonstration of the framework is presented in Sections 3.5 and 5.

4.2 ALM framework

ALM Framework version three (Paper III) complemented by the ALM/GSD mapping (Paper VI) forms the proposed ALM framework. <u>The proposed ALM framework comprises the following segments</u>:

- <u>The principal elements of ALM</u>: elements of ALM that form basic building blocks of ALM. They introduce a generic set of elements that typically occurs in some form in every SW development project.
- <u>Description of relations between ALM elements</u>: ALM elements form a complex in which elements complement each other and thus contribute to more complete ALM features.
- <u>Mapping between ALM elements and GSD patterns</u>: the mapping illustrates how ALM may provide support for defined GSD patterns that are proposed solutions for GSD problems.

Each of the segments presented above will be introduced in the following subsections.

The use of the framework for documentation and analysis of an ALM solution requires an understanding of the background of different ALM-related disciplines. In the cases, the author has used his prior knowledge of requirements management, software configuration management, and tool integration. Prior knowledge of agile methods also facilitated the documentation and analysis of ALM cases, as both SW teams used the SCRUM development method. The form of documentation evolved during the research process (see, for example, Paper II) and resulted in a preliminary structure of the ALM document template. According to the case studies, the following issues should be documented for each ALM framework element during the ALM documentation in an organisation under study:

- The solution's current state description: textual description for each element and additional descriptions for three elements (discussed in more detail in the next section):
    - Summary table providing an overview of lifecycle artefacts ('Creation and management of lifecycle artefacts' element).
    - Traceability model presenting lifecycle artefacts, tool/database information and relations between artefacts ('Traceability of lifecycle artefacts' and 'Tool integration' elements).
- Positive things related to the ALM framework element found in this study.
- What to improve related to the ALM framework element found in this study.

The summary of improvement ideas could be collected in a list at the end of the current state document. The detailed agreement about the description format should be agreed with the representative of an organisation under study.

4.2.1 The principal elements of ALM

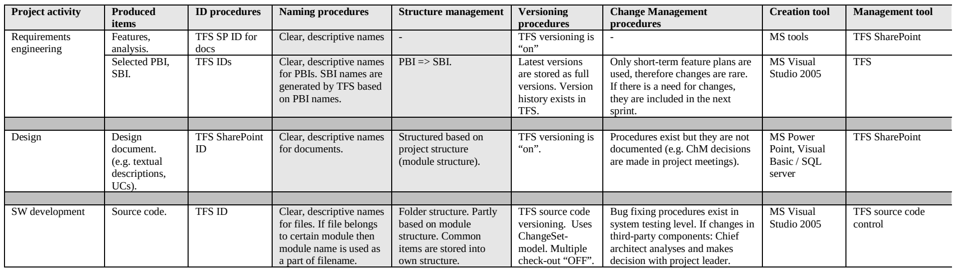
This section summarises the principal elements of the ALM framework:

- Creation and management of lifecycle artefacts (Section 4.2.1.1)
- Traceability of lifecycle artefacts (Section 4.2.1.2)
- Reporting of lifecycle artefacts (Section 4.2.1.3)
- Communication (Section 4.2.1.4)
- Process support (Section 4.2.1.5)
- Tool integration (Section 4.2.1.6).

4.2.1.1 Creation and management of lifecycle artefacts element

The ‘creation and management of lifecycle artefacts’ element is the foundation of ALM. It provides the mechanisms needed to create, store and manage lifecycle artefacts. First, in the ALM documentation and analysis, this element is used to document the development lifecycle activities (or disciplines as expressed in Shaw, 2007). Next, the element documents and analyses which data items are created, stored, identified, versioned and managed during the various activities of the development lifecycle as well as which tools are used to create and manage these artefacts. These artefacts are typically requirements, designs, source code, test cases, etc. These artefacts may also relate to, for instance, project management (e.g., task, sprint) and resources (e.g., team) that are needed to organise and follow the development work. The element is important as it provides basic information for further analysis with subsequent framework elements such as traceability, reporting and tool integration. The documentation and analysis of this element may reveal issues that are subject to improvement. Unambiguous identification of lifecycle artefacts, for instance, is an essential pre-requisite of information management activities (see, for example, Kotonya and Sommerville (1998) from a requirements management point of view and Whitgift (1991) from a configuration management point of view). Therefore, if, for example, the *product idea* item needs to be traced to the requirements, the essential prerequisite is that product ideas are stored and unambiguously identified so that these unique identifiers may be used to define traceability information (Sommerville and Sawyer, 1997).

As an example, in the ALM documentation the artefact information related to this element was collected in a table comprising each lifecycle artefact (Figure 4 shows an example): identification procedures, naming procedures, structure management, versioning procedures, change management procedures, tools that are used to create the artefact and tools/databases that are used to manage the artefact.



Project activity	Produced items	ID procedures	Naming procedures	Structure management	Versioning procedures	Change Management procedures	Creation tool	Management tool
Requirements engineering	Features, analysis.	TFS SP ID for docs	Clear, descriptive names	-	TFS versioning is "on"	-	MS tools	TFS SharePoint
	Selected PBI, SBI.	TFS IDs	Clear, descriptive names for PBIs. SBI names are generated by TFS based on PBI names.	PBI => SBI.	Latest versions are stored as full versions. Version history exists in TFS.	Only short-term feature plans are used, therefore changes are rare. If there is a need for changes, they are included in the next sprint.	MS Visual Studio 2005	TFS
Design	Design document. (e.g. textual descriptions, UCs).	TFS SharePoint ID	Clear, descriptive names for documents.	Structured based on project structure (module structure).	TFS versioning is "on".	Procedures exist but they are not documented (e.g. ChM decisions are made in project meetings).	MS Power Point, Visual Basic / SQL server	TFS SharePoint
SW development	Source code.	TFS ID	Clear, descriptive names for files. If file belongs to certain module then module name is used as a part of filename.	Folder structure. Partly based on module structure. Common items are stored into own structure.	TFS source code versioning. Uses ChangeSet-model. Multiple check-out "OFF".	Bug fixing procedures exist in system testing level. If changes in third-party components: Chief architect analyses and makes decision with project leader.	MS Visual Studio 2005	TFS source code control

Figure 4. An example from the 'Creation and management of lifecycle artefacts' element summary documentation from the ALM documentation.

4.2.1.2 Traceability of lifecycle artefacts element

The 'traceability of lifecycle artefacts' element provides the means to identify and maintain relationships between artefacts managed by the 'Creation and management of lifecycle artefacts' element. In the ALM documentation and analysis, this element is used to document and analyse the traceability of lifecycle artefacts. The purpose of this element is wider than the concept of traceability in requirements management. This element relates to the traceability of any artefacts produced during the development lifecycle (see, for example, the definition of ALM in Schwaber, 2006). It therefore facilitates reporting, change impact analysis and information visibility through the development lifecycle.

In order to understand and gain an overview of the traceability of the lifecycle artefacts in the cases presented in this thesis, traceability models were used to describe the associations between the lifecycle artefacts. The requirements traceability literature has used models to describe the relations between different items, for instance, in Ramesh and Jarke (2001), and Toranzo and Castro (1999). An example of a general traceability model constructed based on an industrial survey is presented by Gills (2005). This kind of presentation can be equipped with the direction of traceability (i.e., forwards – backwards, (Kotonya and Sommerville, 1998)). Furthermore, if database/tool-related information is embedded into the models (see Figure 5 as an example), it is possible to visualise which associations occur within a database and which are realised as some sort of references between different databases. The presentation allows much information to be described in quite a compact format. During the cases, it became evident that if there are many artefacts, associations and databases, the models easily become quite complex and difficult to read. The models therefore need to be explained to understand, for instance, the reasons for the association or the solution that implements the association (i.e., practical integration). This could be facilitated using a spreadsheet application, by simply documenting associations in the intersections of lifecycle artefacts in a matrix. This could be seen as an application of a traceability matrix (Sommerville and Sawyer, 1997) to document and explain the associations between different types of lifecycle artefacts in table format.

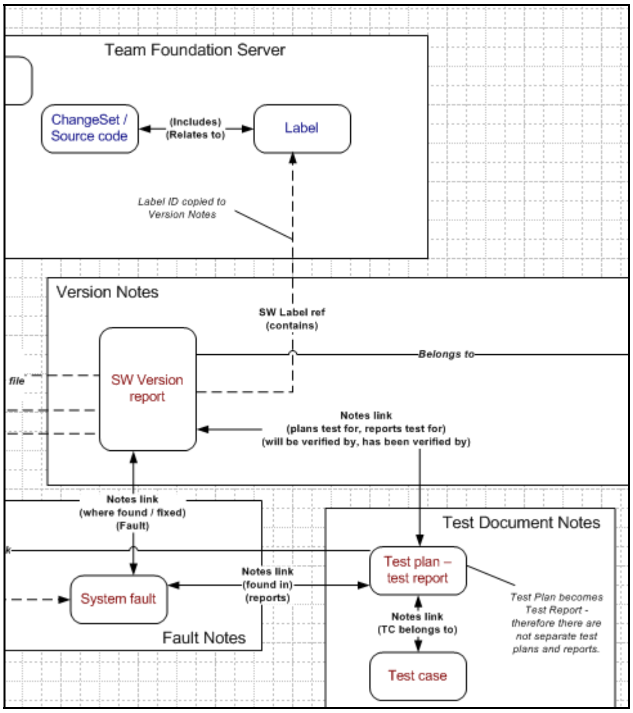
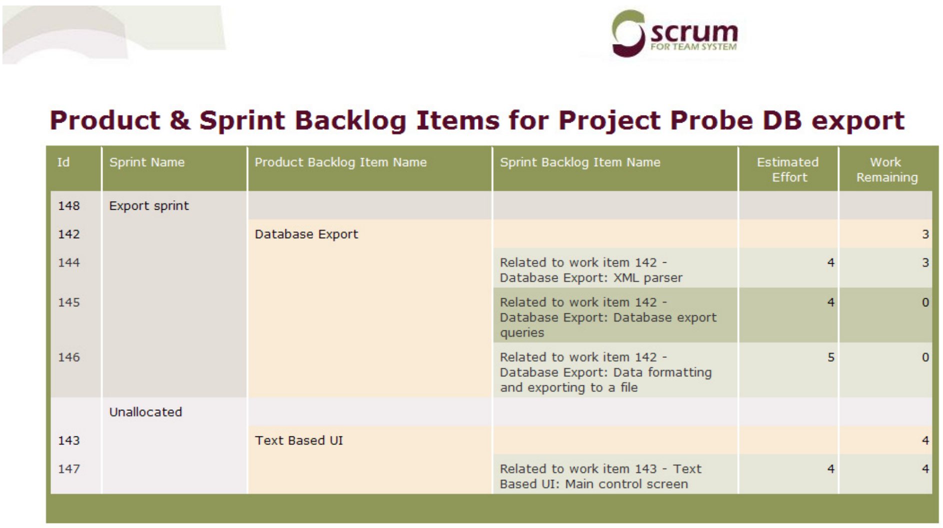

Team Foundation Server
ChangeSet / Source code
(Includes)
(Relates to)
Label
Label ID copied to Version Notes
Version Notes
SW Label ref (contains)
Belongs to
file
SW Version report
Notes link (plans test for, reports test for) (will be verified by, has been verified by)
Notes link (where found / fixed) (Fault)
Test Document Notes
Test plan – test report
Notes link (found in) (reports)
System fault
Fault Notes
Notes link (TC belongs to)
Test case
Test Plan becomes Test Report - therefore there are not separate test plans and reports.

Figure 5. An example of the documentation of traceability between lifecycle artefacts.

4.2.1.3 Reporting of lifecycle artefacts element

The ‘reporting of lifecycle artefacts’ element provides mechanisms to generate information about the status of lifecycle artefacts and the status of SW development in general. This is important, as reports are feedback mechanisms for project managers to see if the project is on schedule (Doyle and Lloyd, 2007). In the ALM documentation and analysis, this element is used to document and analyse how the solution supports reporting of the lifecycle artefacts. Reporting is acknowledged as an important part of configuration management (i.e., status accounting, see, for example, IEEE Std-828 (2005) and Buckley (1996)). According to Leon (2000) and Buckley (1996), the reports in configuration management can be divided into:

- standard reports: the generation of standard reports should be as automated as possible
- ad hoc (on-demand) reports: ad-hoc reports are needed more occasionally and the solution should provide the possibility to extract ad hoc queries from project data.



Product & Sprint Backlog Items for Project Probe DB export

Id	Sprint Name	Product Backlog Item Name	Sprint Backlog Item Name	Estimated Effort	Work Remaining
148	Export sprint				
142		Database Export			3
144			Related to work item 142 - Database Export: XML parser	4	3
145			Related to work item 142 - Database Export: Database export queries	4	0
146			Related to work item 142 - Database Export: Data formatting and exporting to a file	5	0
	Unallocated				
143		Text Based UI			4
147			Related to work item 143 - Text Based UI: Main control screen	4	4

Figure 6. An example of a TFS report.

In the cases, the reports used by the SW teams were listed (generated automatically or manually from the ALM solution). The templates and terminology of the reports are development-method and organisation specific. During the case studies, case organisation 1 therefore provided some examples from the reports generated by their solution. Furthermore, in order to understand reporting in the context of tools, the generation of reports and their communication in TFS (Figure 6) (e.g., using Project Portal and Team Explorer) were demonstrated at VTT (Vitikka, 2009).

4.2.1.4 Communication element

In the ALM analysis, the ‘communication’ element is used to document and analyse the kinds of means or practices that exist for communication in an ALM solution. This element includes, for example, data sharing and communication tools and practices in the development lifecycle. Communication can be divided into synchronous and asynchronous communication. Synchronous communication consists of communication forms such as meetings, phone, videoconference, chat, etc. Asynchronous communication consists of communication forms such as discussion forums, email, notifications, fax, common databases, etc. The type of communication required for a particular situation depends on the time/place matrix (Table 1) (Schlicter et al., 1998).

Table 1. Time – place matrix.

	Same time	**Different time**
Same place	1	2
Different place	3	4

The communication in quadrants 1 and 3 allow the use of synchronous communications (e.g., meetings, phone, videoconferences), while quadrants 2 and 4 only allow asynchronous communication (e.g., email, fax) (Meade, 2001). Different communication tools can be used separately or integrated into development environments.

In the cases, solutions used for synchronous and asynchronous communication were listed. The ALM tool vendors have expanded their tools towards communication. The TFS project portal, for example, enables the use of discussion forums and announcements/notifications for the SW team. An example of an asynchronous communication channel is the Report Portal in the Project Portal (see Figure 6).

4.2.1.5 Process support element

The ‘process support’ element provides means to support a company’s predefined processes (e.g., product development methods, change process, etc.). In the ALM analysis, this element is used to document and analyse the way the whole development lifecycle is supported by the ALM solution. Each organisation usually has its own SW development needs and therefore ‘methods should be tailored to the actual needs of the development context’ (Fitzgerald et al., 2003). It is thus also important that the ALM solution support the selected SW development methods and practices used in the organisation (or even project). ALM works in conjunction with selected SW development methods, providing support and automation for various activities such as a storage place for artefacts, information visibility and traceability. The configuration of an ALM tool therefore depends on the development methods. The agile development method, Scrum, for instance, operates with certain items (product backlog item, sprint backlog item, etc.) and requires certain reports (e.g., a sprint burn-down chart).

According to Estublier (2000), there are, technically, two techniques that can be used to provide tool support for processes: State Transition Diagrams (STDs) and Activity Centred Modelling. In practice, STD-based solutions provide the possibility to define the states and transitions between the states for an item type to support operation according to the company’s processes. It therefore describes the legal way for an item to evolve (Estublier 2000). In Activity Centred Modelling, the activity plays a central role, and models express the data and control flow between the activities (Estublier 2000). This type of modelling emphasises the work (or tasks) that needs to be done. A more advanced approach to process support is the process template. The IBM Rational Method Composer and Microsoft’s MSF are examples of this kind of utilities. IBM Rational Method Composer provides a process template and tools for application lifecycle management. Microsoft’s MSF provides process guidance that can be tailored to different variations, and VSTS can be configured to support these definitions (Guckenheimer, 2006).

In the cases, the mechanisms for process support were documented and analysed. In the ALM cases, the solutions for process support were mostly STD based, but TFS process templates that can be used to configure the whole ALM tool from requirements gathering to release represented an advanced approach to the process support. The use of process templates (Process Manager) and the tailoring of these templates using the Process Editor were demonstrated (Vitikka, 2009) (Figure 7) to provide an understanding of the tool capabilities and the effort required to process template tailoring. The ToolChain case represented different approach to process support – a kind of work guide (named ‘workflow support’ in the tool) that presented tasks with tool instructions to carry out the tasks the user has to follow to complete certain development steps (i.e. Activity Centred Modelling type of process support).

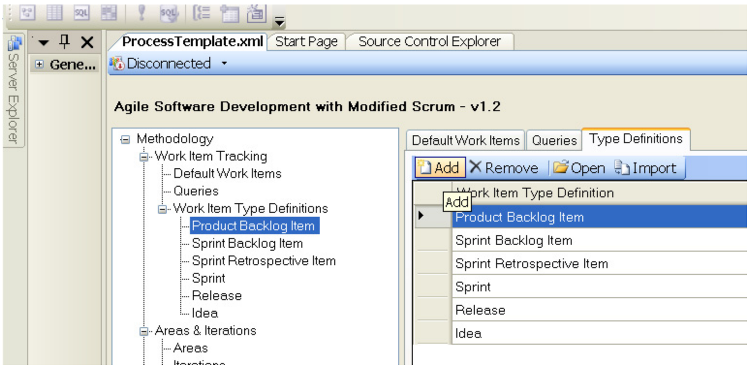


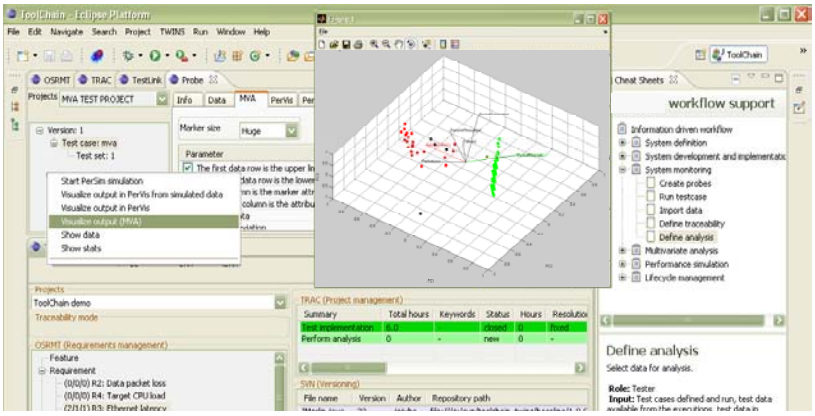
Figure 7. Process Editor for the modification of process templates.

4.2.1.6 Tool integration element

In the ALM analysis, the ‘tool integration’ element is used to document and analyse the kinds of tool integrations that exist in the development environment. There are different ways to handle integration, for instance, integration types presented in Crnkovic et al. (2003), and Sääksvuori and Immonen (2004).

In the cases, the tool integration was documented. In the ALM cases, the integration of project data management systems (i.e., CM, test document DB, fault DB, etc.) was presented as part of the traceability model comprising items grouped according to the databases and traceability information between the artefacts (see Figure 5). In the ToolChain case, the integrated development environment comprised a set of tools from requirements management to test data management integrated into the Eclipse tool integration environment (Figure 8).

Even though tool integration is recognised as an important aspect of ALM (e.g., Schwaber, 2006; Schwaber, 2005) it is not self-evident. Some operations that need to be carried out just a couple of times during the development project can also be supported using manual import/export routines, i.e., loosely coupled integration with a proper process (Paper III).


ToolChain - Eclipse Platform
File Edit Navigate Search Project TWINS Run Window Help
OSRMT TRAC TestLink Probe
Projects MVA TEST PROJECT
Info Data MVA PerVis
Version: 1
Test case: mva
Test set: 1
Marker size Huge
Parameter
The first data row is the upper li
Start PerSim simulation
Visualize output in PerVis from simulated data
Visualize output in PerVis
Visualize output (MVA)
Show data
Show stats
ToolChain
Cheat Sheets
workflow support
Information driven workflow
System definition
System development and implementatio
System monitoring
Create probes
Run testcase
Import data
Define traceability
Define analysis
Multivariate analysis
Performance simulation
Lifecycle management
Projects
ToolChain demo
Traceability mode
OSRMT (Requirements management)
Feature
Requirement
(0/0/0) R2: Data packet loss
(0/0/0) R4: Target CPU load
(2/1/1) R3: Ethernet latency
TRAC (Project management)
SVN (Versioning)
File name Version Author Repository path

Summary	Total hours	Keywords	Status	Hours	Resolution
Test implementation	6.0	-	closed	0	fixed
Perform analysis	0	-	new	0	-

Define analysis

Select data for analysis.

Role: Tester
Input: Test cases defined and run, test data available from the executions, test data in

Figure 8. The ToolChain integrated development environment for managing and visualising test data.

4.2.2 Description of relations between ALM elements

When the practical implementations of ALM in the case study company and the ALM elements in the framework were compared, it was possible to define relations between the elements (see Section 5.3 for the initial presentation of the relationships). The key ALM elements are ‘creation and management of lifecycle artefacts’, ‘traceability of lifecycle artefacts’, ‘reporting of lifecycle artefacts’ and ‘communication’. These elements have a hierarchical relationship to each other (Fisher, 2007) in which the elements form successive activities in which previous activity is needed to accomplish successive activities. Similarly, Buckley (1996) has presented configuration management activities as a chronological process. In the process, previous steps form a basis for successive steps. For instance, identification activities are used to establish and maintain a definitive basis for control (i.e., Change Management) and status accounting (i.e., reporting).

The remaining ALM elements, namely ‘process support’ and ‘tool integration’, focus on setting the effective development environment for the ALM key elements. Figure 9 explains the relations between the ALM elements. In the figure, the ALM elements are expressed as boxes, with arrows between the boxes showing the relations (read the direction of the to-arrow). Four ALM key elements are presented on the right-hand side inside the box.

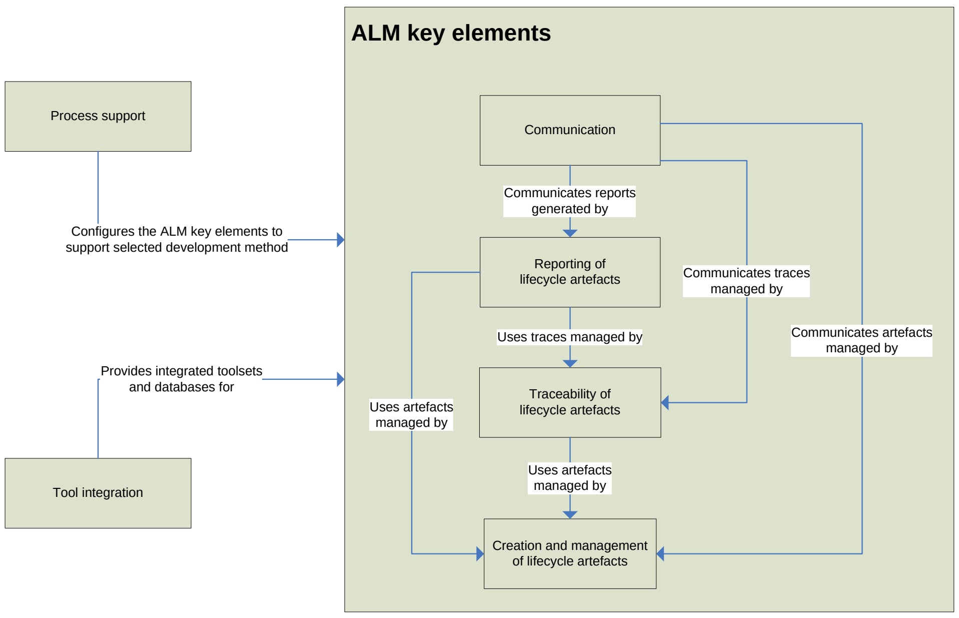

ALM key elements
Process support
Configures the ALM key elements to support selected development method
Provides integrated toolsets and databases for
Tool integration
Communication
Communicates reports generated by
Reporting of lifecycle artefacts
Communicates traces managed by
Communicates artefacts managed by
Uses traces managed by
Traceability of lifecycle artefacts
Uses artefacts managed by
Uses artefacts managed by
Creation and management of lifecycle artefacts

Figure 9. Relations between different elements of ALM explained.

The ‘creation and management of lifecycle artefacts’ element is the foundation of ALM. The information collected and managed by this element is needed for, for instance, traceability and reporting activities.

The ‘traceability of lifecycle artefacts’ element provides a means to identify and maintain relationships between managed lifecycle artefacts and, therefore, facilitates reporting, change impact analysis and information visibility through the development lifecycle. The ‘traceability of lifecycle artefacts’ element therefore *uses artefacts managed by* the ‘creation and management of lifecycle artefacts’ element in order to identify and maintain traces between the artefacts for traceability purposes.

The ‘reporting of lifecycle artefacts’ element utilises the managed lifecycle artefacts and traceability information to generate the necessary reports to support SW development and management. The ‘reporting of lifecycle artefacts’ element therefore *uses artefacts managed by* the ‘creation and management of lifecycle artefacts’ element and *uses traces managed by* the ‘traceability of lifecycle artefacts’ element in order to generate reports from this data.

The ‘communication’ element provides communication tools (e.g., chat) as well as channels for distributing information on product lifecycle artefacts, links and reports and thus facilitates product information visibility for the whole SW project. The ‘communication’ element therefore *communicates artefacts managed by* the ‘creation and management of lifecycle artefacts’ element, *communicates traces managed by* the ‘traceability of lifecycle artefacts’ element and *communicates reports generated by* the ‘reporting of lifecycle artefacts’ element.

The ‘process support’ and ‘tool integration’ elements are used to configure the ALM solution to support SW development procedures and facilitate a productive development environment by allowing the user to launch tools and transfer information easily between different tools and databases. The ‘tool integration’ element therefore *provides integrated toolsets and databases for* ALM key elements, and, on the other hand, the ‘process support’ element *configures the ALM key elements to support a selected development method* in the organisation.

An example in the TFS environment that reflects these relations is the generation of a ‘product backlog composition’ report for the Project Portal. The TFS Scrum process template contains a ‘product backlog composition’ report. The report collects managed Scrum items (PBIs, SBIs) as well as their relations to generate a report that presents PBIs and their related SBIs as well as their realisation-related information (hours). This report can then be made visible through a Project Portal that facilitates real-time information visibility via a web browser for the whole SW project.

4.2.3 Mapping between ALM elements and GSD patterns

ALM support for Global Software Development has been analysed using the GSD Project Management Pattern Language (Paper IV). This proposed Pattern Language includes 18 process patterns that have been found to be important in the area of project management in GSD. When these GSD patterns were compared with ALM elements in Papers IV and VI, it was noted that some patterns related to ALM elements (Table 2). Patterns named ‘Communication Tools’, and ‘Common Repositories and Tools’ related to ALM elements. Furthermore, ‘Use Common Processes’ related to an ALM element called ‘Process Support’. This shows that ALM databases may be used as solutions to meet the problems indicated in GSD patterns. Some other patterns relate indirectly to ALM elements. An ‘Iteration Planning’ pattern, for instance, uses ‘Communication Tools’ and ‘Common Repositories and Tools’ patterns to support synchronous communication and information visibility during planning meeting. These indirect relations are presented in Figure 10. Lines between the dashed line boxes and solid line boxes represent relations between ALM elements and GSD patterns. Lines between solid line boxes represent relations between GSD patterns and, therefore, indicate indirect relations between ALM elements and GSD patterns.

Table 2. Mapping between ALM elements and related GSD patterns (Paper VI).

ALM elements	Related GSD patterns	How GSD patterns cover ALM elements
Creation and management of lifecycle artefacts	Common Repositories and Tools (ID 07)	Global databases to support the management and visibility of lifecycle artefacts
Traceability of life-cycle artefacts	Common Repositories and Tools (ID 07)	Traceability of lifecycle artefacts in the GSD environment
Reporting of life-cycle artefacts	Common Repositories and Tools (ID 07)	Reporting of lifecycle artefacts and traces in the GSD environment
Communication	Common Repositories and Tools (ID 07)	Asynchronous communication (visibility of lifecycle artefacts)
	Communication Tools (ID 06)	Synchronous/asynchronous communication tools (e.g., net meeting, chat, conference phone, discussion forum)
Process support	Common Repositories and Tools (ID 07)	Process support features such as state models, workflows or process templates
	Use Common Processes (ID 12)	Common upper level GSD process and ability to tailor process support for a project or a team at site level
Tool integration	Common Repositories and Tools (ID 07)	In practice, a common repository can be a single central database or several integrated databases.

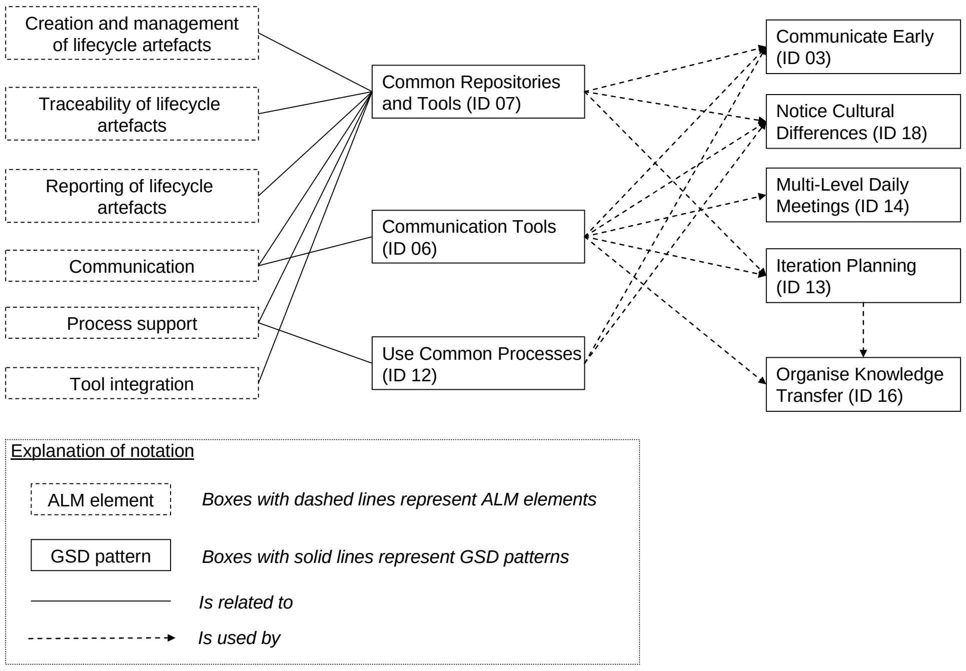

Creation and management of lifecycle artefacts
Traceability of lifecycle artefacts
Reporting of lifecycle artefacts
Communication
Process support
Tool integration
Common Repositories and Tools (ID 07)
Communication Tools (ID 06)
Use Common Processes (ID 12)
Communicate Early (ID 03)
Notice Cultural Differences (ID 18)
Multi-Level Daily Meetings (ID 14)
Iteration Planning (ID 13)
Organise Knowledge Transfer (ID 16)
Explanation of notation
ALM element
Boxes with dashed lines represent ALM elements
GSD pattern
Boxes with solid lines represent GSD patterns
Is related to
Is used by

Figure 10. Relations between ALM elements and GSD patterns.

5. Contribution of the thesis

The key contribution of the thesis is the construction and demonstration of an Application Lifecycle Management framework. Research phases presented in Section 3 form a path from the initial ALM literature study to the application of the third ALM framework version in ToolChain (global tool integration environment) case (Table 3).

Table 3. Research phases and papers related to the phases.

Research phases	Papers included in this thesis and how they contribute to different research phases
Literature study	**Paper V:** First ALM-related literature study of this research **Paper I:** ALM framework first version, first ALM case study (referred to as 'case study 1') using the first version of the ALM framework **Paper II:** Description of the second version of the ALM framework
Phase 1	
Phase 2	**Paper II:** Second ALM case study (referred to as 'case study 2') using the second version of the ALM framework. Analysing the ALM solution against a 3C model
Phase 3	**Paper III:** Third ALM case study (referred to as 'case study 3') using the second version of the ALM framework. Description of the third version of the ALM framework
Phase 4	**Paper: IV:** First analysis of GSD patterns and ALM elements **Paper: V:** ToolChain background study **Paper VI:** ALM/GSD mapping. Fourth ALM case study (referred to as 'case study 4', ToolChain case) using the third version of the ALM framework

Papers I, II, III and VI form the main storyline of the thesis. They correspond to the different phases of the research from the literature study through phases one, two, three and four. Papers IV and V are important to this thesis by contributing to it as follows. Paper V presents our first literature study related to ALM. Furthermore, it presents the background to VTT's ToolChain solution that is analysed in Paper VI from an ALM and GSD point of views. Paper IV presents the first analysis in which GSD patterns have been analysed against ALM elements. The paper also indicates that based on the study presented in the paper, ALM-related GSD patterns support the operation in a global development environment well.

The contribution of the papers to the research phases is explained in the next subsections.

5.1 PAPER I: Impact of Application Lifecycle Management – A Case Study

Paper I focuses on the key issue of the thesis – constructing and demonstrating the Application Lifecycle Management framework. The paper is based on data collected from case organisation 1 and the literature study of the author. The starting point for the ALM framework construction was that the term ALM was unclear (see, for example, Weiß et al., 2009) and that its improvement needed to be supported in case organisation 1. In this case study, the aim was to gather and analyse the first experiences when a company moves towards distributed ALM. From a research point of view, the aim was to create an ALM framework that can be used to analyse the current state of the ALM solution in a target organisation and to detect ALM elements that may need to be improved. This case study started a longitudinal case study in which the ALM framework was iteratively applied in case studies and refined based on the results of the case studies. The paper discusses the concept of ALM and constructs the first version of the ALM framework. The framework is based on information gathered from literature and the vendors' web pages. The elements of the *first version of ALM framework* are:

- Creation and management of project artefacts: How different data items are created, identified, stored and versioned in various phases of a project lifecycle? All project data should be shared securely and easily by all stakeholders. Team communication should be supported.

- Traceability of lifecycle artefacts: How is traceability in a project lifecycle handled? Traceability provides a means to identify and maintain relationships between artefacts and, therefore, facilitates reporting, change impact analysis and information visibility through the product lifecycle.
- Reporting of lifecycle artefacts: How does the solution support reporting in a project lifecycle? The solution should facilitate the gathering, processing and presentation of information related to process and configuration items for an organisation.
- Process automation and tool integration: How well do the tools support lifecycle processes and what kind of tool integrations are there? An ALM solution should support the procedures of the project and facilitate fluent data exchange and queries between various development and management tools.

When constructing the first version of the ALM framework, the main source was the description of three pillars of ALM by Schwaber (2006). Case data were collected from two SW teams by means of a questionnaire (project members and project managers) and interviews (project managers). The framework was used in the data collection, and in documenting and analysing the organisation's ALM solution.

This four-element framework was found to be too rough. The study showed that the framework needed to be elaborated, especially with regard to the 'communication' element of ALM. First, communication was treated in an initial framework as part of the 'creation and management of project artefacts' element, as common databases provide means for sharing product- and project-related information. The role of communication in ALM is broader however. It covers communication channels for exchanging product and project data, but also other informal communication channels, such as chat, discussion forums, etc. Both aspects, synchronous and asynchronous communication, which are especially relevant to support geographically distributed development, needed to be handled. This was detected in a case study. Process automation and tool integration were also handled in the same element, as ALM should support the procedures of the project in an integrated development environment to support the whole development lifecycle. Even though the case data also supported the view that an integrated toolset with process support is important, these elements needed to be separated in the ALM framework for documentation purposes. As the ALM framework is used to document and analyse practical ALM solutions, it is reasonable to form it so that it provides clear structure for the documentation of the ALM solution. The ALM framework element named ‘process automation’ was changed to ‘process support’, as that is also used in SCM literature (e.g., Estublier, 2000) and it better describes the element. Furthermore, the terminology of the ALM elements was made more consistent by replacing the name ‘creation and management of project artefacts’ with ‘creation and management of lifecycle artefacts’. These refinements finally resulted in the second version of the ALM framework, which is introduced in more detail in Paper II.

The author is the main writer of and contributor to Paper I. He carried out literature studies, ALM framework construction, planning of questionnaire and interview questions, conducted interviews, analysed case data, and made refinements to the initial ALM framework. Mr. Antti Välimäki (co-author) contributed to Paper I by helping to arrange the case and field visit, reviewing the questionnaire and interview questions, and helped that the necessary information were gained from case organisation 1. As a company insider, he provided his contribution to the analysis of the case data. He was especially interested in the distributed development and analysed data as input to his research relating to the ‘GSD patterns for project management’.

Key results of the paper:

- First version of the ALM Framework based on the literature and vendor’s web pages and its application in an industrial case study for documenting and analysing a case study company’s ALM solution.
- Refinements to the initial ALM framework based on case results (resulted in the second version of the ALM framework introduced in Paper II).

5.2 PAPER II: Get a Grip on your Distributed Software Development with Application Lifecycle Management

Paper II continues the work started in Paper I. It presents the second ALM case study of the longitudinal case study in case organisation 1. The paper focuses on the key issue of the thesis – constructing and demonstrating the Application Lifecycle Management framework. The paper describes the *second version of the ALM framework* (Figure 11) and presents the second case study in case organisation 1 in which the ALM framework was applied.

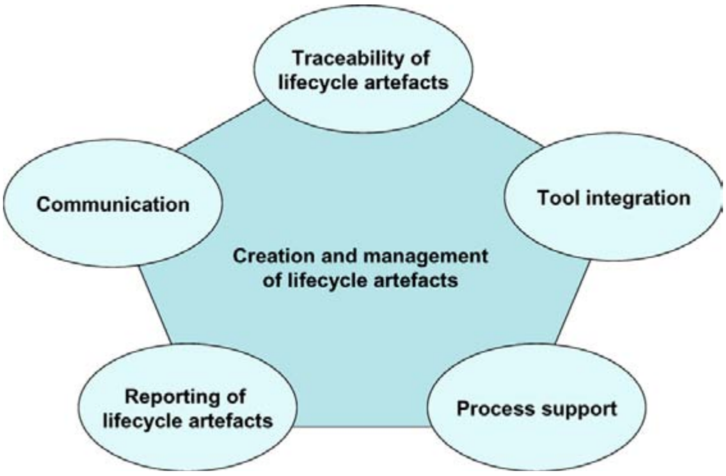

Traceability of lifecycle artefacts
Communication
Tool integration
Creation and management of lifecycle artefacts
Reporting of lifecycle artefacts
Process support

Figure 11. Second version of the ALM framework.

- Creation and management of lifecycle artefacts: This element is the foundation for the whole ALM framework. It is therefore located in the centre of Figure 11. This element is used to analyse which data items are created, stored, identified, versioned and managed during various phases of the development lifecycle.
- Traceability of lifecycle artefacts: This element is used to analyse the traceability of lifecycle artefacts.
- Reporting of lifecycle artefacts: This element is used to analyse the way the solution supports reporting in the development lifecycle.
- Communication: This element is used to analyse what kinds of means or practices there are for communication in an ALM solution.
- Process support: This element is used to analyse how the whole development lifecycle is supported by the ALM solution.
- Tool integration: This element is used to analyse what kinds of tool integrations exist in the development environment.

The same SW teams were the subject of the study as in the first case study. Case data were collected by interviewing project managers. The case data comprised data from the previous case study round (Paper I) complemented by the project manager's second interview round. All the data were documented and analysed using the second version of the ALM framework. Furthermore, the paper presented the analysis of the organisation’s ALM tool, Microsoft’s Team Foundation Server. The analysis was carried out at VTT.

The case study shows that the ALM framework can be used to document and analyse the organisation’s ALM solution. The relations between the ALM elements were still unclear however. The ALM framework element ‘creation and management of lifecycle artefacts’ forms the foundation of the whole ALM framework in which traceability, reporting and communication use the element, but the roles of subsequent relations remained unclear.

The case study also resulted in an analysis based on the 3C model of how ALM could support the management of distributed SW projects. An interesting finding from the study presented in Paper II was that two SW teams starting with similar ALM solutions deployed some team-specific adaptations to their ALM solutions. The ALM solutions of the teams therefore became different – the result of their different operational environments.

The author is the main writer of and contributor to Paper II. He conducted a literature study, the ALM framework description, carried out a second interview round, documented and analysed case data, and analysed the ALM framework against the 3C model. The analysis of TFS was performed in cooperation with Mr. Juha Vitikka (Vitikka, 2009). Mr. Antti Välimäki (co-author) contributed to Paper II by helping to arrange the second interview round, reviewing interview questions and helping to ensure that the necessary information was obtained from case organisation 1. He provided his contribution to the analysis and understanding of the case data and was especially interested from a distributed development point of view and analysed data as input to his research related to the ‘GSD patterns for project management’. Mr. Välimäki also participated in the analysis of ALM elements against the 3C model.

Key results of the paper:

- Detailed description of the <u>second version of the ALM Framework</u> and its application in an industrial case study for documenting and analysing a case study company’s ALM solution for a GSD environment.
- Description of experiences (solution, +, -, distribution) according to the ALM framework elements.
- Mapping how the ALM framework elements support the elements of the 3C model. ALM supports GSD from a 3C model point of view.

5.3 PAPER III: Applying Application Lifecycle Management for the Development of Complex Systems: Experiences from the Automation Industry

Paper III continues the work started in Papers I and II and forms a third case study of the longitudinal ALM case study in case organisation 1. The paper focuses on the key issue of the thesis – constructing and demonstrating the Application Lifecycle Management framework. This case study analysed the ALM solutions of the two SW teams. The SW teams were the same as in the two previous case studies (Paper I and II). The motivation for this case study was the prior knowledge that both SW teams successfully implemented improvements related to the requirements management features of their ALM solutions and that these solutions were different. It was therefore interesting to find out why the teams ended up with different solutions and check if it is possible to describe these fairly different solutions using the ALM framework.

The key results of the paper were the application of the second version of the ALM Framework in the industrial case study for documenting and analysing the case study company's ALM solutions in two SW teams. The case study also analysed the history of ALM solution development in the case study company and revealed that the development of the ALM solutions in a case study company in two SW teams was iterative starting from quite similar solutions and ending up as fairly different solutions based on their different needs. SW team 1 started with the solution that contained TFS in the central role, even though there were also some Notes databases. After a few years, the team ended up with integrated Notes databases, and TFS was only used for source code control. SW team 2 started with a TFS-centric solution with some Notes databases. They ended up with a solution in which the TFS had an even stronger role. SW team 1 therefore ended up with a Notes-dominant solution and SW team 2 with a TFS-dominant solution even though the purpose of SW systems they produce are the same (reporting SW). SW Team 1 produces SW that is part of an evolving platform product, however, whereas SW Team 2 produces industry-specific SW products. SW team 1 therefore needed to adapt to the whole development infrastructure of the organisation. It was also notable for both teams that in the beginning it was beneficial to have a ready ALM tool (TFS with a SCRUM template) to support the deployment of a new development method and after successful deployment, the teams were able to start adapting ALM solution better to fit their special needs.

The case study resulted in an elaborated version of the ALM Framework including relations between ALM framework elements (*third version of the ALM Framework,* Figure 12). The framework relations were defined by comparing the practical implementations of ALM in a case study company with ALM elements in the framework. Based on the comparison, the four ALM elements in the middle form the levels of the ALM elements (hierarchy) with the upper element using artefacts provided by the lower level elements. The role of process support and tool integration, however, is to provide an efficient working environment by equipping the elements presented in the middle to support an overall lifecycle process and tool integration.

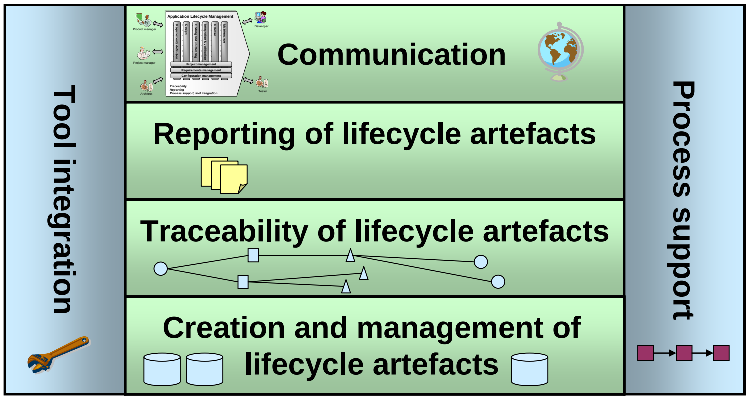

Tool integration
Communication
Reporting of lifecycle artefacts
Traceability of lifecycle artefacts
Creation and management of lifecycle artefacts
Process support

Figure 12. Third version of the ALM framework.

- ‘Creation and management of lifecycle artefacts’ is the foundation of ALM. The product information collected and managed by this element is needed for, for instance, traceability and reporting activities.
- ‘Traceability of lifecycle artefacts’ provides a means to identify and maintain relationships between managed lifecycle artefacts and, therefore, facilitates reporting, change impact analysis and information visibility through the development lifecycle.
- ‘Reporting of lifecycle artefacts’ utilises managed lifecycle artefacts and traceability information to generate needed reports from the lifecycle product information to support SW development and management.
- ‘Communication’ provides communication tools (e.g., chat) as well as channels for distributing information on product lifecycle artefacts, links

and reports and thus facilitates product information visibility for the whole SW project.

- <u>‘Process support’</u> and <u>‘Tool integration’</u> are the elements used to configure the ALM solution to support SW development procedures and facilitate a productive development environment by allowing the user to launch tools and transfer information easily between different tools and databases.

In this case, the case study shows that the ALM framework can be used to document and analyse different kinds of ALM solutions. ALM framework facilitated the documentation, understanding and analysis of ALM solutions during the iterative improvement effort in case organisation 1. On the other hand, close long-term cooperation with case organisation 1 allowed the author to test the evolving ALM framework in a complex real-life situation and deepen the practical understanding of the concept of ALM. This kind of long-term cooperation is very important to research scientists to ground their research and provide improvement ideas for the company.

The author is the main writer of and contributor to Paper III. He carried out an additional literature study, analysed and documented case data, and refined the ALM framework to cover also the relations of the ALM elements (ALM framework version three). Mr. Antti Välimäki (co-author) contributed to Paper III by carrying out a complementary interview round in case organisation 1. He also provided his contribution to the analysis and understanding of the case data and was especially interested from a distributed development point of view and analysed data as input to his research related to the ‘GSD patterns for project management’.

Key results of the paper:

- Application of the second version of the ALM framework for documenting and analysing the case study company’s ALM solutions.

- Description of the history of ALM development in case organisation 1. How and why ALM solutions evolved in the context of case organisation 1.

- Experiences from the history of ALM improvement and deployment in case organisation 1 from 2006 to 2009.

- Description of the third version of the ALM Framework. The ALM framework was complemented with the relations between the framework elements.

5.4 PAPER IV: Global Software Development Patterns for Project Management (co-author)

The paper focuses on one of the issues of the thesis – global software development. The paper presents GSD patterns and their validation using a scenario-based method. The main author, writer and contributor of the paper is Mr. Antti Välimäki. Mr. Välimäki introduces GSD pattern language and its validation using a scenario-based validation method. The third author of the paper is Professor Kai Koskimies who has worked as a mentor on this paper.

In Paper IV, the author of this thesis was a co-author (second author) and focused on the analysis of the results of GSD pattern validation from an ALM point of view. The key finding of this paper from the thesis point of view was the evidence on the applicability of ALM for the management of distributed SW projects. The results obtained from the evaluation of the GSD patterns indicate suitable practices for global software development. Earlier results in Papers I, II and III indicated that ALM supported the operation in a global development environment (e.g., analysis of ALM elements against the 3C-model in Paper II). The results of the Q-PAM analysis presented in Paper IV strengthen this claim. From all the GSD process patterns presented in this paper, 'Communication Tools' and 'Common Repositories and Tools' are particularly related to ALM. Analysis results indicate that these ALM-related GSD patterns support the selected GSD scenarios.

The paper provided more evidence that ALM is an important enabler of GSD. This evidence was utilised to further analyse the ALM elements and GSD patterns in Paper VI and applying them to the ToolChain analysis.

Key results of the paper:

- Preliminary analysis of how GSD patterns relate to ALM elements
- More evidence that ALM is an important concept to support GSD. Strengthen the results presented in Papers I, II and III.

5.5 PAPER V: Challenges in Collaboration: Tool Chain Enables Transparency Beyond Partner Borders (co-author)

Paper V presents the background to ToolChain and the first study on the concept of ALM. The paper presents the way requirements tracing and global collaboration can be supported during the software development lifecycle. The key results of this paper from a thesis point of view relate to the ALM concept and ToolChain.

The author of this thesis is the co-author of paper V (second author) and initiated the writing of the paper. In this paper, the author was responsible for carrying out the first study of the concept of ALM and also contributed to the paper from a requirements traceability point of view. Furthermore, the author participated in the planning and realisation of the research process. The paper defined the first traceability model for the ToolChain implementation based on the enquiry and workshop for the industrial and academic partners of the research project. Mr. Samuli Heinonen is the main author and writer of the paper and was responsible for the ToolChain implementation. Mr. Juha Takalo is co-author of the paper and was responsible for the tool integration research.

At this point, tool integration was seen as a vital element to support the management of lifecycle artefacts. Furthermore, traceability and information visibility were highlighted to support collaboration in a global development environment. The paper included our first study towards a better understanding of the ALM concept. After this paper, ALM and tool integration research were carried out in cooperation in two separate research tracks over three years. Paper VI combines these two research tracks and proposes modifications to the ToolChain implementation based on ALM research and the related GSD pattern research.

Key results of the paper:

- First literature study on ALM and related concepts (traceability, tool integration)
- First implementation of ToolChain, which is a tool-independent platform for integrating the different tools needed during the development lifecycle.

5.6 PAPER VI: Extending Tool Integration Environment Towards Application Lifecycle Management

In Paper VI, the current version of the ToolChain tool integration environment was studied to find out how well it supports lifecycle management and the global development environment. The paper focuses on the key issue of the thesis – constructing and demonstrating the Application Lifecycle Management framework. At this point, the ALM framework was already successfully applied in ALM cases in case organisation 1. It was therefore interesting to apply it to analyse a technically different kind of solution, the Eclipse-based solution, in the context of another organisation – case organisation 2. Furthermore, the case allowed the results of the GSD pattern and ALM framework mapping (Table 4) to be applied to analyse the GSD support of ToolChain.

Table 4. GSD pattern and ALM framework mapping.

ALM elements	**Related GSD patterns**	**How GSD patterns cover ALM elements**
Creation and management of lifecycle artefacts	Common Repositories and Tools (ID 07)	Global databases to support the management and visibility of lifecycle artefacts
Traceability of lifecycle artefacts	Common Repositories and Tools (ID 07)	Traceability of lifecycle artefacts in a GSD environment
Reporting of lifecycle artefacts	Common Repositories and Tools (ID 07)	Reporting of lifecycle artefacts and traces in a GSD environment
Communication	Common Repositories and Tools (ID 07)	Asynchronous communication (visibility of lifecycle artefacts)
	Communication Tools (ID 06)	Synchronous/asynchronous communication tools (e.g., net meeting, chat, conference phone, discussion forum)
Process support	Common Repositories and Tools (ID 07)	Process support features such as state models, workflows and process templates
	Use Common Processes (ID 12)	Common upper level GSD process and the ability to tailor the process support for a project or team at site level
Tool integration	Common Repositories and Tools (ID 07)	In practice, a common repository can be a single central database or several integrated databases.

The author of this thesis is the main author and writer of the paper. The author analysed ToolChain and the ToolChain demonstration using ALM framework version three. The author was responsible for planning and monitoring the case, data collection and analysis of the case data from an ALM framework point of view. The author, in cooperation with Mr. Antti Välimäki, analysed how the GSD patterns map to the ALM framework elements. An ALM analysis complemented by a GSD pattern analysis specified how ToolChain should be improved towards better lifecycle management in a GSD environment. Mr. Antti Välimäki, in cooperation with the author, analysed the ToolChain against ALM-related GSD patterns and wrote the analysis results in the paper.

Mr. Juho Eskeli has been responsible for the implementation of the latest version of ToolChain. He has also contributed to the description of the ToolChain and participated in the case data collection and analysis. Mrs. Susanna Teppola contributed to the paper from a workflows point of view. She has carried out a detailed literature study on workflow as a concept and on workflow-related techniques. A subset of this study is included in this paper. Mr. Pekka Tuuttila contributed to the paper from a demonstration point of view. He was responsible for all the practical arrangements for the case study in case organisation 2 and contributed to the paper from the description of demonstration point of view. Mr. Markus Piippola contributed to the paper from a system under analysis (JiVe platform) point of view. He introduced the JiVe platform. All the authors participated in the workshop, where questionnaire results were presented, discussed and refined.

The study demonstrated the relations of the ALM framework elements, as it showed that ToolChain provided basic ALM features, though more advanced features based on the basic features were missing. It was promising that most of these advanced features could be extended into ToolChain, as the basic features needed to implement the advanced features are in place in the current version of the ToolChain. The functionality and toolset of the ToolChain prototype have evolved gradually. More advanced features that would provide additional value for stakeholders operating in a global development environment are missing however. The results of the demonstration and analysis indicate that further extensions are needed, especially relating to test automation, lifecycle reporting, synchronous communication and workflow support.

Key results of the paper:

- Application of ALM framework version three to the analysis of ToolChain and the ToolChain demonstration
- Mapping between GSD patterns and ALM framework elements. ToolChain analysis against the ALM framework-related GSD patterns
- Proposals for ToolChain improvements based on the ALM framework and GSD/ALM analysis.

5.7 Summary

Figure 13 summarises how the research work has proceeded as a series of case studies constructing and using the versions of the ALM framework to document and analyse the ALM solutions of case organisation 1 and the ToolChain implementation in case organisation 2. Arrows point to the direction of reading in the figure.

Case studies have analysed practical ALM solution versions by collecting experiences from the real users of the ALM solutions and analysing the ALM solutions themselves as used in case organisations 1 and 2 using the ALM framework versions. Case results have been provided for case organisations. The case studies have contributed iteratively to the versions of the ALM framework. Finally, this research has resulted in ALM Framework version three. Furthermore, the GSD patterns developed by Mr. Antti Välimäki (presented in Paper IV) have been analysed against the ALM framework elements that created the ALM/GSD mapping that shows how the ALM framework elements can support GSD patterns. This mapping was then used to analyse the implementation of ToolChain to analyse its support for GSD. ALM Framework version three, complemented by the ALM/GSD mapping, forms the proposed ALM framework presented in Chapter 4 of this thesis.

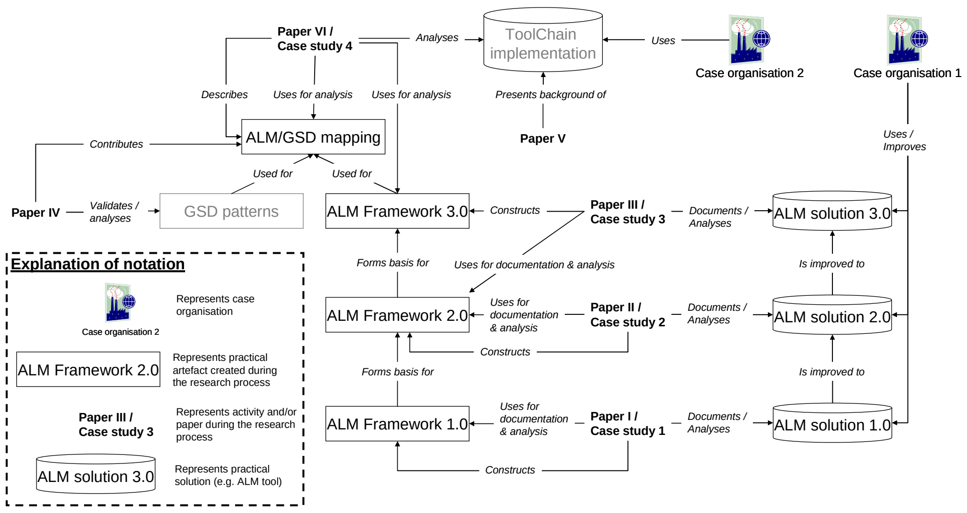


Figure 13. Research as sequential case studies contributing to the construction and demonstration of the ALM framework versions.

6. Discussion

This section discusses the results of the thesis. First, Section 6.1 compares the ALM framework with the other related ALM results from the current literature. Second, Section 6.2 discusses the experiences gained from the case studies. Third, Section 6.3 discusses the implications for research and practice, and fourth, Section 6.4 evaluates the construct validity, external validity and reliability of this research.

6.1 Comparing the ALM framework with related research

Chapter 2 of this thesis provides a background to ALM-related scientific and professional literature. In all the work that has been carried out, there are surprisingly few empirical and research reports in which the concept of ALM has been studied. During the literature studies, the author found a few contributions related to the topic of the research:

- ALM framework from Microsoft (Rossberg, 2008): service offering from Microsoft
- Solution concepts for the integration of product engineering and lifecycle management (Weiß et al., 2009; Pirklbauer et al., 2009): scientific article
- Three pillars of ALM (Schwaber, 2006): professional article
- ALME Classification Framework (Kravchik, 2009): Master of Science thesis
- Software Engineering Environment (SEE) Services (ISO/IEC 15940, 2006): international standard.

ALM framework from Microsoft:

Microsoft provides an Application Lifecycle Management Framework that is a set of service offerings that provides a foundation for development solutions (Microsoft, 2010). An interesting service from the point of view of this thesis is the 'Assessment' service, which provides an insight into the current solution and provides improvement proposals. The solution seems very comprehensive. The framework is provided by the ALM vendor, however, and it is therefore difficult to consider it a neutral approach. Furthermore, the proposed framework presented in this thesis is aimed at the documentation of the organisation's ALM solution – the overview of the ALM solution in an organisation as discussed in Section 4.2 of this thesis. This thesis also aims to motivate further scientific discussion on ALM as a concept.

Solution concepts:

In the chapter of the Hagenberg Research book published by Springer, Weiß et al. (2009) present their ongoing research related to the integration of product engineering and lifecycle management. This research is also published as a short paper by Pirklbauer et al. (2009). In their research, they have detected that the concept of ALM is vague. This has also been detected in the research presented in this thesis. Weiß et al. (2009) have studied the problems related to the integration of product engineering and lifecycle management. Furthermore, they have defined a conceptual model for product engineering and lifecycle management integration. According to Pirklbauer et al. (2009), this model has been used as guidance to identify and prioritise problem areas and to plan for a tailored ALM solution. The purpose of the model is similar to that of the ALM framework presented in this thesis. In the conceptual model, the *solution concepts* seem to be similar entities to the elements of the ALM framework presented in this thesis. When comparing the *solution concepts* presented by Pirklbauer et al. (2009) with the *ALM elements* presented in this thesis, the following mapping can be made (Table 5).

Table 5. Comparing ALM elements and solution concepts.

ALM elements presented in this thesis	**Solution concepts** (Pirklbauer et al., 2009)
Creation and management of lifecycle artefacts	Version control
Traceability of lifecycle artefacts	Traceability
Reporting of lifecycle artefacts	Measurement
Communication	Collaboration support
Process support	Workflow support
Tool integration	-
-	Shared services

This mapping shows that there are several similarities between these elements and concepts, even though the extent of the element or concept may differ. For instance, the ‘Creation and management of lifecycle artefacts’ element is wider than the pure ‘Version control’ concept. ‘Shared services’ in solution concepts are interesting from an ALM point of view, as they would bring the management of users and access rights to the ALM framework. The importance of centralised access rights management in integrated development environments has also been detected in Pesola et al. (2008).

Three pillars of ALM:

Schwaber (2006) presents three pillars of ALM in a professional article published several years ago. This work has been referred in many articles relating to ALM (e.g., Rossberg, 2008, and Kravchik, 2009). The pillars of ALM are traceability, process automation and reporting (‘reporting’ is referred to as ‘visibility’ in Rossberg, 2008). These three pillars of ALM are also one of the bases of the ALM framework presented in this thesis.

ALME Classification Framework:

The ALME Classification Framework (Kravchik, 2009) is intended as a comparative study of a number of existing ALM environments. The aim of this framework is to provide a tool for assessing ALM environments, understanding their characteristics, and discussing and comparing them. When comparing the ALM framework presented in this thesis and the ALME classification framework, it can be noted that they are intended for different purposes. The ALME

Classification Framework is intended for assessing ALM environments (i.e., tools), whereas the ALM framework is intended for documenting and analysing an organisation's ALM solution containing tools and practices. The ALM framework has a stronger focus on the documentation of the solution: how to document the ALM solution of the organisation. Kravchik (2009) provides a valuable insight into current ALM tools (capabilities of current ALM tools) on the markets that may be used in conjunction with the ALM framework when documenting and analysing the ALM solutions of an organisation.

<u>Software Engineering Environment (SEE) Services (ISO/IEC 15940)</u>

The Software Engineering Environment (SEE) Services (ISO/IEC 15940, 2006) standard describes services that support all of the software lifecycle processes defined in ISO/IEC 12207. The application of the standard is presented in Annex C of the standard (ISO/IEC 15940, 2006). The standard defines that it:

'can provide benefits to software engineers who use tools, those involved in process improvements and who acquire tools, suppliers who provide tools, and educators and software engineering consultats who provide advise on tools and SEEs.'

It seems that the elements of the ALM framework can be related to the services of the SEE standard. ALM elements cover a subset of the standard. 'Software engineering services' and 'SEE Infrastructure services' are missing from the ALM framework. The purposes of the standard and the ALM framework are slightly different however. Besides the analysis, the ALM framework is intended to support the documentation and, therefore, considers, based on the experiences from the case studies, what should be documented and how. This is not presented in the ISO/IEC 15940 standard. The use of the ISO/IEC 15940 standard in conjunction with the ALM framework when documenting an organisation's ALM solution would be beneficial. A more detailed mapping of how the ALM framework and SEE services relate to each other would definitely be worth researching in the future.

6.2 Discussions on case experiences

This section discusses the experiences gained from the cases. Each experience is discussed with references to the related scientific or professional articles.

Development of complex products:

In the ALM cases, the interfaces to system-level product information management tools affected the SW projects' ALM solutions (company/organisation constraints for SW projects). In SW team 1, this led to the use of several databases, as the inter-project product information management practices and tools needed to be collectively agreed and compatible between development projects. This creates challenges for different tools and databases to interoperate to keep data consistent and traceable. The organisation's ALM solution is therefore part of the overall PLM solution, possibly covering a number of product information management systems that need to interface and be compatible with each other. The author also detected the same issue in the telecommunications industry from a configuration management point of view (Kääriäinen et al., 2004). The study showed that without the inter-project approach, the CM may be well realised inside one project, but the whole system's CM would be inadequate. The need for unified management of complex products has also been studied in Crnkovic et al. (2003) and Svensson (2003).

This research also shows that, in practice, the ALM solution can comprise a central database or a collection of databases. In the case of several databases, the interoperability of the databases is essential to maintain consistency of product information (e.g., tight integration or loosely coupled integration with a proper process, Krikhaar et al., 2009).

ALM support for global software development:

A central TFS database allowed SW team 2 to have a consistent view of the project data in a global development environment. SW team 1 used a collection of Notes databases however. Moore et al. (2007) have reported the experiences of using TFS to manage global software development. Doyle and Lloyd (2007) have also stressed the use of the ALM solution with a central repository to support work in a global development environment. On the other hand, Portillo-Rodríguez et al. (2010) present Notes as one tool to support GSD from a document management point of view. They conclude that the tool’s ability to allow working when the server is offline and updating information when the server is online is a desirable feature. All the databases for both teams (TFS and Notes) are accessible globally, an essential prerequisite for the databases when case organisation 1 started to improve the solutions for global SW development.

In the GSD-ALM mapping in the research presented in this thesis, the importance of ALM to GSD was detected. Even though the use of the ALM tool and common upper level GSD processes are important in GSD, the GSD patterns suggest that site-level tailoring should be allowed if it does not cause problems with upper level processes (Paper IV).

Integration of different technologies:

This research shows that the integration of tools from different vendors, which may be implemented using different technologies, is still difficult. In the ALM cases, for example, the integration of TFS and Notes was reported to be difficult and it is therefore easier to focus on a particular technology and build a solution around it. The author wants to bring out this experience, as tool integration has been under active research for a long time (see, for example, the comprehensive literature study and the research agenda for tool integration presented by Wicks and Dewar, 2007). Wicks and Dewar (2007) report that tool integration has been studied, especially from a technological viewpoint. They suggest a more business-oriented approach to future tool integration research however.

Another interesting aspect of tool integration relates to integration platforms. Eclipse (Eclipse, 2010), for example, provides mechanisms to integrate various tools into one platform. This kind of tool integration approach does not lock in an organisation with one vendor (Portillo-Rodríguez et al., 2010). The Eclipse open-source community started the project with the aim of providing a logical definition of the overall interoperability business process known as the Application Lifecycle Framework (ALF). The project failed to gain the support of significant vendors, however, and it was terminated in 2008 (Kravchik, 2009). Similarly, from a PLM point of view, Abramovici (2007) states that despite intensive standardisation activities, general accepted industry standards for PLM meta-data models and PLM processes are still missing.

Adapting ALM based on the needs of the SW team:

The ALM cases show that different SW teams have different needs for the ALM solution. In the ALM cases, the two SW teams were almost identical, only the internal purpose of the SW subsystem they produced was different. Both teams produced reporting SW. SW Team 1 produced a subsystem that is part of an evolving platform product, however, whereas SW Team 2 produced subsystems that were part of industry-specific products. The study showed that SW team 1 needed a common way to share and access the same type of information with the other platform projects. SW Team 2, on the other hand, had a single, central ALM solution that worked well as it did not have strong relations to the platform level.

The adaptation of the solution based on the needs of the development project has been reported from, for instance, a configuration management point of view by Leon (2000), Buckley (1996), Whitgift (1991) and IEEE Std-828 (2005), and from the requirements management point of view by Sommerville and Sawyer (1997). In the ALM cases, a commercial ALM tool with a process template that could be used to configure the whole system to support the selected development method facilitated the deployment of a new development method (Scrum). In the beginning, it was beneficial to have a ready ALM solution (TFS with a SCRUM template) to support the deployment of a new development method, and after successful deployment the teams could start to adapt the solution better to fit their special needs. This approach was different from the experiences of Moore et al. (2007). Moore et al. (2007) report that they tailored the TFS process template considerably for their purposes right from the start. In the ALM cases, however, it was feasible to start from a ready ALM solution and, after successful deployment, start to tailor it.

With regard to process template tailoring, Medina-Domínguez et al. (2007) present interesting research. They propose a project pattern concept and a model to support process improvement based on patterns in the TFS environment. This would ease the selection of appropriate practices based on patterns for SW projects that use TFS for the management of development projects.

6.3 Implications for research and practice

The purpose of this research is to understand better the concept of ALM and to support the improvement of ALM solutions in industry. The research showed that there is little scientific discussion about the concept of ALM. Tool vendors use the term ALM widely, however, to indicate their integrated tools for SW development and management. The research showed that there are some concepts that relate to ALM, such as CDE (Collaborative Development Environment) and SEE (Software Engineering Environment). The purpose of the ALM framework and the improvement models also overlaps, i.e., the description and analysis of the current state. In the future, the research should therefore study the relation between the ALM framework and related concepts, standards and improvement models.

The framework can be useful to ALM researchers and practitioners. For researchers, the framework comprises principal elements of ALM collected and demonstrated using industrial case studies. As ALM is not well defined (Weiß et al., 2009, and Göthe et al., 2008), this framework is a starting point, an essential step, towards a more elaborate framework and understanding of the concept of ALM. For ALM practitioners, the framework provides tried and tested means to document and analyse an organisation’s ALM solution. In this research, the industrial domains in the case studies were automation and telecommunication. The proposed ALM framework proved adequate to document and analyse different kinds of ALM solutions (a central ALM database or a collection of databases). For practitioners, the concept of ALM is usually realised through the vendors of ALM tools. These definitions are driven by existing or planned marketing strategies by tool vendors, however, and therefore, up to now, a broad variety of tools has been labelled as ALM tools (Pirklbauer et al., 2009). The research presented in this thesis has contributed to the scientific discussion on what constitutes ALM. This thesis also reports the experiences from gradual ALM improvement. These experiences are potentially in the interest of ALM practitioners seeking ALM solutions for projects that operate within a similar context to those presented in this research. This study therefore also provides a practical viewpoint of ALM so that ALM practitioners can compare the operational environment of their projects with the case studies presented in this thesis and apply the findings of this study to their daily work when appropriate. The contribution of this research therefore brings researchers and practitioners closer to each other and motivates further discussion on ALM as a concept, and industrial-based challenges and practical ALM experiences in scientific forums.

6.4 Evaluation of research validity and reliability

This thesis presents interpretive research that contains a set of case studies in which the use of an evolving ALM framework is demonstrated in an industrial context. The research can be evaluated through validity (construct, internal, external) and reliability (Yin, 2003). From the interpretive research point of view, the most appropriate, according to Holmström Olsson et al. (2008), are *construct validity*, *external validity* and *reliability*. Each of these will be discussed from the point of view of the research of this thesis in the next subsections.

6.4.1 Construct validity

'Construct validity has to do with the extent to which the constructs as operationalised relate to the phenomenon the research purports to address' (Holmström Olsson et al., 2008). Yin (2003) proposes three tactics to increase construct validity for case studies: multiple sources of evidence, establishing a chain of evidence and having key informants review draft case study reports. 'Multiple sources of evidence' are covered in the research by using data triangulation. Case data have been collected from several sources. During the first ALM case study (presented in Paper I), for instance, the data were collected using a questionnaire and interviews. The case data were also complemented by tool demonstrations while the author visited case organisation 1 and from the discussions with the development manager of the organisation. 'Establish chain of evidence' is covered in the research by collecting case data (questionnaire responses, interview notes, discussions, etc.) and making notes on data that indicate the relation to the results (ALM elements and ALM experiences). Many claims that have been stressed by the project managers or development manager of case organisation 1 have been taken 'as such' into the experiences. Furthermore, Runeson and Höst (2009) state that if questions are not interpreted in the same way by the researcher and the interviewee, there is a threat to the construct validity. The motivation letter and questions for the questionnaire/interviews in the ALM cases were reviewed by a company representative so that all the necessary concepts were explained and the terminology was suitable for the case. 'Have key informants review the draft case study report' is covered in this research by asking key informants (project managers and the development manager in the ALM cases, and contributors in the ToolChain case) to review and comment on the results and conclusions of the studies.

6.4.2 External validity

External validity concerns the extent to which the research results apply in other real-world settings (Holmström Olsson et al., 2008). According to Runeson and Höst (2009), this is concerned with the extent to which it is possible to generalise the findings. As already discussed in Section 3.2, the generalisation of results is a problem in case studies. The nature of the case study method limits the generalisation of results. According to Yin (2003), a case study 'investigates a contemporary phenomenon within its real-life context'. Yin (2003) continues to say that 'case studies, like experiments, are generalisable to theoretical propositions and not to populations or universes'. This research has produced a theoretical proposition about the ALM framework. The evolving framework has been used practically in the series of case studies. There is therefore evidence of its applicability in more than one case. This proposed framework then needs to be applied to future cases in order to gain evidence on its suitability in other contexts.

6.4.3 Reliability

Repeatability of research is at the heart of reliability (Holmström Olsson et al., 2008). The goal of reliability is to minimise the errors and biases in a study (Yin, 2003). Runeson and Höst (2009) present a threat to reliability if, for example, the coding of case data is unclear. In the ALM cases, the author used primarily the ALM framework elements to code the case data. In the second case study, the case data were also coded with their relation to distributed development.

The whole research process including planning, data collection, analysis and reporting has been documented in Chapter 3. In interpretive research, the researcher interprets other people's interpretations (Walsham, 1995) and there can therefore be subjective interpretations from the case data. It may therefore sometimes be difficult to follow the chain from case data to conclusions. Yin (2003) advises that it would be good practice to carry out (and document) research, as if someone is always looking over your shoulder. In this research, the chain from informants to results is not complex. In particular, many experiences reported by informants have been written into the publications as such, and therefore indicate the interpretations of the real users of the ALM solution in their context (i.e., what they responded to). To make this chain (even it is short) fully traceable requires adequate means for documenting the interpretations. This issue is a subject of critique in the research process presented in this thesis. In this research, interview notes were written, but the interviews were not tape-recorded and transcribed. There are several benefits of tape-recording interviews (Walsham, 2006). The disadvantages of tape-recording, however, include the inhibition of interviewees in the case of sensitive material and the time spent transcribing the interview recordings (Walsham, 1995). Walsham (2006) also argues that time spent on transcriptions could be used for, for instance, a more detailed analysis of the case data. At the beginning of this research, it was agreed with the representative of case organisation 1 that interviews would not be recorded and transcribed even though the transcription task could be provided to the subcontractor (e.g., a translation office). To maintain trust, however, this would also have required a separate non-disclosure agreement (Runeson and Höst, 2009) with the subcontractor.

7. Conclusions

This thesis proposes a framework for documenting and analysing the organisation's application lifecycle management solution. This thesis studies the concept of ALM based on literature and four industrial case studies. The main contributions of the thesis are the proposed ALM framework and its demonstration in industrial cases relating to complex systems development. Furthermore, the thesis reports on the industrial experiences of ALM solutions in an automation and telecommunications industrial context.

7.1 Answers to research questions

The principal research question of the thesis was as follows:

RQ1: How can the documentation and analysis of the ALM solution be facilitated in an organisation operating in a Global Software Development environment?

This research presents a proposal for the ALM framework that can be used to facilitate the improvement of ALM in an organisation that operates in a global development environment. The ALM framework intends to answer the main research question by providing a means to document and analyse the organisation's ALM solution in order to understand it and find possible improvement targets. In order to support the documentation and analysis, there is a need to understand what constitutes ALM. The ALM framework contains the description of the ALM elements, element relations and mapping with global software development patterns. This framework is described in Chapter 4 of this thesis. Furthermore, case studies reported in Papers I, II, III and VI present industrial experiences of the deployment and improvement of practical ALM solutions in industrial contexts. The principal research question was divided into the following sub-questions, each of which focuses on a specific part of the main research question:

RQ1.1: What are the main elements of Application Lifecycle Management?

The elements of ALM were defined by iteratively constructing and demonstrating the ALM framework based on literature and case studies presented in Papers I, II, III, V and VI. The current version of the ALM framework contains six elements, as presented in Section 4.2.1 of this thesis. These elements form the basis of the framework.

RQ1.2: What are the relations between the elements of Application Lifecycle Management?

The relations between the elements of ALM were described in Paper III. The relations were defined by comparing the practical implementations of the ALM in case organisation 1 with the ALM elements in the framework. The ALM framework containing the description of the elements and relations has been applied to a case study presented in Paper VI. The thesis summarises the relations between the ALM elements in Section 4.2.2.

RQ1.3: How can Application Lifecycle Management be applied to a Global Software Development context?

In this research, global software development has been a factor relating to the case context. The different phases of the research have therefore accumulated knowledge on the applicability of ALM to global software development (for example, Paper I and Paper II report how ALM solution supported GSD in ALM cases). To make the study more detailed, the research applied the proposed GSD patterns and reflected on the ALM elements against the patterns to reveal junctions between the concepts. This work has been described in Papers IV and VI, and summarised in Section 4.2.3.

7.2 Limitations and future research

This thesis reports on an effort towards the ALM framework that can be used for documenting and analysing an organisation's ALM solution. The framework is a starting point – a proposal towards a more extensive framework. This study attempts to provide a practical viewpoint for ALM improvement so that practitioners can compare the operational environment of their organisations and teams with our case studies and apply the results and experiences of this research to their ALM improvement work when appropriate.

The research presented in this thesis has been oriented in a horizontal way, i.e., by analysing ALM as a whole, instead of with a vertical orientation, i.e., analysing each ALM element in depth. ALM as a concept, however, is wide and each ALM element presented in the proposed ALM framework is worth its own research effort, for instance, traceability, tool integration or process support. The vertical in-depth research of each ALM element was therefore set outside the scope of this research. The vertical study of elements is a potential topic for future research, however, in order to construct a more elaborate version of the framework.

The development cycle of the product contains activities related to the definition and realisation of the product – from business needs to delivery. This research has limited the scope to the development cycle of the product even though the full lifecycle is broader, and it should therefore be the subject of future research. This means that research should be extended towards other lifecycle phases to cover also operation and maintenance phase of the lifecycle.

The nature of the case study method limits the generalisation of the results. According to Yin (2003), 'case studies, like experiments, are generalisable to theoretical propositions and not to populations or universes'. This research has generated a proposal for an ALM framework. The framework has been constructed and demonstrated iteratively in four cases in two different organisations. The framework has been used primarily in the context of the automation industry and in a fourth case study in the context of the telecommunications industry. Both organisations represent large companies that produce complex multi-technological products for which SW is just one part of the whole product. This research therefore reflects results that have been gained in the context of the cases, and the analysis of suitability in the other contexts was not carried out. This will be the subject of future research. There is therefore a need to harvest more experiences about the use of the ALM framework in contexts such as other industrial domains, SMEs, IT organisations and companies producing pure SW products.

The purpose of the ALM framework and the improvement/reference models overlaps. ISO 15504, ISO 15940 and CMMI, for instance, are relevant improvement and reference models relating to the ALM framework. In this thesis, the comparison of the ALM framework with existing models was defined as being outside its scope. This does not mean that this should not be done. On the contrary, this thesis enables such a study, as it defines the ALM elements and their relations, and that can be used for such a comparison. Further analysis of using the framework in improvement activities and the way the framework is positioned with regard to existing standards, models and concepts will be an important subject of future research (i.e., usage of the ALM framework in conjunction with the standards and improvement models).

References

Abramovici, M. (2007) *Future Trends in Product Lifecycle Management (PLM)*, The Future of Product Development: Proceedings of the 17th CIRP Design Conference, Berlin, Germany, March 26–28, pp. 665–674.

Anderl, R., Völz, D. and Rollmann, T. (2008) *Knowledge integration in global engineering*, International Conference on Interoperability of Enterprise, Software and Applications, 25–28 March, Berlin, German, pp. 471–482.

Atos Origin. (2003) *Product Lifecycle Management*, White Paper, 08-2003, http://www.atosorigin.com/NR/rdonlyres/2ADE0CAA-4B15-43A6-AA4D-2F6F0D2A8395/0/wp_PLM.pdf (available 26.04.2010).

Battin, R.D., Crocker, R., Kreidler, J. and Subramanian, K. (2001) *Leveraging resources in global software development*, IEEE Software, Vol. 18, Issue 2, March-April 2001, pp. 70–77.

Blyler, J. (2002) *Will Baseband Technology Slow Base-Station Evolution?* Wireless Systems Design. Vol. 7, No. 7, July/August (2002), pp. 19–21.

Booch, G. and Brown, A. (2003) *Collaborative development environments*, Advances in Computers, Vol. 59, Academic Press.

Braa, K. and Vidgen, R. (1999) *Interpretation, intervention, and reduction in the organisational laboratory: a framework for in-context information system research*, Accounting management and information technologies, Vol. 9, No. 1, pp. 25–47.

Brandao, R. and Wynn, M. (2008) *Product Lifecycle Management Systems and Business Process Improvement – A Report on Case Study Research*, Proceedings of the Third International Multi-Conference on Computing in the Global Information Technology (ICCGI '08). Pp. 113–118.

Broy, M. (2006) *The 'Grand Challenge' in Informatics: Engineering Software-Intensive Systems*, Computer, Vol. 39, No. 10, pp. 72–80.

Buckley, F. (1996) *Implementing configuration management: hardware, software, and firmware*, IEEE Computer Society Press, Los Alamitos.

Carmel, E. and Tjia, P. (2005) *Offshore Information Technology: Sourcing and Outsourcing to a Global Workforce*, Cambridge University Press, Cambridge.

Carrillo, J. and McKorkle, S. (2008) *Application lifecycle management meets model-driven development*, Dr. Dobb's Journal, Vol. 33, No. 9, pp. 20–24.

Chappell, D. (2008) *What is application lifecycle management*, Chappell & Associates, White paper.

Collis, J. and Hussey, R. (2003) *Business research: a practical guide for undergraduate and postgraduate students*, Palgrave MacMillan.

Conchúir, E., Holmström, H., Ågerfalk, P. and Fizgerald, B. (2006) *Exploring the Assumed Benefits of Global Software Development*, IEEE International Conference on Global Software Engineering (ICGSE'06), pp. 159–168.

Coplien, J. and Harrison, N. (2005) *Organizational Patterns of Agile Software Development*, Pearson Prentice Hall.

Crnkovic, I., Asklund, U. and Persson Dahlqvist, A. (2003) *Implementing and Integrating Product Data Management and Software Configuration Management*, Artech House, Boston.

Crnkovic, I., Funk, P. and Larsson, M. (1999) *Processing requirements by software configuration management*, Proceedings of the EUROMICRO'99 Conference, Vol. 2, pp. 260–265.

Damian, D. and Moitra, D. (2006) *Guest editors' introduction: global software development: How far have we come?*, IEEE Software, Vol. 23, No. 5, pp.17–19.

Dearle, A. (2007) *Software deployment, past, present and future*, Future of Software Engineering (FOSE '07), 23–25 May, Minneapolis, MN, USA, pp. 269–284.

Doyle, C. (2007) *The importance of ALM for aerospace and defence (A&D)*, Embedded System Engineering (ESE Magazine), Vol. 15, No. 5, June, pp. 28–29.

Doyle, C. and Lloyd, R. (2007) *Application lifecycle management in embedded systems engineerin*g, Embedded System Engineering (ESE magazine), Vol. 15, No. 2, March, pp. 24–25.

Dömges, R. and Pohl, K. (1998) *Adapting traceability environments to project-specific needs*, Communications of the ACM, Vol. 41, No. 12, December 1998, pp. 54–62.

Ebert, C. and De Neve, P. (2001) *Surviving global software development*. IEEE Software, Vol. 18, Iss. 2, March–April 2001, pp. 62–69.

Eclipse (2010) *Eclipse web-pages*, www.eclipse.org (available 26.4.2010).

El-khoury, J., Redell, O. and Torngren, M. (2005) *A tool integration platform for multi-disciplinary development*, 31st EUROMICRO Conference on Software Engineering and Advanced Applications, 30 Aug.–3 Sept. 2005. Pp. 442–449.

Eskeli, J. (2009) *Integrated tool support for hardware related software development*, VTT Publications 725, VTT, Espoo. 83 p. http://www.vtt.fi/inf/pdf/publications/2009/P725.pdf

Espinoza, A. and Garbajosa, J. (2008) *A Proposal for Defining a Set of Basic Items for Project-specific Traceability Methodologies*, Proceedings of the 2008 32nd Annual IEEE Software Engineering Workshop, pp. 175–184.

Espotel. (2010) *Espotel web-pages*, www.espotel.fi (available 26.4.2010).

Estublier, J. (2000) *Software Configuration Management: A Roadmap*. Finkelstein, A. (ed.) The Future of Software Engineering, 22nd International Conference on Software Engineering (ICSE 2000), pp. 279–289.

Estublier, J., Leblang, D., van der Hoek, A., Conradi, R., Clemm, G., Tichy, W. and Wiborg-Weber, D. (2005) *Impact of software engineering research on the practice of software configuration management*, ACM Transactions on Software Engineering and Methodology (TOSEM), Vol. 14, No. 4, October 2005, pp. 1–48.

Fisher, C. (2007) *Researching and writing a dissertation – a guidebook for business students*, Prentice Hall.

Fitzgerald, B., Russo, N. and O'Kane, T. (2003*) Software development method tailoring at motorola*, Communications of the ACM, Vol. 46, No. 4, April, pp. 64–70.

Gills, M. (2005) *Survey of Traceability Models in IT projects*, Proceedings of the ECMDA Traceability Workshop (ECMDA-TW), November 8th 2005, Nuremberg, Germany, pp. 39–46.

Gotel, O. (1995) *Contribution Structures for Requirements Traceability*, Ph.D. Thesis, Imperial College of Science, Technology and Medicine, University of London, August, 1995.

Gotel, O. and Finkelstein, A. (1994) *An Analysis of the Requirements Traceability Problem*, Proceedings of the First International Conference on Requirements Engineering, pp. 94–101.

Goth, G. (2009) *Agile Tool Market Growing with the Philosophy*, IEEE Software, Vol. 26, No. 2, pp. 88–91.

Guckenheimer, S. (2006) *Software Engineering with Microsoft Visual Studio Team System*, Addison Wesley, Upper Saddle River, NJ.

Göthe, M., Pampino, C., Monson, P., Nizami, K., Patel, K., Smith, B. and Yuce, N. (2008) *Collaborative Application Lifecycle Management with IBM Rational Products*, An IBM Redbooks publication, December 2008.

Heindl, M., Reinisch, F. and Biffl, S. (2007) *Requirements management infrastructures in global software development – towards application lifecycle management with role-based intime notification*, International Conference on Global Software Engineering (ICGSE), Workshop on Tool-Supported Requirements Management in Distributed Projects (REMIDI), Munich, pp. 46–51.

Herbsleb, J. and Grinter, R. (1999) *Splitting the organisation and integrating the code: Conway's law revisited*, Proceedings of the 1999 International Conference on Software Engineering, 16–22 May, Los Angeles, California, USA, pp. 85–95.

Holmström Olsson, H., Ó Conchúir, E., Ågerfalk, P. and Fitzgerald, B. (2008) *Two-Stage Offshoring: An Investigation of the Irish Bridge*, MIS Quarterly, Vol. 32, No. 2, pp. 257–279.

IEEE Std-828. (2005) *IEEE Standard For Software Configuration Management Plans*. IEEE Std 828–2005 (Revision of IEEE Std 828–1998).

ISO/IEC 15940. (2006) *Information Technology – Software Engineering Environment Services*. International Organization for Standardization. First edition.

Jalali, S. and Wohlin, C. (2010) *Agile practices in global software engineering – a systematic map*, International Conference on Global Software Engineering, ICGSE 2010, Princeton, NJ, USA, August 23–26, 2010, pp. 45–54.

Jiménez, M. and Piattini, M. (2009) *Problems and solutions in distributed software development: a systematic review*, Software Engineering Approaches for Offshore and Outsourced Development, Second International Conference SEAFOOD

2008, Zurich, Switzerland, July 2–3, 2008, Revised Papers, Lecture Notes in Business Information Processing, Springer, 2009, pp. 107–125.

Kanstrén, T. (2010) *A Framework for Observation-Based Modelling in Model-Based Testing*, VTT Publications 727, VTT, Espoo. 93 p. + app. 118 p. http://www.vtt.fi/inf/pdf/publications/2010/P727.pdf

Kolawa, A. (2006) *The Future of ALM and CM*, CM Journal, CM Crossroads – The configuration management community, January 2006. http://www.cmcrossroads.com/cm-journal-articles/6651-the-future-of-alm-and-cm (available 26.4.2010).

Kotonya, G. and Sommerville, I. (1998) *Requirements Engineering: Process and Techniques*, John Wiley & Sons, Chichester.

Kravchik, M. (2009) *Application Lifecycle Management Environments: Past, Present and Future*, M.Sc. Thesis, The Open University of Israel, Computer Science Division.

Krikhaar, R., Mosterman, W., Veerman, N. and Verhoef, C. (2009) *Enabling system evolution through configuration management on the hardware/software boundary*, Systems Engineering, Vol. 12, No. 3, pp. 233–264.

Kääriäinen, J. (2006) *Practical adaptation of configuration management. Three case studies*, VTT Publications 605, VTT, Espoo. 71 p. + app. 48 p.

Kääriäinen, J., Taramaa, J. and Alenius, J. (2004) *Configuration management support for the development of an embedded system: experiences in the telecommunication industry*. Tools and methods of competitive engineering. Vol. 2. Millpress. The Fifth International Symposium on Tools and Methods of Competitive Engineering (TMCE 2004). Lausanne, CH, 13–7 April 2004. Pp. 605–616.

Lane, M. and Ågerfalk, P. (2009) *Experiences in Global Software Development – A Framework-Based Analysis of Distributed Product Development Projects*. In: Proceedings of the 2009 Fourth IEEE international Conference on Global Software Engineering (July 13–16, 2009). ICGSE. IEEE Computer Society, Washington, DC, pp. 244–248.

Leon, A. (2000) *A Guide to Software Configuration Management*, Artech House, Boston.

Macfarlane, I.A. and Reilly, I. (1995*) Requirements traceability in an integrated development environment*. In: Proceedings of the Second IEEE International Symposium on Requirements Engineering, pp. 116–123.

Meade, P. (2001) *Supporting Payload Processing With Distributed Teams*, Florida Logistics Workshop, Orlando, Florida.

Medina-Domínguez, F., Sanchez-Segura, M., Amescua, A. and García, J. (2007) *Extending microsoft team foundation server architecture to support collaborative product patterns*, International Conference on Software Process (ICSP 2007), Minneapolis, USA, pp. 1–11.

Microsoft. (2010) *Web-pages*, www.microsoft.com (available 26.4.2010).

Miles, M. and Huberman, A. (1994) *Qualitative Data Analysis: An Expanded Sourcebook, 2nd edn*. Sage, Thousand Oaks, California.

Moore, R. Reff, K. Graham, J. and Hackerson, B. (2007) *Scrum at a Fortune 500 Manufacturing Company*, AGILE 2007, pp. 175–180.

Moreira, M., (2004) *Software configuration management implementation roadmap*, John Wiley & Sons, 244 p.

Murta, L., Werner, C. and Estublier, J. (2010) *The Configuration Management Role in Collaborative Software Engineering*, In: Mistrík et al., *Collaborative Software Engineering*, Springer-Verlag, Berlin, Heidelberg, pp. 179–194.

Pederson, J. (2006) *Creating a tool independent system engineering environment*, IEEE Aerospace Conference, 4–11 March 2006.

Pesola, J., Eskeli, J., Parviainen, P., Kommeren, R. and Gramza, M. (2008) *Experiences of tool integration: development and validation*, International Conference on Interoperability of Enterprise, Software and Applications, 25–28 March, Berlin, German, pp. 499–510.

Pikkarainen, M. (2008) *Towards a Framework for Improving Software Development Process Mediated with CMMI Goals and Agile Practices*, VTT Publications 695, VTT, Espoo. 119 p. + app. 193 p., http://www.vtt.fi/inf/pdf/publications/2008/P695.pdf

Pirklbauer, G., Ramler, R.and Zeilinger, R. (2009) *An Integration-Oriented Model for Application Lifecycle Management*, Short paper, ICEIS 2009, 11th International Conference on Enterprise Information Systems, pp. 399–402.

Pollari, M. and Kanstrén, T. (2009) *A Probe Framework for Monitoring Embedded Real-time Systems*, Fourth International Conference on Internet Monitoring and Protection (ICIMP), Venice/Mastre, Italy, 24–28 May 2009, IEEE, Piscataway, NJ, USA, pp. 109–115.

Portillo-Rodríguez, J., Vizcaíno, A., Ebert, C. and Piattini, M. (2010) *Tools to Support Global Software Development Processes: a Survey*, 2010 International Conference on Global Software Engineering (ICGSE 2010), pp. 13–22.

Pretschner, A., Salzmann, C. and Stauner, T. (2004) *Software Engineering for Automotive Systems at ICSE 2004: Workshop Summary*, ACM SIGSOFT Software Engineering Notes, Vol. 29, Issue 5, September 2004.

Ramesh, B. and Dhar, V. (1992) *Supporting systems development by capturing deliberations during requirements engineering*, IEEE Transactions on Software Engineering, Vo. 18, No. 6 , June 1992, pp. 498–510.

Ramesh, B. and Jarke, M. (2001) *Toward reference models for requirements traceability*, IEEE Transactions on Software Engineering, Vol. 27, No. 1, pp. 58–93.

Ramesh, B., Cao, L., Mohan, K. and Xu, P. (2006) *Can Distributed Software Development Be Agile?* Communications of the ACM, Vol. 49, No. 10, pp. 41–46.

Ronkainen, J., Taramaa, J. and Savuoja, A., (2002) *Characteristics of Process Improvement of Hardware related SW*, LNCS 2559: Product Focused Software Process Improvement, 4th International Conference on Product Focused Software Process Improvement, PROFES 2002, Rovaniemi, Finland, December 2002. Springer-Verlag. Berlin Heidelberg, pp. 247–257.

Rossberg, J. (2008) *Pro Visual Studio Team System: Application Lifecycle Management*, Apress; 1 edition, 344 p.

Ruiz-González, F. and Canfora, G. (2004) *Software Process: Characteristics, Technology and Environments*, The European Journal for the Informatics Professional, http://www.upgrade-cepis.org, Vol. 2004, No. 5, pp. 6–10.

Runeson, P. and Höst, M. (2009) *Guidelines for Conducting and Reporting Case Study Research in Software Engineering*, Empirical Software Engineering, Vol. 14, Iss. 2, pp. 131–164.

Schlicter, J., Koch, M. and Burger, M. (1998) *Workspace Awareness for Distributed Teams*, Lecture Notes in Computer Science, Volume: 1364, Coordination Technology for Collaborative Applications Organizations, Processes, and Agents, Eds. Conen, W., Neumann, G., Springer Berlin / Heidelberg, pp. 199–218.

Schwaber, C. (2005) *The Expanding Purview Of Software Configuration Management*, Forrester Research Inc., White Paper, 22 July.

Schwaber, C. (2006) *The Changing Face of Application Life-Cycle Management*, Forrester Research Inc., White Paper, 18 August.

Shaw, K. (2007) *Application Lifecycle Management for the Enterprise*, Serena Software, White Paper, April, http://www.serena.com/docs/repository/company/serena_alm_2.0_for_t.pdf (available 26.4.2010).

Shroff, G., Mehta, A., Agarwal, P. and Sinha, R. (2005) *Collaborative development of business applications*, International Conference on Collaborative Computing: Networking, Applications and Worksharing, 19–21 December, San Jose, CA, USA, pp. 6–15.

Šmite, D., Wohlin, C., Gorschek, T. and Feldt, R. (2010) *Empirical evidence in global software engineering: a systematic review*, *Empirical Software Engineering*, Vol. 15, Iss. 1, pp. 91–118.

Sommerville, I. and Sawyer, P. (1997) *Requirements Engineering: A Good Practise Guide*. John Wiley & Sons, West Sussex.

Stark, J. (2005) *Product Lifecycle Management – 21st Century Paradigm for Product Realisation*, Springer-Verlag London Limited.

Stevens, R., Brook, P., Jackson, K. and Arnold, S. (1998) *Systems Engineering: Coping with Complexity*. Pearson Education.

Sutherland, J., Viktorov, A., Blount, J. and Puntikov, J. (2007) *Distributed Scrum: Agile Project Management with Outsourced Development Teams*, Proceedings of the 40th Annual Hawaii International Conference on System Sciences (HICSS).

Svensson, D. (2003) *Towards Product Structure Management in Heterogeneous Environments*, Doctoral thesis, Chalmers University of Technology, Göteborg, Sweden.

SWEBOK (2004) *SweBok: Guide to the Software Engineering Body of Knowledge*, Abran, A. and Moore, J. (Executive editors), 2004 version, IEEE Computer Society.

Sääksvuori, A. and Immonen, A. (2004) *Product Lifecycle Management*, Springer-Verlag, Berlin.

Taramaa, J. (1998) *Practical development of software configuration management for embedded systems*, VTT Publications 366, VTT, Espoo, 147 p. + app. 110 p.

Toranzo M. and Castro J. (1999) *Multiview++ Environment: Requirements traceability from the perspective of different stakeholders*, WER99 – II IberoAmerican Workshop on Requirements Engineering, Buenos Aires.

Tuuttila, P. and Kanstrén, T. (2008) *Experiences in Using Principal Component Analysis for Testing and Analysing Complex System Behaviour*, ICSSEA 2008 (International Conference on Software & Systems Engineering and their Applications), Paris, France, December 2008.

TWINS (2010) *Project web-pages*, www.twins-itea.org (available 26.4.2010).

Vitikka, J. (2009) *Supporting Database Interface Development with Application Lifecycle Management Solution*, VTT Publications 714, VTT, Espoo, 54 p. http://www.vtt.fi/inf/pdf/publications/2009/P714.pdf

Välimäki A. and Koskimies K. (2006) *Mining Best Practices of Project Management as Patterns in Distributed Software Development*, EuroSPI 2006 Industrial Proceedings, University of Joensuu, Finland, October 2006, pp. 6.27–6.35.

Välimäki, A. and Kääriäinen, J. (2007) *Requirements management practices as patterns for distributed product management*, 8th International PROFES (Product Focused Software Development and Process Improvement) conference, 2–4 July, Riga, Latvia, pp. 188–200.

Välimäki, A. and Kääriäinen, J. (2008) *Patterns for distributed scrum – a case study*, International Conference on Interoperability of Enterprise, Software and Applications, 25–28 March, Berlin, German, pp. 85–97.

Walsham, G. (1995*) Interpretive case studies in IS research: Nature and Method*, European Journal of Informatioin Systems, Vol. 4, No.1, pp. 74–81.

Walsham, G. (2006) *Doing interpretive research*. European Journal of Informatioin Systems, Vol. 15, No. 3, pp. 320–330.

Walsham, G. and Sahay, S. (1999) *GIS for District-Level Administration in India: Problems and Opportunities*, MIS Quarterly, Vol. 23, No.1, pp. 39–66.

Wasserman, A. (1989) *Tool integration in software engineering environments*. In: F. Long, Editor, The International Workshop on Environments (Software Engineering Environments), Lecture Notes in Computer Science vol. 647, Springer-Verlag, Berlin, Chinon, France (1989), pp. 137–149.

Weatherall, B. (2007) *Application lifecycle management – a look back*, CM Journal, CM Crossroads – The Configuration Management Community, January, http://www.cmcrossroads.com/cm-journal-articles/7530-application-lifecycle-management-a-look-back (available 26.4.2010).

Weiß, G., Pomberger, G., Beer, W., Buchgeher, G., Dorninger, B., Pichler, J., Prähofer, H., Ramler, R., Stallinger, F. and Weinreich, R. (2009) *Software engineering – processes and tools*, In: Hagenberg Research, Eds.: Buchberger, B., Affenzeller, M., Ferscha, A., Haller, M., Jebelean, T., Klement, E.P., Paule, P., Pomberger, G., Schreiner, W., Stubenrauch, R., Wagner, R., Weiß, G. and Windsteiger, W., Springer-Verlag, Berlin, Heidelberg, pp. 157–235.

Whitgift, D., (1991) *Methods and Tools for Software Configuration Management*, John Wiley & Sons, England.

Wicks, M. and Dewar, R. (2007) *A new research agenda for tool integration*, The Journal of Systems and Software Vol. 80, No. 9, pp. 1569–1585.

Yin, R. K. (2003) *Case study research: Design and methods*. 3rd edition. Sage, Newbury Park, CA.

Zeller, A. (1997) *Configuration Management with Version Sets – A Unified Software Versioning Model and its Applications*, PhD thesis, Technishe Universität Braunschweig.

Papers I–VI are not included in the PDF version.
Please order the printed version to get the complete publication
(http://www.vtt.fi/publications/index.jsp).

Series title, number and report code of publication

VTT Publications 759
VTT-PUBS-759

Author(s)
Kääriäinen, Jukka

Title

Towards an Application Lifecycle Management Framework

Abstract

Lifecycle Management approaches promise more systematic and efficient ways to support the development and management of complex products. Product Lifecycle Management (PLM) means the activity of managing a company's products across their lifecycles in the most effective way. The concept of Application Lifecycle Management (ALM), on the other hand, indicates the coordination of activities and the management of artefacts (e.g., requirements, source code, test cases) during the software (SW) product's lifecycle. There are surprisingly few scientific efforts to define what ALM constitutes and scientifically reported experiences of the practical development and deployment of ALM solutions in an industrial context. ALM solutions tend to be complex, integrating different tools and practices that are used to produce and manage artefacts during the SW development lifecycle, and there is therefore an apparent need to support the development of such complex solutions for industrial contexts.

This thesis presents an effort towards an ALM framework that can be used to document and analyse an organisation's ALM solution and find improvement ideas for it. The evolving framework has been demonstrated in four industrial case studies and gradually refined based on the experiences gained from the studies. This thesis presents the four case studies to the reader and explains the whole research process from the initial literature study, via four phases of constructing and demonstrating the evolving ALM framework, to a proposal for an ALM framework. Furthermore, the series of case studies revealed several experiences related to the application and improvement of an ALM solution in an industrial context. These experiences are also presented and discussed in this thesis.

ISBN
978-951-38-7728-6 (soft back ed.)
978-951-38-7729-3 (URL: http://www.vtt.fi/publications/index.jsp)

Series title and ISSN	Project number
VTT Publications 1235-0621 (soft back ed.) 1455-0849 (URL: http://www.vtt.fi/publications/index.jsp)	24506

Date	Language	Pages
March 2011	English	103 p. + app. 81 p.

Name of project	Commissioned by

Keywords	Publisher
Application lifecycle management, global software development, software configuration management	VTT Technical Research Centre of Finland P.O.Box 1000, FI-02044 VTT, Finland Phone internat. +358 20 722 4520 Fax +358 20 722 4374

ISBN 978-951-38-7728-6 (soft back ed.)
ISSN 1235-0621 (soft back ed.)

ISBN 978-951-38-7729-3 (URL: http://www.vtt.fi/publications/index.jsp)
ISSN 1455-0849 (URL: http://www.vtt.fi/publications/index.jsp)