TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

Antti Välimäki

Pattern Language for Project Management in Global Software Development

Julkaisu 970 • Publication 970

Tampere 2011

Tampereen teknillinen yliopisto. Julkaisu 970
Tampere University of Technology. Publication 970

Antti Välimäki

Pattern Language for Project Management in Global Software Development

Thesis for the degree of Doctor of Science in Technology to be presented with due permission for public examination and criticism in Tietotalo Building, Auditorium TB109, at Tampere University of Technology, on the 10th of June 2011, at 12 noon.

Tampereen teknillinen yliopisto - Tampere University of Technology
Tampere 2011

ISBN 978-952-15-2581-0 (printed)
ISBN 978-952-15-2616-9 (PDF)
ISSN 1459-2045

ABSTRACT

Globalization is a general trend in the world which will also affect the way software is developed and the kinds of organization which will make this development. Global Software Development (GSD) offers new benefits but also new challenges which make the work more difficult than earlier. This thesis tries to find solutions to these challenges from the viewpoint of project management. Rather than developing a totally new GSD process that addresses these problems, I try to come up with solutions to specific problems, and present these solutions in such a way that they can be easily integrated with existing processes. An attractive way to document proven solutions to specific development process problems is to use process patterns as done in this work.

Pattern mining is not an easy task and different methods have been used to mine patterns. In this thesis, a pattern mining method has been developed which is applied in case projects. The solutions in these patterns have been mined from the practices that have been found to work well in a case organization.

How to apply patterns is not always clear, because usually there are different kinds of patterns for different purposes. Patterns should be organized in a way which makes it easier to use them. In this work a process view of project management has been used to organize the found process patterns.

When patterns are found, it is not always easy to tell if they are good or not. In this work, a new approach to evaluate process patterns was developed. The resulting patterns have been evaluated by using the proposed scenario-based assessment method.

To summarize, the main contributions of this thesis consist of a method to find GSD process patterns, a collection of process patterns for project management in GSD, a pattern language organization based on the structure of the PRINCE2 project management process, and finally, a new assessment method to evaluate process patterns.

Keywords: Process patterns, Global software development, Agile project management

Preface

The work was carried out at Metso Automation as part of the EU ITEA/ITEA2 research projects TWINS (Optimizing HW-SW Co-Design Flow for Software Intensive System Development) and ITEI (Information Technologies supporting the Execution of Innovation Projects) during 2006-2011.

I wish to thank my supervisor, Professor Kai Koskimies for his excellent guidance and encouragement during these years. I also wish to thank PhD Ita Richardson and Professor Ilkka Tervonen for reviewing the thesis and giving constructive feedback. I would also like to thank Professor Ilkka Haikala in memoriam for his valuable feedback and support of my thesis. I am grateful to Jukka Kääriäinen, Minna Pikkarainen and Sari Vesiluoma, who provided valuable support and cooperation during the research work, and to Sara Verville who reviewed the language of all my papers and the thesis. Furthermore, many thanks to my colleagues Mika Vanne, Jukka Ylijoki, Pekka Kimpimäki, Paavo Parkkonen, Juha Viljamaa and many other persons at Metso Automation and other companies who made this work possible.

I would also like to acknowledge the financial support provided by Metso Automation and the Academy of Finland and the research support provided by SoSE (The Graduate School on Software Systems and Engineering).

Finally, I would like to extend my gratitude to my wife Merja, and my children Matti and Mika and my parents Mirja and Esko Välimäki.

Tampere, April 2011

Antti Välimäki

Contents

List of Figures

List of Tables

List of Publications

I. Välimäki, A., Kääriäinen, J. *Requirements Management Practices as Patterns for Distributed Product Management*. In: Munch, Abrahamsson, . (ed.), Product-Focused Software Process Improvement, PROFES 2007, Riga, Latvia, July 2007, LNCS 4589, Springer (2007), pp. 188-200.

II. Välimäki, A., Kääriäinen, J. *Patterns for Distributed Scrum – 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 Approached. Springer (2008), pp. 85 – 97.

III. Kääriäinen, J.; 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.

IV. Välimäki, A., Vesiluoma, S. and Koskimies, K. *Scenario-Based Assessment of Process Pattern Languages*. In: Proceedings of Profes 2009, June 2009, (Eds). Springer (2009), pp. 246 – 260.

V. 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. Eds: R.V. O’Connor et al. (Eds.): EuroSPI 2009, Springer, Berlin - Heidelberg (2009), pp. 137 – 148.

VI. Kääriäinen, J, 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. Eds: R.V. O’Connor et al. (Eds.): EuroSPI 2009, Springer, Berlin - Heidelberg (2009), pp. 149 – 160.

1 INTRODUCTION

This chapter gives a general introduction to the setup and results of the thesis. The context, problem statement and approach of this research are first discussed. The research method and contributions are briefly described and finally, the structure of the thesis is presented. I assume that the reader is familiar with project management in software development.

1.1 Context

Globalization is a general trend in the world and it will also affect the way software is developed in companies. Global Software Development (GSD) can be defined as software development that uses teams from multiple geographic locations [Sangwan et al. 2006, p.3]. GSD offers new motivators but also new challenges which make development of software more difficult than earlier. The research of GSD tries to find solutions to address these challenges.

The motivation for using GSD in companies is that it brings benefits such as access to cost savings, worldwide talent, knowledge networks, and follow-the-sun development [Carmel and Tija 2005, pp. 94-100]. Other reasons for GSD implementation are mergers and acquisitions made by companies and companies' need to be close to a local market. Enablers for GSD are advances in infrastructure and software development tools. However, GSD is also a challenge for software corporations because there are breakdowns of communication, coordination and control and also cohesion barriers between different groups during the development and project management [Carmel and Tija 2005, pp. 12-13]. Other challenge categories are legal issues, knowledge transfer, quality management as well as culture, language and time differences and infrastructure [Kobitzsch et al. 2001].

Project management is one of the key software development processes to cope with these challenges. In fact, it has a central role in coordinating and controlling how software is developed. Project management is a process which strives to ensure that tasks are done within a given budget and time schedule. Project management is a challenging activity in the best of circumstances, and it has become even more challenging in distributed scenarios [Kock 2008, p.315].

Different kinds of development models can be used in GSD such as plan-driven models, Agile models, or a combination of these. Agile development methods include e.g. Extreme Programming (XP) [Beck et al. 2004], Feature Driven Development (FDD) [Palmer and Felsing 2002], and Scrum [Schwaber and Beedle 2002]. Agile methods have been originally planned to be used in collocated projects [Abrahamsson et al. 2002]. However, there is an increasing trend of using some of the Agile practices to improve the efficiency and quality of GSD project management ([Ramesh et al. 2006], [Sutherland et al. 2007], and [Farmer 2004]).

GSD is becoming more and more common in the software development industry ([Conchuir et al. 2009], and [Šmite et al. 2010]) due to the various GSD benefits mentioned earlier.

However, there are still many problems to be solved in complex, distributed software development environments. In this thesis I adopt the viewpoint that many of these problems can be alleviated by improved project management. In particular, I argue that GSD practitioners and researchers need to identify better project management practices in order to address the distribution challenges both in traditional plan-driven and in Agile processes. The identification of such practices is the main objective of this work. I expect that the application of the identified practices provide improvements in functionality, efficiency and adaptability in the GSD process.

There is also a presentation problem involved: how to present the knowledge of good practices in a way that is easy to access, understand and reuse? In this thesis, I explore the use of the pattern concept [Alexander et al. 1977] for presenting the recommended practices in project management in GSD, originally introduced for the design of building architecture [Alexander et al. 1977]. According to Alexander, each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem in such a way that this solution can be used when the problem reappears. [Alexander et al. 1977]. Patterns have been applied in various contexts of software engineering, also for describing process practices e.g. [Coplien and Harrison 2005].

Patterns are not isolated solutions but they are often applied in concert to solve a larger problem. A pattern language is a collective of patterns which, at every level of scale, work together to resolve a complex problem into an orderly solution according to a pre-defined goal [Appleton 1997]. The central aim of this work is to develop a comprehensive pattern collection and a pattern language for project management in GSD. In the next section I discuss the problem statement in more detail.

1.2 Problem Statement

GSD offers many new possibilities to improve the efficiency of software development but it is much more difficult to implement than collocated development. GSD usually involves geographical, temporal and socio-cultural distances that may create a number of challenges and may restrict project communication, coordination and control processes [Ågerfalk et al. 2005]. Thus, it is necessary for GSD project managers to mitigate project management challenges by introducing better processes, practices or tools that will ensure effective project communication, coordination and control processes. GSD challenges can be studied on various levels, but in this thesis the focus is on the project level.

The primary objective of this thesis is to find and represent effective practices that enable to minimize project distribution challenges and also provide better GSD project management. A sub-problem related to the primary objective is to represent these practices in a systematic and easily understandable form. Another sub-problem is the definition of a method for finding good practices in research projects, and literature. Finally, the resulting representation of the practices has to be evaluated systematically to ensure that it fulfills its purpose.

Thus, the broad objective of this research is to address the following research questions:

RQ1: How should good project management practices in GSD projects be presented?

RQ2: How can good project management practices be found in GSD projects?
RQ3: What are the good project management practices in GSD projects?
RQ4: How can these practices be evaluated?

1.3 Research Approach and Methods

Information concerning challenges and solutions of GSD in this thesis is gathered both from industry and literature. Information from industry is gathered by questionnaires and interviews from GSD projects which have been managed using either the traditional plan-driven method or a combination of plan-driven and Agile methods. Project Management practices are studied from the client perspective in which the case company is a client and its subcontractors are vendors located in different sites. Good project management practices are reviewed and improved in interviews and workshops. Literature plays a corroborating role for identifying good practices: the appearance of a similar practice recommendation in the literature confirms the justification for the practice.

Gathered information is presented in the form of patterns. A collection of process patterns for project management in GSD is organized into a pattern language (GSD Pattern Language) based on the project management processes of PRINCE2 (PRojects IN Controlled Environments2) [Bentley 2005]. Process patterns are in general difficult to evaluate empirically in a real environment because it would mean time and resource consumption monitoring of the effect of the patterns in different projects, possibly in different companies. Even the "effect" is difficult to define and measure. Thus it was necessary in this work to develop a more feasible method for evaluating the resulting pattern language. The evaluation is carried out here using a scenario-based assessment method which is derived from ATAM (Architecture Tradeoff Analysis Method) [Clements et al. 2002].

The industrial context in this research is a company which operates in the automation industry. The company produces complex automation systems where software development is a part of system development. Product development in the case company is organized according to product lines. Most of the research focused on two product lines consisting of several products that were partly developed in different sites. The cases were further focused on two teams each having 5 to 6 projects running in parallel. The projects were geographically distributed over 2 or 3 sites. The industrial context is described in more detail in Section 3.3.

In this thesis the research approach is based on design science [Hevner et al. 2004] and case studies [Yin 2003].

Design science has two parts: to build and to evaluate [Hevner et al. 2004]. The build part was implemented by the case study approach, resulting in a pattern language that reflects the problems and solutions in a representative GSD environment. In general, case studies are useful when research data is collected through observations [Yin 2003]. Case study research was implemented based on Yin's steps in case study research methods, i.e. preparing data collection, collecting evidence, analysing evidence and reporting case studies. The evaluation part of the pattern language was implemented both by the new process pattern assessment method and by literature-based GSD challenges.

Hevner et al. [2004] have described seven guidelines for design science in information systems research. The seven guidelines and how these guidelines are covered in this thesis are presented in Table 1.

Table 1. An analysis of Design Science Research Guidelines [Hevner et al. 2004]

Guideline	**Description**	**How covered in this thesis?**
Guideline 1: Design as an Artifact	Design-science research must provide a viable artifact in the form of a construct, a model, a method or an instantiation.	In this thesis, the artifact is GSD Pattern Language.
Guideline 2: Problem Relevance	The objective of design-science research is to develop technology-based solutions to important and relevant business problems.	The business problems are related to project management in agile GSD projects.
Guideline 3: Design Evaluation	The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods.	Evaluation is made using a new process pattern assessment method, Quality-oriented Process Assessment Method (Q-PAM).
Guideline 4: Research Contributions	Effective design-science research must provide clear and verifiable contributions in the areas of the design artifact, design foundations, and/or design methodologies.	This thesis provides GSD Pattern Language.
Guideline 5: Research Rigor	Design-science research relies upon the application of rigorous methods in both the construction and the evaluation of the design artifact.	Construction is carried out using a Pattern mining method and evaluation is based on Q-PAM.
Guideline 6: Design as a Search Process	The search for an effective artifact requires utilizing available means to reach desired ends while satisfying laws in the problem environment.	Construction relies on the Pattern mining method which aims at systematic search.
Guideline 7: Communication of Research	Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences.	This thesis describes results at a project management level which takes into account both technology and management audiences.

The overall flow of the research conducted is presented in Table 2 which describes each step of the research. For each research step, the associated paper or an action, the research theme, and the research questions are given, as well as a classification of the nature of the research work. The first three steps introduce pattern mining methods and results of the case studies applying the methods. The fourth step gives information for a specific Application Lifecycle Management (ALM) pattern. In steps five and six GSD Pattern Language is created. Step seven concerns the development of a new process pattern assessment method (Q-PAM), while in step eight this method is used to evaluate GSD Pattern Language. Finally, step nine, gives updated information for ALM patterns.

Table 2. The overall flow of the research

Research step	**Theme**	**Research interest**	**Research method**
1) [Välimäki and Koskimies 2006]	The best practices of project managers in plan-driven projects in a GSD environment.	What are best practices in a form of process patterns for project managers?	Case study
2) Paper I / 2007	Requirements management practices as patterns for distributed product Management	What are the best practices to support distributed business requirements management during the early phases of product development? What is the process of mining requirements management process patterns?	Case study
3) Paper II / 2008	The best practices of project managers in a distributed Agile development environment.	What are project management patterns in a distributed Agile development environment? How could the pattern mining method be improved?	Case study
4) Paper III / 2008 (co-author)	ALM related best practices	What is the content of ALM related pattern?	Case study
5) Study of research material / 2008	GSD Patterns for project managers	Which patterns are chosen to GSD patterns for project managers?	Design meetings
6) Review workshops / 2008	Review of GSD Patterns	Which patterns are chosen to GSD patterns for project managers?	Workshops

7) Paper IV / 2009	Scenario-Based Assessment of Process Pattern Languages	What kind of light-weight method can be used for assessing pattern-based processes and how can such a method be applied?	Workshops
8)Paper V / 2009	Introduction of GSD Pattern Language and its validation using the validation method described in Paper IV.	What is the content of GSD Patterns and what are the results of validation of GSD Pattern Language by Q-PAM method?	Workshops
9)Paper VI / 2009 (co-author)	An update to ALM related best practices	What is updated information about the ALM-related process pattern?	Case study

1.4 Contributions

The main contributions of this thesis are the following:

1. A method to find process patterns, based on identifying and analyzing challenges, current practices and ideas to improve current practices in case projects.

2. A collection of process patterns for project management in GSD.

3. GSD Pattern Language for project management organization based on the structure of the PRINCE2 project management process. (GSD Pattern Language)

4. A new Quality-oriented Process Assessment Method (Q-PAM) to evaluate process patterns.

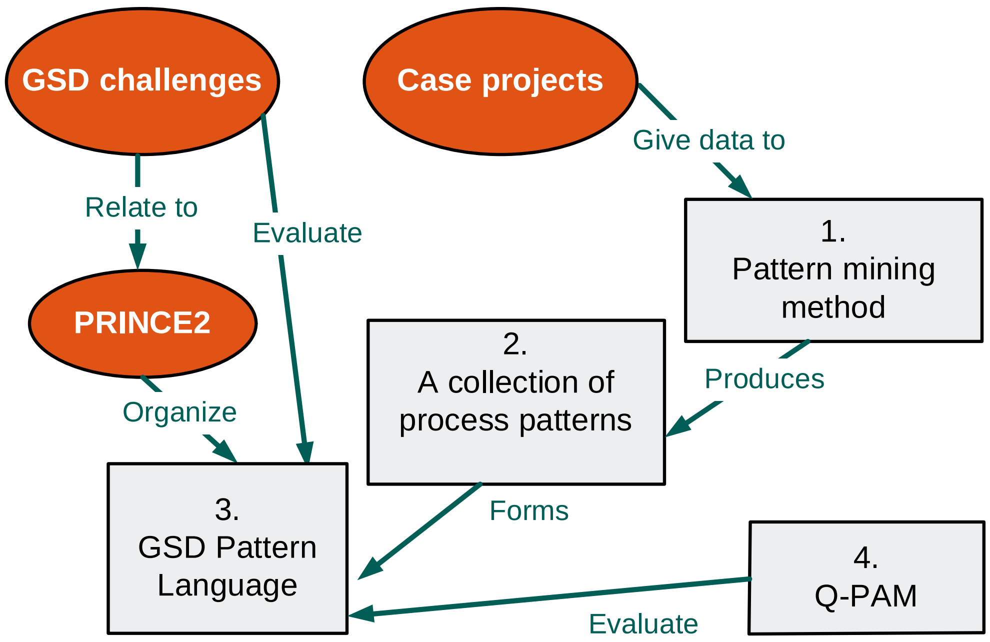


Figure 1. The main contributions of the thesis

The main contributions, the related elements and the relations between them are illustrated in Figure 1. The related elements which are not regarded as contributions to this research are literature based GSD challenges, PRINCE2, and Case projects.

The main contributions are presented as numbers in Figure 1. *A pattern mining method* (1st contribution) is used to find *a collection of process patterns for project management in GSD* (2rd contribution) from Case projects. PRINCE2 is used to organize the collection of process patterns for *GSD Pattern Language for project management* (3th contribution). *Quality-oriented Process Assessment Method* (4th contribution) is used to evaluate GSD Pattern Language. Also the literature-based GSD challenges are used as the basis of another evaluation of GSD Pattern Language.

1.5 Structure of the Thesis

This thesis includes an introduction, six previously published articles, and one appendix. The introduction consists of eight chapters. Chapter 1 gives a brief introduction to the motivation and contents of this thesis and Chapter 2 gives background information about the relevant research areas. In Chapter 3, the pattern mining method is presented and in Chapter 4 the GSD Process Pattern Language for project management in GSD is presented. In Chapter 5, the new evaluation method and its application to the GSD Process Pattern Language are presented. Chapter 6 discusses related work, Chapter 7 gives an introduction to the included publications. Finally, in Chapter 8 are the conclusions.

The thesis is based on six publications. Paper I discusses pattern mining (Chapter 3). This paper includes also process patterns which are needed when the fuzzy front end of a development project is clarified by product managers in GSD (Chapter 4). Paper II also discusses pattern mining (Chapter 3) and presents the first project management patterns in a distributed Agile development environment (Chapter 4). Paper III introduces the framework for project management and development tools and processes in the form of the Application Lifecycle Management (ALM) concept and gives information about ALM-related patterns (Chapter 4). Paper IV describes a new scenario-based assessment method for evaluating process pattern languages (Chapter 5). Paper V presents both the first version of Global Software Development Patterns for Project management (Chapter 4) and evaluation of these patterns (Chapter 5). Paper VI presents the latest ALM framework and updated information about the ALM-related process pattern (Chapter 4).

The first contribution is presented in Papers I and II. The first parts of the second contribution are presented in Papers I and II and related information for the second contribution is presented in III and VI. The third contribution is presented in Paper V. Finally, the fourth contribution is presented in Papers IV and V.

2 BACKGROUND

This chapter discusses the relevant background information related to this research. Section 2.1 introduces Global Software Development and PRINCE2. Section 2.2 discusses Agile Methods and Agile Project Management in GSD, in particular Scrum [Schwaber and Beedle 2002]. Section 2.3 briefly presents Application Lifecycle Management (ALM). ALM-based approaches are often exploited in a GSD environment. Finally, patterns and pattern languages are discussed in Section 2.4. Patterns are a central utility in this work, enabling the presentation of the results in a systematic form.

2.1 Global Software Development

This section discusses GSD as far as it is related to this research. Subsection 2.1.1 discusses GSD definitions and presents the definition used in this thesis. Subsection 2.1.2 discusses different GSD co-operation models. Subsection 2.1.3 discusses CCC definitions in GSD and broad categorization of GSD challenges in general. The project management process PRINCE2 is used later to organize a pattern language from a collection of process patterns for project management. Subsection 2.1.4 summarizes GSD challenges discussed in literature. Subsection 2.1.5 introduces briefly PRINCE2.

2.1.1 Definition of GSD

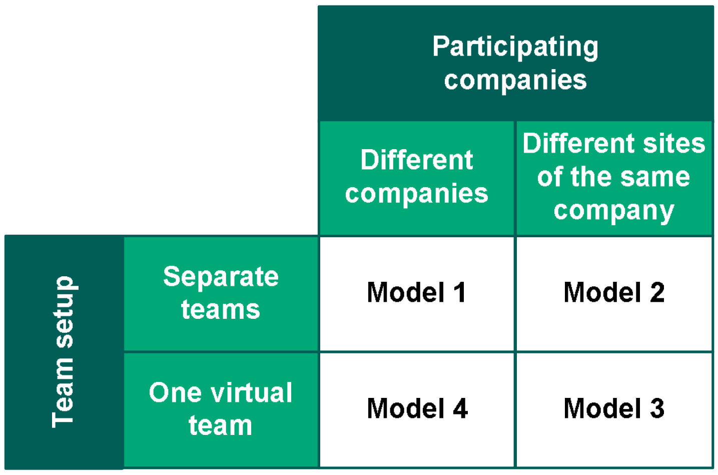
The concept of Global Software Development (GSD) is very broad and can be used to define any contemporary form of software development where project stakeholders are dispersed in distributed locations. One example of a GSD definition is by Sangwan stating that GSD is software development that uses teams from multiple geographical locations. In some cases, these teams may be from the same organization, in other cases, there may be collaboration or outsourcing that involves different organizations. These teams could be within one country or on the other side of the world. [Sangwan et al. 2006, pp. 3-4]. Another GSD definition is according to Wiredu defining GSD as a contemporary form of software development undertaken in globally distributed locations and facilitated by advanced information and communication technology (ICT), with the predominant aim of rationalizing the development process [Wiredu 2006]. Yet another GSD definition is according to Conchuir et al. [2006] saying that GSD is characterized by stakeholders from different national and organizational cultures, located in separate geographic- locations and time-zones, using information and communication technologies to collaborate. Based on these definitions, GSD is defined in this thesis as follows:

GSD is a software engineering approach where stakeholders from different national and organizational cultures, located in separate geographic locations and possibly different time zones, use information and communication technologies to collaborate on implementing needed products.

GSD is defined in a slightly new way in this thesis to emphasize that GSD is a software engineering approach and that the results are software or system products.

2.1.2 Different GSD Co-Operation Models and Team Setups

GSD development can be organized based on different models which affect not only the challenges related to GSD organization but also the solutions to solve these challenges. In Figure 2, it is possible to see different organization models related to GSD. One organizational criterion is how GSD companies relate to each other. They can be different companies or different sites of the same company. Another criterion is what the team setup is. Teams can be separate or they can constitute one virtual team.



		Participating companies	
		Different companies	Different sites of the same company
Team setup	Separate teams	Model 1	Model 2
	One virtual team	Model 4	Model 3

Figure 2. The four cooperation models according to company relationship and team setup adapted from [Kobitzsch et al. 2001]

The first model covers separate teams which are in different companies. This is a normal contractor-subcontractor relationship between two separate companies. The second model also covers separate teams, but these teams are in different sites of the same company. Holmström Olsson et al. [2008] note that separate teams (i.e. loosely coupled teams) work more independently than a virtual team. Communication and day-to-day management between teams can be kept to a minimum.

The third model covers one virtual team which is distributed across multiple sites of different sites of the same company. The fourth model covers one virtual team which is also distributed across multiple sites of at least two different companies. This means that there can be e.g. one team in a main company and at least one team in another company which also constitute the same virtual team. Holmström Olsson et al. [2008] note that virtual teams (i.e. tightly coupled teams) facilitate organizational unity and “teamness”, but tightly coupled teams also create additional communication and managerial overhead.

2.1.3 Communication, Coordination, and Control in GSD

In this subsection CCC (Communication, Coordination, and Control) is discussed in the context of GSD. CCC has been used as a framework to study GSD challenges, proposed by Ågerfalk [Ågerfalk et al. 2005]. I also use CCC in this thesis for classifying GSD challenges, which are presented in more detail in Subsection 2.1.4.

Communication is sending, giving, or exchanging information and ideas, which is often expressed nonverbally and verbally. Communication can also be defined as the exchange of complete and unambiguous information in which the sender and receiver can reach a common understanding [Carmel and Agarwal 2001]. Project members’ distribution usually changes the communication context away from an “ideal” face to face setting into a more restricted technology-oriented environment [Holmström et al. 2006]. According to Mockus and Herbsleb [2001] factors like time zone differences, language and cultural differences, communication bandwidth, and multi-site development have an impact on GSD communication processes.

Communication problems are e.g. the loss of communication richness with tools such as e-mail or telephone, difficulties to understand each other over distance [Carmel and Tija 2005]. The frequency of communication between team members may decrease with distance. It has been observed that the frequency of communication is almost the same when members are about 30 meters or many miles from each other. [Allen 1984]. These problems lead to delays and to rework. Delays are created because messages need to be clarified and reworked and because they were not really understood in the first place. This miscommunication leads in turn to conflicts and decreased trust between members of a project in different sites [Carmel and Tija 2005].

Coordination is the act of integrating each task with each organizational unit so the unit contributes to the overall objective [Carmel and Agarwal 2001]. Coordination is also usually defined as managing dependencies among various software development tasks. In GSD projects, tasks are distributed over time, space and across cultural borders; this increases the need for coordination processes [Herbsleb and Moitra 2001]. Factors like increased number of distributed sites and lack of tool support may have an impact on GSD coordination processes. A coordination problem is e.g. the difficulty to make fast, small or large adjustments to the work by members of a project because of distance and differences in time zones. Adjustments are made by e.g. questions, requests for clarification or fast ad hoc meetings and these are easier and faster to implement in collocated development because one is aware of who to contact for help. This coordination problem creates delays with problem solving and projects often head in the wrong direction due to lack of proper coordination information. [Carmel and Tija 2005].

Control is the process of adhering to goals, policies and standards, or quality levels [Carmel and Agarwal 2001]. Control is difficult in GSD basically because a project manager cannot implement Management by Walking Around (MBWA) [Carmel and Tija 2005]. Another problem is that managers usually pay more attention to those who are near. These problems with control also increase delays and make it more difficult to find problems early enough.

2.1.4 Challenges in GSD

GSD challenges have an impact on communication, coordination and control (CCC) in software development processes. These arise due to the distances involved in three dimensions – temporal, geographical and socio-cultural. In the following I will briefly describe GSD challenges related to distances and CCC. After that, I will present GSD challenges according to the framework proposed by Ågerfalk. [Ågerfalk et al. 2005].

Temporal distance across multiple time zones reduces the number of overlapping working hours between two or more distributed parties. *Geographical distance* is related to the accessibility of locations between collaborating parties which includes physical distance and issues related to transportation or political border controls [Lane and Ågerfalk 2009]. *Socio-cultural distance* is related to the people from different countries who have different backgrounds, values, work habits, language etc. [Ågerfalk et al. 2005].

There are many challenges with communication. Carmel [1999] mentioned that a major challenge for GSD teams is the lack of informal communication which is very important in any software development process. Also Conchuir et al. [2009] note that it is difficult to increase communication between team members because geographical distance makes it more difficult to meet colleagues from other locations. Although face-to-face meetings are very important for team members, it is often too expensive for everyone to travel from one site to another to meet each other. This makes it difficult to organize face-to-face meetings in GSD.

Other challenges for communication are lack of overlapping working hours which can also lead to delays in feedback thus decreasing the effectiveness of the development process [Conchuir et al. 2009]. Also Herbsleb and Mockus [2003] claim that modification requests (MRs) involving multiple sites took 2,5 longer to resolve and usually need more people to solve modification requests, compared to single-site MRs. One solution is to increase asynchronous communication. Kobitzsch et al. [2001] note that an advanced communication infrastructure is a key component for GSD because team communication becomes much more difficult when the participants are geographically dispersed but there is a risk with communication dependency on ICT tools [Hossain et al. 2009]. Mockus and Herbsleb [2001] also claim that one potentially serious issue is that project participants have backgrounds that are different in each site. They have not been in the same projects, have used different processes, come from different cultures and speak different native languages which all make communication more difficult.

Key risks of using different coordinating mechanisms in GSD projects are also presented in [Hossain et al. 2009]. Hossain et al. [2009] divide coordination into three basic coordination mechanisms according to [Mintzberg 1989] mutual adjustment, direct supervision and standardization. Risks are presented from the viewpoint of GSD distances (temporal, geographical and socio-cultural) and three coordination mechanisms.

Different distances in GSD are described to impact negatively for key variables for success in GSD. These variables are effective coordination, visibility, communication and cooperation

[Casey and Richardson 2006]. Another analysis from the viewpoint of GSD distances is in Table 3. It presents GSD challenges which have an impact on communication, coordination and control due to geographical, temporal and socio-cultural distance adapted from [Ågerfalk et al. 2005]. As Ågerfalk et al. [2005] describe, GSD challenges are related to each other and the positioning used in the table can be questioned. However, Table 3 aims to bring a primary order to the complex interrelationship between the challenges.

Table 3. GSD challenges adapted from [Ågerfalk 2005]

CCC/Distances	**Temporal Distance**	**Geographical Distance**	**Socio-cultural Distance**
Communication	Reduced synchronous communication (1)	Face-to-face meetings difficult (4)	Cultural misunderstandings (9)
Coordination	Typically increased coordination costs (2)	Reduced trust (5) and A lack of critical task awareness (6)	Inconsistent work practices (10) and Reduced cooperation (11)
Control	Management of project artifacts (3)	Difficult to convey strategy (7) and Low-cost “rivals” (8)	Different perceptions of authority (12) and Managers must adapt to local regulations (13)

Table 3 is organized based on the GSD project distances which are broadly classified as temporal, geographical and socio-cultural distances that have an impact on communication, coordination and control in software development processes.
The first challenge is *Reduced opportunities for synchronous communication (1)*. GSD teams often suffer response delays due to the unavailability of remote team members in different time zones. Further delays were incurred due to the misinterpretation of emails or voicemails [Lane and Ågerfalk 2009].

Typically increased coordination costs (2) challenge is related to temporal distance and coordination: GSD teams do not have as many overlapping working hours as in collocated teams. This can mean e.g. that there can be coordination meetings which are late at night or early in the morning depending on the time zone difference between different sites [Battin et al. 2001].

Management of project artifacts may be subject to delays (3) challenge is related to challenges with temporal distance and control because e.g. scheduling of certain artifact review meetings is difficult due to the reduced availability of different GSD team members [Lane and Ågerfalk 2009]. Lane and Ågerfalk [2009] also note that many project dependencies are seriously impacted when additional review meetings are required due to the failure of an artifact to pass its review. One day of rework could result in a two-week delay due to scheduling difficulties. Evaristo et al. [2004] also note that distributed development may cause difficulties in maintaining standard project artifacts throughout the sites.

Travel difficulties due to geographical distance create a *Face-to-face meetings difficult (4)* challenge between distributed project stakeholders. This situation leads to a communication breakdown [Carmel and Tija 2005] and there is e.g. loss of communication richness with tools like e-mail or telephone and difficulties to understand each other over distance [Carmel and Tija 2005]. Communication problems lead to delays and to rework. Delays are created because messages need to be clarified. Rework is needed because it was not really understood what was meant the first time round. [Carmel and Tija 2005].

Reduced informal contact can lead to reduced trust (5) challenge is related to geographical distance and coordination. Lack of informal communication due to geographical distance is one of the major obstacles in building trust among distributed project stakeholders. As described earlier, there can be problems with communication which can lead to miscommunication and which can lead to further conflicts and decrease of trust between members of a project in different sites ([Carmel and Tija 2005] and [Moe and Šmite 2008]) Problems with communication can also lead to *A lack of critical task awareness* (6) challenge.

Difficult to convey vision and strategy (7) challenge is mainly due to the project's geographical distance and impact on the project's control process. GSD strategy can be implemented with different phase-based process structures which can have different benefits and difficulties [Lane and Ågerfalk 2009].

Perceived threats from training low-cost "rivals" (8) is another challenge related to geographical issues caused by the syndrome "My job went to India and all I got was this lousy T-shirt" which can lead to unwillingness to facilitate knowledge transfer from the main site to other sites [Holmström Olsson et al. 2008]

Cultural misunderstanding (9) challenge due to socio-cultural distance may have an impact on the project communication process. This challenge leads to poor depth of communication caused by dilution and communication errors due to cultural differences [Carmel and Tija 2005]. Although English is a common language in projects, there are differences with language competency between team members and also different dialects and local accents can make communication inefficient [Ågerfalk et al. 2005].

Inconsistent work practices can impinge on effective coordination(10) challenge is related to socio-cultural distance and coordination. Usually different sites have different processes and templates which make it difficult to understand different sites process and result deliverables [Battin et al. 2001]. These problems increase coordination work with different sites. Also *Reduced cooperation arising from misunderstandings (11)* challenge increases coordination work [Herbsleb and Grinter 1999]

Different perceptions of authority can undermine morale (12) challenge is related to socio-cultural distance and control. For this reason, management control over the team highly depends on the culture and also geography.

Managers must adapt to local regulations (13) challenge is also related to socio-cultural distance and control. This challenge emphasizes the fact that management of different teams with a different culture is an important issue to solve in order to establish an efficient GSD project.

2.1.5 PRINCE2

PRINCE2 version 2005 includes three parts which are processes, components and techniques [Bentley 2005]. PRINCE2 is a project management method which is in use in over 50 countries around the world and over 20 000 organizations utilizes the method guidelines of PRINCE2 [Smith 2009].

PRINCE was established in 1989 by CCTA (the Central Computer and Telecommunications Agency), since renamed the OGC (the Office of Government Commerce). PRINCE was originally based on PROMPT, a project management method created by Simpact Systems Ltd in 1975. PROMPT was adopted by CCTA in 1979 as the standard to be used for all Government information system projects. When PRINCE was launched in 1989, it effectively superseded PROMPT within Government projects. It remains in the public domain and copyright is retained by the Crown. It is a registered trademark of OGC. PRINCE2 was published in 1996, having been contributed to by a consortium of some 150 European organizations. In 2002 and 2005 PRINCE2 was updated in consultation with the international user community. PRINCE2 was updated in 2009 to make it simpler and easier to customize. [Haughey 2010].

The PRINCE2 version 2005 (hereafter called PRINCE2) project management process overview is presented in Figure 3. PRINCE2 is comprised of eight major processes which are collections of subprocesses. PRINCE2 will be used later to organize a pattern language from a collection of process patterns for project management in Section 4.3.

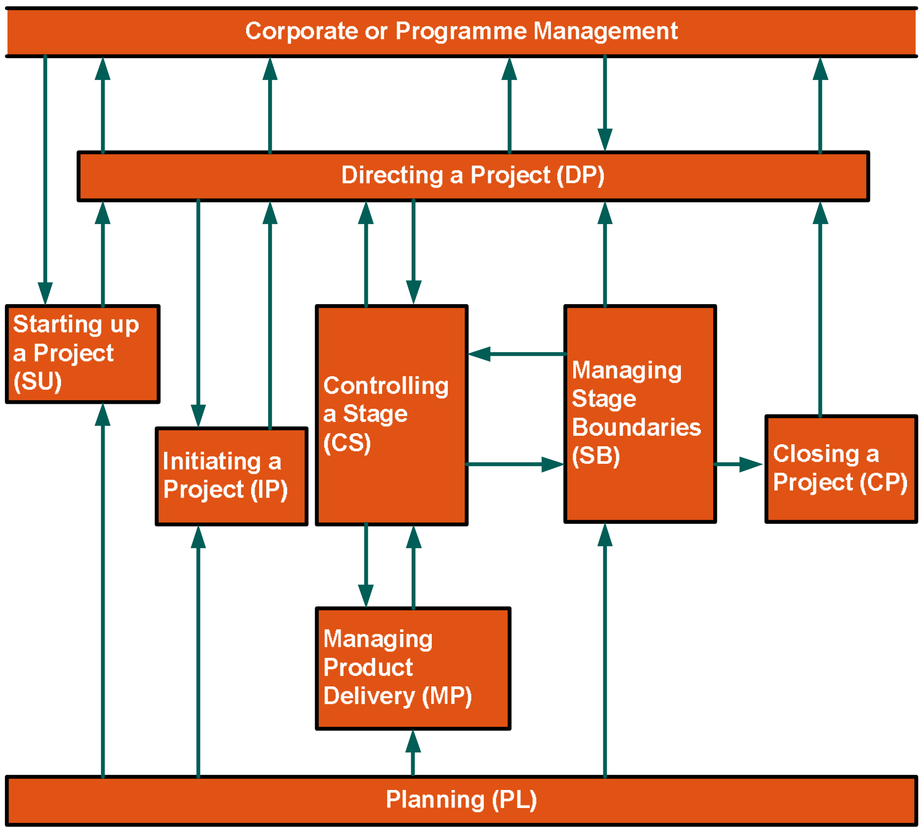

Corporate or Programme Management
Directing a Project (DP)
Starting up a Project (SU)
Initiating a Project (IP)
Controlling a Stage (CS)
Managing Stage Boundaries (SB)
Closing a Project (CP)
Managing Product Delivery (MP)
Planning (PL)

Figure 3. Process overview of PRINCE2 [Bentley 2005]

2.2 Agile Methods

The objective of this chapter is to discuss Agile concepts related to this research. Subsection 2.2.1 discusses Agile values and 2.2.2 discusses Lean principles in general. Subsection 2.2.3 briefly presents Scrum and finally Subsection 2.2.4 discusses Agile and Scrum in GSD with some examples.

2.2.1 Agile Values

One response to an ever-complicated environment is the rise of Agile methods [Abrahamsson et al. 2002]. These methods or approaches value the following [AgileAlliance 2009]:

- *Individuals and interactions over processes and tools*
- *Working software over comprehensive documentation*
- *Customer collaboration over contract negotiation*
- *Responding to change over following a plan*

The explanations have been given e.g. by Ambler [Ambler 2010] and by Ryder [Ryder 2005]. According to Ambler [Ambler 2010] it is important to understand that the concepts on the left hand side are not alternatives to the concepts on the right hand side. One should preference things on the left hand side, but concepts on the right hand side are also important in Agile development.

Individuals and interactions over processes and tools: Teams of people build software systems, and to do that they need to work together effectively. People in Agile development are developers, testers, managers, customers, other stakeholders etc. The most important factors that you need to consider are the people and how they work together; if you do not get that right the best tools and processes will not be of any use. Tools and processes are important, but they are not as important as teams of skillful people.

Working software over comprehensive documentation: The most important goal of software development is to create software, not documents. In any case, documentation has its place, written properly it is a valuable guide for people’s understanding of how and why a system is built and how to work with the system.

Customer collaboration over contract negotiation: Successful developers work closely with their customers, invest the effort to discover what their customers need, and educate their customers along the way. Working together with your customers is hard, but it is important to understand what they really need. Having a contract with your customers is important, having an understanding of everyone’s rights and responsibilities may form the foundation of that contract, but a contract is not a substitute for communication.

Responding to change over following a plan: People change their priorities for a variety of reasons. Change is a reality of software development, a reality that your software process must reflect. It is important to have a project plan but there must be room to change it as your situation changes, otherwise your plan quickly becomes irrelevant.

2.2.2 Lean Principles

Agile methods are partly based on Lean Software Development [Poppendieck M. and T. 2003] which is in turn based on Lean manufacturing thinking [Ohno 1988]. Lean manufacturing thinking was developed at Toyota in Japan. According to Leffingwell [Leffingwell 2007, pages 63-65] the Lean approach from the view point of manufacturing decreases cycle times and manufacturing costs in four different ways:

- *Reduced work in process and inventory*
- *Reduced cycle times*
- *Cross-training and cell-based manufacturing*
- *Continuous process improvement*

Reduced work in process and inventory: Years of manufacturing cost analysis showed that almost all inventory and related work to inventory were fundamentally bad. Inventory tended to become obsolete before it was shipped or needed to be reworked before it could be used. Also, owning inventory meant maintaining it and substantial personnel and administrative costs were needed to do the counting, moving, storing, and otherwise maintaining it prior to production. Over time, manufacturers came to understand that all of this work is not value-added activity; there is no value to the customer in the manufacturer owning a piece of inventory that it does not need or to which it has no access.

Reduced cycle times: By building smaller lots in a factory tuned for rapid flow, cycle times are dramatically reduced. The ability to fill customer orders more quickly is a hallmark of a Lean enterprise.

Cross-training and cell-based manufacturing: In Lean manufacturing, workers are cross-trained to be able to play multiple roles. Cross-training is required by Lean because leaner factories sometimes have bottlenecks that must be solved with additional cross-trained workers.

Continuous process improvement: In Lean, there is no such thing as “too short a cycle time” or “too much reduction in waste and overhead”. Therefore, Lean principles drive teams to a need for continuous and self-directed process improvements. Because bottlenecks and inefficiencies are continually exposed by leaner thinking, continuous improvement is always on the forefront of Lean and Six Sigma initiatives. Table 4 shows how Lean manufacturing principles relate to Agile Software Development.

Table 4. Lean manufacturing principles relate to Agile Software Development [Leffingwell 2007]

Lean Manufacturing Principles	Relate to Agile Software Development
Reduced work in process and inventory	Reduced investment in elaborated requirement, documented designs. Reduced process overhead, compliance checks, audits etc.
Reduced cycle times	Build all software in much smaller lots (chunks, stories, use cases).
Cross-training and cell-based manufacturing	Increase cross-training with pair programming and shared code assets. Have developers write tests as part of their code. Move entire teams to test and test automation. Collocate team commits to delivering the iteration. Entire team commits to delivering the iteration.
Continuous process improvement	Continuous reflection and adaptation. Self-organizing, self-managing software development teams.

Another view to implement Lean principles in software development is presented in [Poppendieck M. and T. 2006]. Table 5 briefly describes the seven principles of Lean software development according to [Poppendieck M. and T. 2006].

Table 5. Seven Principles of Lean Software Development [Poppendieck M. and T. 2006]

Principles of Lean Software Development	Description
Eliminate Waste	Focus on removing all nonvalue adding wastes such as extra features which are not needed by customers or partially made software or delays in the software development process.
Build Quality In	Goal is to build quality into the code from the start, not test it later. This goal is possible to reach using test-driven approach.
Create Knowledge	Software development is a knowledge-creating process. It is important to gather new knowledge through projects and provide this new information to the whole organization.
Defer Commitment	Schedule irreversible decisions for the last responsible moments and maintain options at the points of software where change is likely to occur.
Deliver Fast	Rapid delivery, high quality, and low costs are the main goals. Make small batches with fast cycle time to have fewer things in the works. Limit amount of work according to capacity of your team.
Respect People	Respect people in your and your partners' team and empower the team.

Optimize the Whole	Focus on the entire value stream and deliver a complete product to customers. Measure UP which is meant to find a higher-level measurement that will drive the right results for the lower level metrics.

2.2.3 Scrum

A popular Agile project management method which is based on Agile values and Lean principles, is Scrum. The Scrum approach is presented in Figure 4. It is an approach applying the ideas of industrial process control theory to a development process resulting in an approach that reintroduces the ideas of flexibility, adaptability and productivity. In the development phase the product is developed in sprints. Sprints are iterative cycles where the functionality is developed or enhanced to produce new increments. Each sprint includes the traditional phases of software development: requirements, analysis, design, evolution and delivery phases. [Abrahamsson et al. 2002]

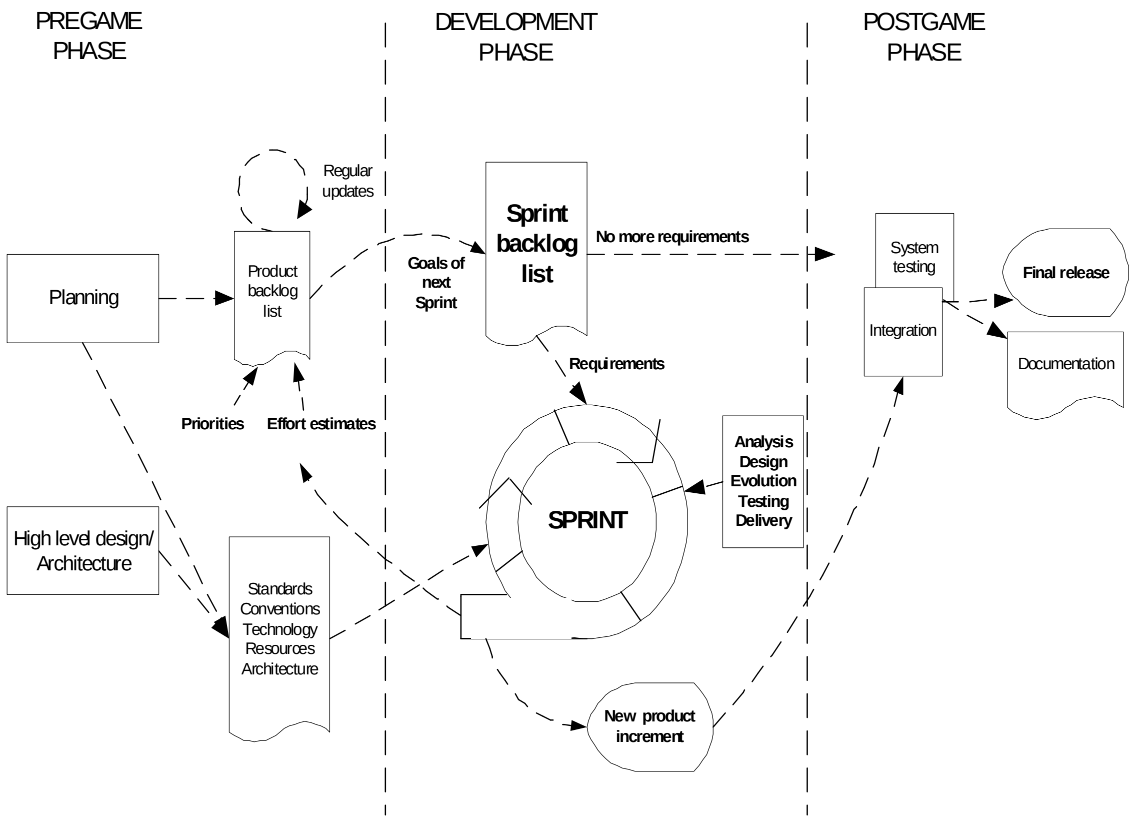


Figure 4. Scrum overview [Abrahamsson et al. 2002]

2.2.4 Agile and Scrum in GSD

At first glance, Agile methods seem to be only suitable for small teams operating in a local environment. Agile methods were usually applied just for local development but nowadays their potential for supporting more effective global development environments has been detected. The usage of Agile methods in a distributed development environment has been studied and reported e.g. in [Ramesh et al. 2006], [Sutherland et al. 2007] and [Farmer 2004].

Ramesh et al. [2006] report case studies conducted in three companies. Companies combined Agile methods for distributed development environments. Ramesh et al. [2006] list the challenges in Agile distributed development that relate to the aspects of communication, control and trust. Then Ramesh et al. [2006] report successful practices observed in three organizations that address these challenges. As a conclusion, the success of distributed Agile development relies on the ability to blend the characteristics of both Agile and distributed environments. Sutherland et al. [2007] report a case study of a distributed Scrum project where they analyze and recommend best practices for distributed Agile teams. They report integrated Scrum as a solution to distributed development in which a Scrum team was formed by the distributed project stakeholders. Farmer [2004] reports experiences when developing software in a large, distributed team. The team worked according to slightly modified Extreme Programming (XP) practices. Farmer [2004] states that the major factors that contributed to their success were that the team consisted of some of the top people in the company, that it was permitted to find its own way and that management support was available.

The usage of Agile methods in large-scale level development has been discussed e.g. in [Leffingwell 2007] and [Leffingwell and Aalto 2009]. Leffinwell [2007] describes practices which have been derived from experiences in applying agile practices in large companies. They include seven scalability agile team practices which can be used as building blocks to create an enterprise level agile development process including agile project management.

2.3 Application Lifecycle Management

The objective of this section is to discuss Application Lifecycle Management (ALM) issues related to this research. ALM is related to some of the most important GSD patterns to implement a GSD environment. Subsection 2.3.1 discusses ALM and project management in general. Subsection 2.3.2 briefly presents the elements of ALM. Subsection 2.3.3 discusses ALM in GSD with some examples. Mr. Jukka Kääriäinen is mainly responsible for the ALM research briefly described in the following Subsections.

2.3.1 ALM and Project Management

Schwaber [2006] states that the three pillars of ALM are process automation, traceability, and reporting. ALM is quite a new term even though tool vendors have used it to indicate tool suites or their approaches that provide support for the various phases of the software development lifecycle. In many scientific articles, the term ALM has been treated only cursory, e.g. in [Shroff et al. 2005], [Dearle 2007] and [Heindl et al. 2007]. However, issues that relate to ALM such as traceability, tool integration, communication and reporting have already been studied for a long time by the scientific community. In ALM, these issues are considered in the context of the whole development lifecycle.

Some kind of solutions for ALM, manual or tool-supported, exist in every company developing software even though toolsets that are marketed specifically as ALM suites were introduced just a few years ago. The role of ALM is to act as a coordination and product information management discipline. The purpose of ALM is to provide integrated tools and practices that support project cooperation and communication through a project's lifecycle (Figure 5) [Kääriäinen 2011]. It breaks the barriers between development activities with collaboration and fluent information flow. According to [Doyle 2007], a proper ALM system provides strong support for project management. For management, ALM provides an objective means to monitor project activities and generate real-time reports from project data.

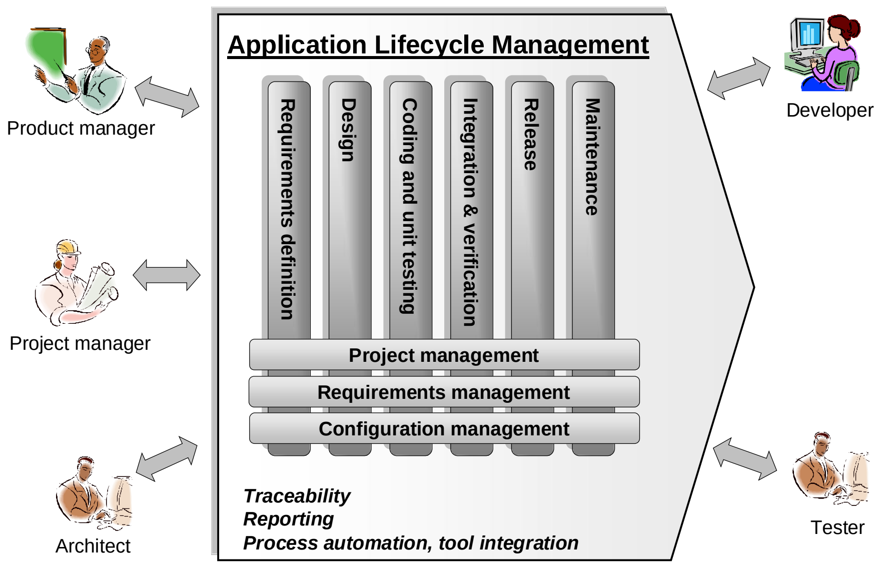


Figure 5. Application Lifecycle Management facilitates project cooperation and communication [Kääriäinen 2011]

The roots of ALM tools are in Configuration Management (CM) and Integrated Development Environments (IDEs) [Weatherall 2007]. CM solutions are usually the foundations of ALM infrastructures providing storage, versioning and traceability between all lifecycle artifacts [Schwaber 2005].

Shaw [2007] argues that with traditional toolsets, traceability and reporting across disciplines is extremely difficult. The same data is often duplicated in various applications (e.g. separate databases for requirements, source code, and documents) that complicate the traceability and maintenance of data. According to [Doyle and Lloyd 2007], having a central repository makes it easier to produce a relevant set of related metrics that span the various development phases of the lifecycle.

There are tool vendors whose target is to provide support for the lifecycle activities of software development. Examples of commercial ALM solution providers are Borland Software Corporation, IBM, Microsoft, Serena Software, and Polarion Software. Basically, ALM solutions provide development environments integrating various lifecycle applications such as requirements management, project management, configuration management and design and development tools. ALM suites provide features such as traceability, reporting, process support, tool integration possibilities and information visibility. Pesola et al. [2008] state that one limitation with existing ALM solutions 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 might need to replace a number of development tools because a certain ALM solution does not integrate with all the existing tools. This may require lots of effort and money. Another approach that aims to be more vendor-independent is called an “Application integration framework”. These are used as platforms to integrate several applications needed during the software development lifecycle. Examples of this kind of framework are Eclipse [Eclipse 2010] and the Application Lifecycle Environment (ALF) [Eclipse 2010] projects.

Although tools are important for efficient ALM, Schwaber [2006] states that ALM does not necessarily require tools. Traditionally, lifecycle activities have been handled by partly using manual operations and solutions. For example, distribution of documents between different functions can be controlled by using a paper-based approval process. However, these activities can be made more efficient through tool integration with process automation.

2.3.2 Application Lifecycle Management Framework

The need to better understand the elements of ALM in order to support the development of ALM in an organization has emerged [Kääriäinen 2011]. In these studies Mr. Jukka Kääriäinen developed a framework consisting of six principal elements that characterize ALM. This framework can be used for documenting and analyzing organizations’ ALM solutions and thus support the practical development of ALM in an organization. This framework was constructed and successfully applied in an automation company to support its ALM documentation and improvement efforts. The current framework version contains six elements as presented in Figure 6.

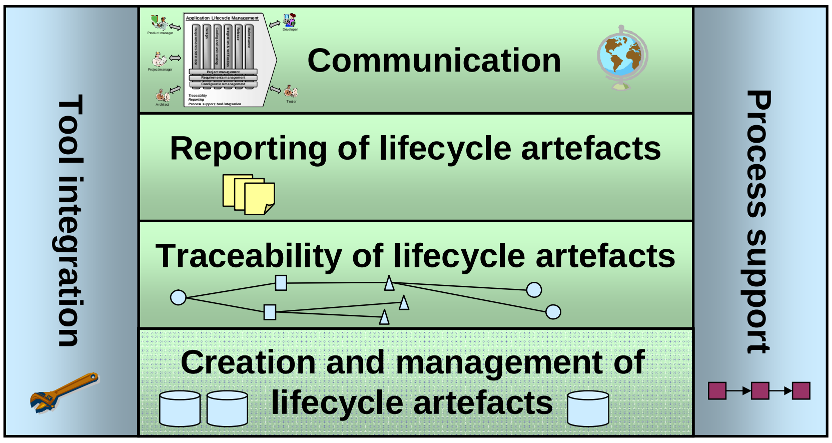

Tool integration
Communication
Reporting of lifecycle artefacts
Traceability of lifecycle artefacts
Creation and management of lifecycle artefacts
Process support

Figure 6. Principal elements of Application Lifecycle Management [Kääriäinen 2011]

“Creation and management of lifecycle artifacts” is the foundation for ALM. The product information collected and managed by this element is needed, for instance, for traceability and reporting activities. “Traceability of lifecycle artifacts” provides a means to identify and maintain relationships between managed lifecycle artifacts and therefore, facilitates reporting, change impact analysis and information visibility throughout the development lifecycle. “Reporting of lifecycle artifacts” utilizes managed lifecycle artifacts and traceability information to generate needed reports from the lifecycle product information to support software development and management. “Communication” provides communication tools (e.g. chat) as well as channels for distributing information about product lifecycle artifacts, links and reports and thus facilitates product information visibility for the whole software project. “Process support” and “Tool integration” are the elements that are used to configure the ALM solution to support software development procedures, to guide the user through development activities and to facilitate a productive development environment by enabling the user to easily launch tools and transfer information between different tools and databases.

2.3.3 ALM in GSD

Modern approaches for product development need to take into account the global development environment and the product’s whole development lifecycle must be covered, from the initial definition right up to maintenance. Such a holistic viewpoint means the efficient deployment of lifecycle management: Product Lifecycle Management (PLM) and from a software development point of view, Application Lifecycle Management (ALM). Furthermore, product development is often distributed over multiple sites and customers might also operate globally. Globalization forces companies to find ways to overcome geographical barriers and modern information technology offers an excellent means to achieve this goal. Recently, ALM solutions have been introduced as the means to tackle the challenges of global development (see e.g. [Doyle and Lloyd 2007] and [Kääriäinen 2011]).

The promise of ALM to support distributed development has been treated in [Doyle and Lloyd 2007]. Doyle and Lloyd [2007] argue that geographically distributed development introduces communication and coordination challenges for ALM. They continue that one possibility is to use a central secure repository with acceptable network performance and implemented work procedures in order to provide real-time information about changes and task assignments to support working in a distributed development environment.

2.4 Patterns

The objective of this chapter is to discuss patterns from the viewpoint of this work. Subsection 2.4.1 discusses the pattern concept in general. Subsection 2.4.2 briefly presents formats of patterns. Finally, Subsection 2.4.3 discusses pattern languages.

2.4.1 The Idea of a Pattern

The idea of a pattern was first introduced by Alexander et al. [1977] in the context of building architecture. According to this definition, a pattern is a context-dependent construct which includes a problem and a reproducible solution to that problem [Alexander et al. 1977]. Patterns are trusted because each one has been used several times in real projects. Patterns are not one-off solutions or good ideas that might or might not work. Patterns are discovered, not created. [Elssamdisy 2008].

The idea of a pattern was applied later in software development processes, leading to the most famous pattern approach in software development, design patterns [Gamma et al. 1995]. These patterns can be used to develop more flexible and reusable software products and applications. Another view for patterns is to use them to describe best practices of human organizations as described in organizational patterns ([Coplien 1996]; [Coplien and Harrison 2005]). Process patterns can also be used to describe best practices as they have been used in this thesis. The process of finding patterns has often been referred to as pattern mining [Appleton 1997].

2.4.2 Formats of Patterns

There are different formats of patterns such as the Alexandrian format [Alexander et al. 1997], the Gang of Four format [Gamma et al. 1995], the Coplien format [Coplien 1996], the Appleton format [Appleton 1997] etc. All these pattern formats have a certain structure and their content is described by informal text. The Coplien format [Coplien 1996] and the Appleton [Appleton 1997] format are almost the same and have been used as a model for GSD patterns format. The Appleton format has the elements described in Table 6.

Table 6. Elements of a pattern [Appleton 1997]

Field of a pattern	Description
Name	Name must be meaningful.
Problem	Problem describes the goals and objectives for the pattern within the given context and forces.
Context	Context describes preconditions under which a problem and its solution seem to recur and for which a solution is desirable
Forces	Forces describe relevant forces and constraints and how they interact or conflict with one another and with goals of the pattern.
Solution	Solution describes how to realize the desired outcome.
Examples	Examples describe the use and applicability of the pattern.
Resulting context	Resulting context describes consequences and side-effects of applying the pattern and other problems and patterns that may arise from the new context. It describes the post conditions and side-effects of the pattern.
Rational	Rational describes why the pattern resolves its forces in a particular way.
Related patterns	Related patterns describe the relationships between this and other patterns.
Known uses	Known uses describe known occurrences of the pattern.

2.4.3 Pattern Languages

The concept of a pattern language has been defined in slightly different ways. Coplien [1996] defines a pattern language as a collection of patterns which generate a system. A pattern in isolation solves an isolated design problem, but a pattern language builds a system. It is through pattern languages that patterns achieve their fullest power. On the other hand, Appleton [1997] defines a pattern language to be a collection of patterns which forms a vocabulary for understanding and communicating ideas. Such a collection may be skillfully woven together into a cohesive "whole" that reveals the inherent structures and relationships of its constituent parts toward fulfilling a shared objective. This is close to what Alexander calls a pattern language. If a pattern is a recurring solution to a problem in a context given by some forces, then a pattern language is a collection of such solutions which, at every level of scale, work together to resolve a complex problem into an orderly solution according to a pre-defined goal [Appleton 1997].

Different ways of organizing a pattern language can be divided into problem-centered and solution-centered approaches [Eloranta et al. 2009]. In the problem-centered approach, the domain can be used as the structure of the language. The solution-centered approaches for organizing a pattern language are based on the relationships between the solutions rather than on structuring the problem domain.

In this study, a pattern language is organized according to a process phase structure which is one example of the problem-centered approach. A process phase structure is chosen to make it easier to find a suitable pattern or patterns for a user’s specific problems. The pattern language of this thesis will be presented in Chapter 4.

2.4.4 Evaluation of a Pattern Language

Usually a pattern describes both best practice solutions and discussion when it is suitable to use with some known uses in a certain context. Since patterns are not invented but rather discovered from existing systems, these known uses serve as a proof that the pattern is a working solution [Kirchner and Völter 2007].

Patterns are typically improved and reviewed in dedicated conferences organized by the pattern community. The pattern community has organized several PLoP (Pattern Languages of Programs) conferences in order to elaborate patterns to ensure their correctness and usefulness. [Kirchner and Völter 2007].

In this work, GSD Pattern Language has been reviewed and improved by persons from case projects. After that GSD Pattern Language was also reviewed by GSD specialists and managers from different GSD companies and a research center (VTT). These persons also gave valuable feedback for improving GSD Pattern Language. However, GSD Pattern Language is still evolving and has not been used in many projects. For this reason, I have also evaluated GSD Pattern Language using a new evaluation method for patterns which will be described in Chapter 5.

3 PATTERN MINING IN GSD

The objective of this chapter is to describe the pattern mining method which was used to collect patterns in this thesis. Section 3.1 describes the pattern mining method at a general level. Section 3.2 describes how this process was applied in this thesis and finally, Section 3.3 describes the industrial context of the pattern mining.

3.1 Pattern Mining Method

Pattern mining can be viewed as exploratory case study research. This kind of case study research is meant to produce new theoretical ideas and proposals by analyzing certain practices. The improved ideas or proposals can be tested in the same or in the following case studies. Exploratory case study research is the first step to create generalizations and theory ([Ryan et al. 1992 p. 115] and [Yin 2003]).

The pattern mining method developed and implemented in this work is based on Yin’s steps of case study research, i.e., preparing data collection, collecting evidence, analyzing evidence and reporting [Yin 2003]. Figure 7 describes the overview of the developed pattern mining method.

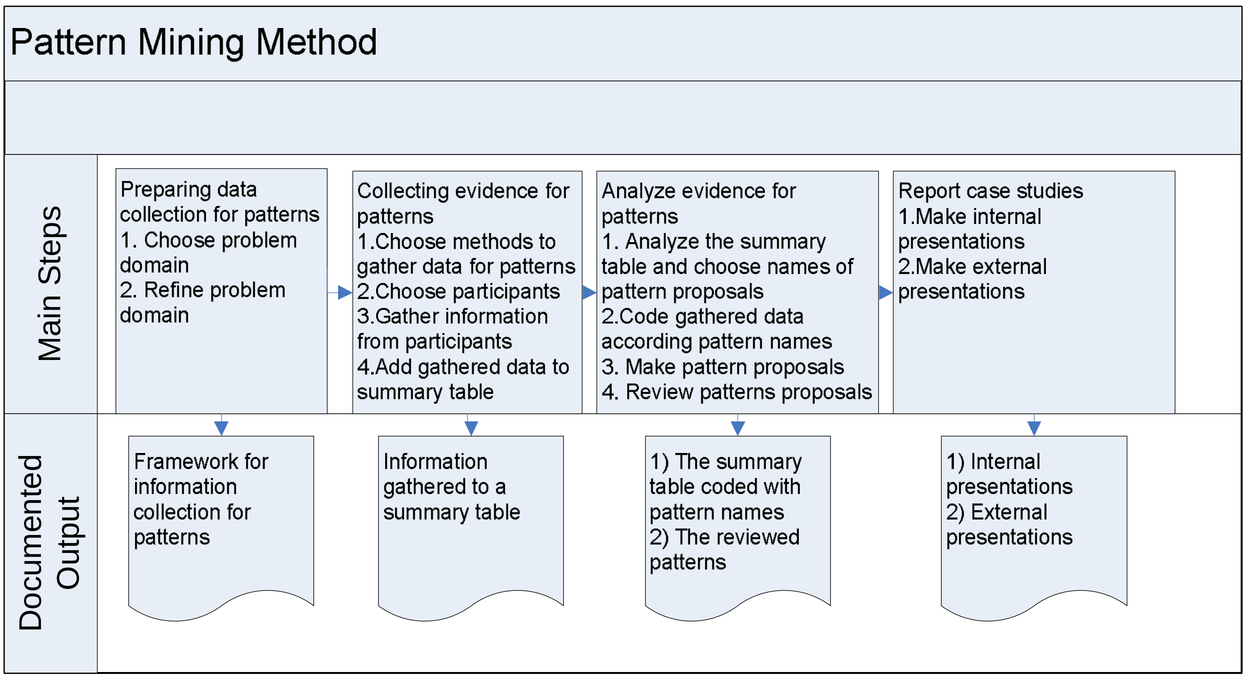

Pattern Mining Method
Main Steps
Documented Output
Preparing data collection for patterns
1. Choose problem domain
2. Refine problem domain
Collecting evidence for patterns
1.Choose methods to gather data for patterns
2.Choose participants
3.Gather information from participants
4.Add gathered data to summary table
Analyze evidence for patterns
1. Analyze the summary table and choose names of pattern proposals
2.Code gathered data according pattern names
3. Make pattern proposals
4. Review patterns proposals
Report case studies
1.Make internal presentations
2.Make external presentations
Framework for information collection for patterns
Information gathered to a summary table
1) The summary table coded with pattern names
2) The reviewed patterns
1) Internal presentations
2) External presentations

Figure 7. Pattern mining method

The first step in the pattern mining method is preparing a data collection which includes the choosing of a problem domain for patterns and refining it.

Each separate case study is started by choosing a certain problem domain for patterns (e.g. Distributed Scrum) in which good practices and problems are collected. The problem domain is further refined into a refined problem domain which can be e.g. different phases (e.g. Sprint planning, Sprint review) or artifacts (e.g. Product backlog, Sprint backlog) depending on which details need to be researched.

The refined problem domain can also be researched from different views (e.g. people, process, tools). For each view there is a questions template consisting of three open-formed questions for data collection. Question derivation framework is depicted in Figure 8.

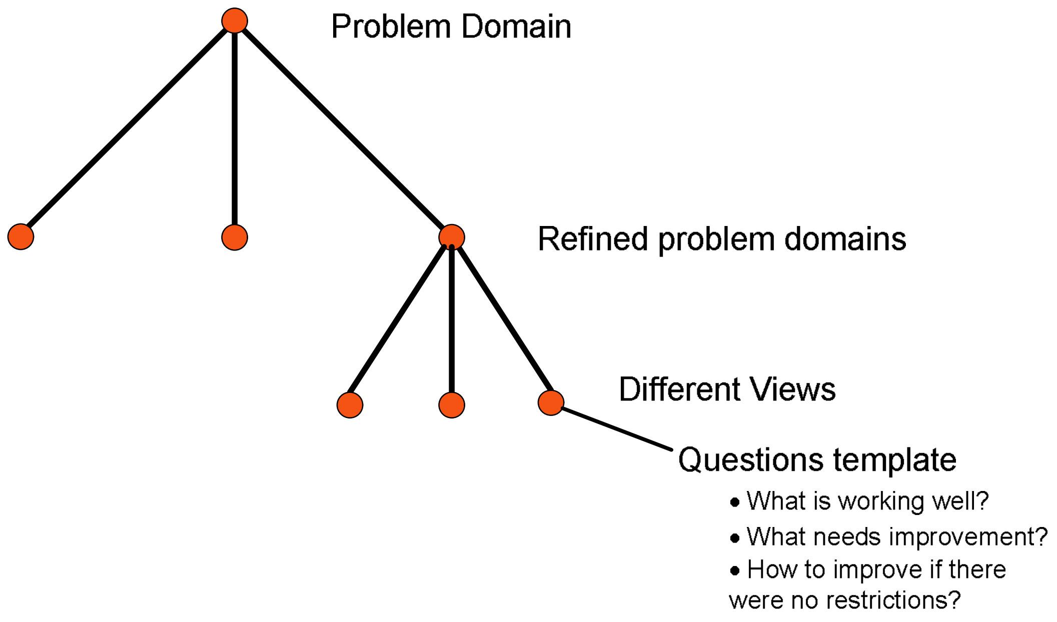


Figure 8. Question derivation framework

The second step in the pattern mining method is collecting evidence for patterns. This is implemented by choosing how to get answers to the questions in the framework and who will participate in this case. Answers can be gathered e.g. by the use of questionnaires and interviews for some key participating team members.

The questionnaire consists of questions about background information of participants and questions in the created framework. The questionnaires are sent to participants of a case project. Answers can be gathered e.g. by e-mails or query tools.

After questionnaires, there are interviews of key persons in this case. Key persons are interviewed in order to get more detailed information about the case. Interviews are organized either face to face or implemented by net meeting and conference phone techniques. In the interview meetings, the framework is used as a checklist to make sure that all related areas are covered. As a result of this step, the questionnaires and interviews produce raw data for analysis according to the used framework. The raw data is gathered in a summary table which is based on the used framework as presented in Table 7.

Table 7. The summary table with pattern hints

Viewpoints	**Pattern hints**
Viewpoint one (e.g. people)	- Pattern hint one - Pattern hint two - Pattern hint three
…	…

The third step in the pattern mining method is to analyze evidence. In this method it means making an analysis for data from the summary table by using coding methods [Miles and Huberman 1994]. In the first analysis of the summary table (Table 7), the names of pattern proposals are chosen. In the second analysis the names of pattern proposals are used to mark pattern hints in the summary table for each pattern proposal. Pattern hints in the summary table could be related to pattern proposals such as problems, forces, solution etc. Table 8 shows one example of the summary tables with pattern proposals. Pattern hints are marked with pattern proposals. The whole pattern mining framework is depicted in Figure 9.

Table 8. The summary table with pattern hints and pattern proposals

Viewpoints	**Pattern hints**	**Pattern proposals**
Viewpoint one (e.g. people)	- Pattern hint one - Pattern hint two - Pattern hint three	Pattern A name Not Applicable Pattern B name
…	…	…

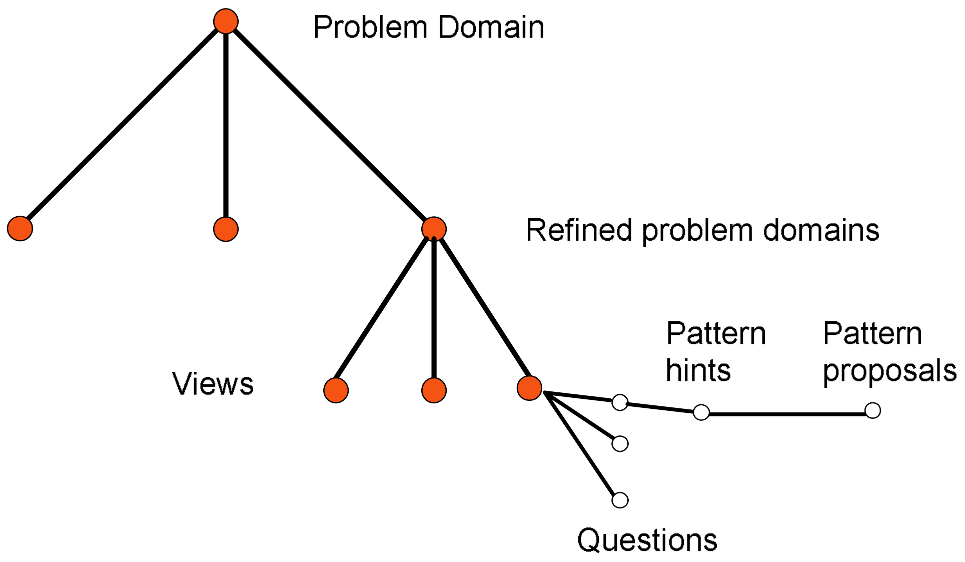


Figure 9. The pattern mining framework

After making the summary table, pattern proposals are formed in a pattern format. The contents of pattern proposals can be based on both pattern mining workshops and related literature. The names of patterns proposals are important because they will be part of the vocabulary which is used when related processes or actions are implemented. To ensure that the patterns are feasible, the proposed process patterns are reviewed by discussing them with the project's key persons. As a result of this step, the patterns are created for the problem domain. These patterns can be further reviewed and improved in other workshops. It should be noted that new patterns can also be introduced in these workshops.

The fourth step in the pattern mining method is reporting case studies. Reporting can be internal in a company or external such as meetings with other companies or presentations in international conferences.

3.2 Applying Pattern Mining Method to GSD Patterns

Pattern mining method which is presented in Section 3.1 is used to find GSD patterns in this thesis. The research conducted in Paper II is an example about applying the pattern mining process to GSD patterns. This method is also used in [Välimäki and Koskimies 2006] and Paper II to mine patterns for GSD patterns. [Välimäki and Koskimies 2006] and Papers II and III are based on three case studies which will be described in more detail in Section 3.3.

As a result of the first step the problem domain was chosen to be Distributed Scrum. This problem domain was further refined to include Scrum practices and artifacts which were Product Backlog, Sprint Planning, Sprint Backlog, Daily Scrum, Sprint Review and Scrum of Scrums.

The details were researched from different views: people, process, tools. For each view there were three open-formed questions for data collection as presented in Section 3.1. A framework for data collection for Distributed Scrum is depicted in Figure 10.

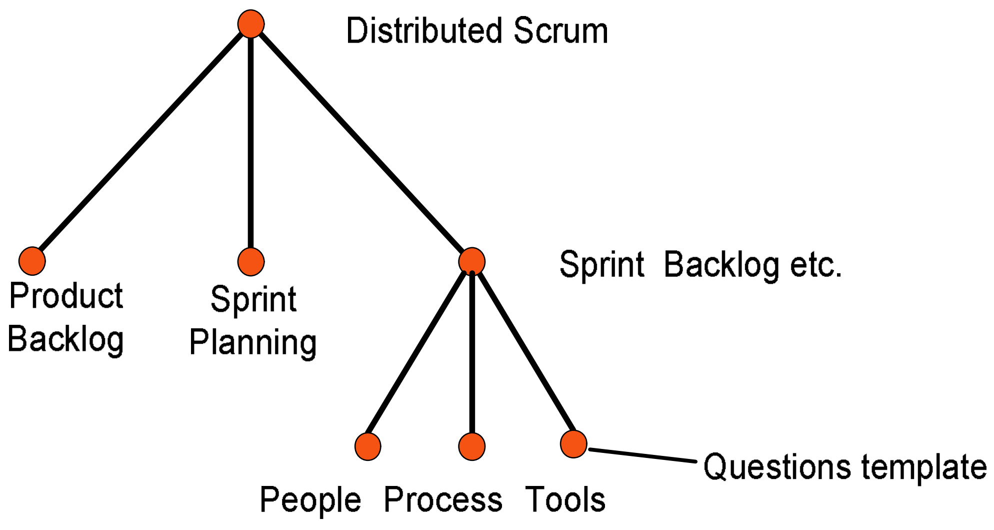


Figure 10. A framework for data collection for Distributed Scrum

The second step in the pattern mining method was collecting evidence for patterns. This was implemented with questionnaires and interviews as described in Section 3.1. As a result of this step, the questionnaires and interviews produced raw data for analysis according to the used framework. The raw data was gathered in a summary table.

The third step in the pattern mining method was to analyze evidence. In the first analysis of summary table, the names of pattern proposals were chosen. In the second analysis the names of pattern proposals were used to mark pattern hints in the summary table for each pattern proposals. The names of the pattern proposals can be changed during the second analysis. Table 9 shows summary table of gathered data for Product Backlog after the third step.

Table 9. The summary table for Product Backlog with potential pattern proposals

Viewpoints	Pattern hints	Pattern proposals
People	In distributed development the replication of Scrum roles through sites is essential to ensure fully functional Scrum teams at every site.	*Organize needed Scrum roles at each sites*
	Knowledge transfer is especially important in a distributed environment. More detailed information about domain and product backlog items is needed to ensure understanding in a global environment.	*Knowledge transfer*
	In a global environment, team members have to rely on more formal communication and IT means	*Establish efficient communication methods*
Process	Product backlog process should allow collecting and processing ideas. Ideas, if accepted for development, will be used for creating backlog items.	*Establish Application Lifecycle Management tool*
Tools	Common secure and integrated repositories for all project data.	*Establish Application Lifecycle Management tool*
	Process support should enable the process and form tailoring, e.g., new item attributes, tailored status models, adding hierarchy, etc.	*Establish Application Lifecycle Management tool*
	Infrastructure and efficient as well as reliable network connections are a must when using a central project database in a distributed environment	*Establish a fast and reliable infrastructure*

After making the summary table, pattern proposals were formed in a pattern format. The contents of pattern proposals were based on both the pattern mining workshops and related literature. To ensure that the patterns were feasible, the proposed process patterns were reviewed by discussing them with the project’s key persons. As a result of this step, the patterns were created for the chosen problem domain.

The described Pattern mining method was applied in three case studies which will be described in more detail in Section 3.3. The patterns from these case studies were further reviewed and improved in other interviews and workshops. These interviews and workshops were organized with GSD specialists and managers from different GSD companies and a research centre (VTT). It should be noted that new patterns were also introduced in these workshops. The current version of GSD Pattern Language will be presented in Chapter 4.

The fourth step in the pattern mining method is reporting case studies. In this research it meant making an internal and external presentation of the results of this research.. The internal presentation was implemented by publishing results in the internal document management tool. The external presentations were implemented by [Välimäki and Koskimies 2006], Paper I and II, and finally Paper V which also includes the research conducted in Papers III and VI.

3.3 Industrial Context and Case Studies

The target company operates in the automation industry, being a global supplier of technology and services for the power generation, oil and gas, pulp and paper industries. The company has engineering, production, procurement, service, sales and other operations in over 100 units in approximately 40 countries and employs about 3 800 professionals. The company is the world's leading supplier of automation solutions for the pulp and paper industry. It designs, develops and delivers automation and information management application networks and systems, intelligent field control solutions, and support and maintenance services.

This work is divided into three case studies in which each case study has included certain projects and persons. They have been used for pattern mining and elaboration from certain areas of project management in GSD.

GSD Project management in plan-driven projects
The purpose of the first case study was to explore the work of project managers in plan-driven projects in a GSD environment. This case study was implemented in February, March and April, 2006. Different GSD projects were carried out between the main company and subcontractors. Four sites of the target companies were in Finland and two subcontractors in Finland and one in Canada. In this case model 1 and 2 were used. The models are described in Figure 2 on page 10. The interviewed nine project managers and two other project members were especially familiar with the challenges in distributed development and gave valuable information in questionnaires and through discussions in personal interviews. In this case, the query was answered by eleven participants who were also briefly interviewed. These persons were in nine projects. The results of this case are presented in [Välimäki and Koskimies 2006] and the main result was a preliminary group of project management patterns in plan-driven projects in a GSD environment.

Some examples of answers from questionnaires related to *In the beginning of a project.*

- “The most important issue in GSD is communication. This makes it sure that we are making right things. This means e.g. more specific and frequent control by project managers.”

- “In the beginning of a project it is important to have a common meeting face to face in which the whole project will be described as exactly as possible. There is need for clear goal setting and division of responsibilities and tasks for all participants in different sites. The project rules need to be agreed (tools, reporting, configuration management, documentation and other guidelines, frequent meetings, reporting etc.)”

Some examples of answers from questionnaires related to *In the middle of the project*

- “There is a need for common process and tools, e.g. project work desk, project reporting. Communication tools must be as good as possible e.g. mail, telephone, net meeting tools. Unfortunately, changes need always bureaucratic change management which is usually much slower with persons in other sites than the main site.”

- "It is important to have regular meetings with subcontractors in the main site. Communication need to be implemented with e.g. task tracking, net meeting, instant messaging, mail, conference phones. Tool to help planning and controlling of tasks is also needed"

Some examples of answers from questionnaires related to *If there are no constraints, what kind of best practices would you choose?*

- "A good, detailed process description with roles. What are the roles used in a project in different sites and what are the tasks of each role?"

- "Tools which support the whole lifecycle of a software project. Tools for configuration management, build management, task management, communication and project management which are integrated with each other very well."

GSD in the work of product managers

The purpose of the second case was to explore the work of product managers in the target company. This case study was implemented in December 2006 and January 2007. In the future, the work of product managers will globalize even more, covering several countries. Therefore, the challenges of the global development environment were studied and suggested solutions to overcome the challenges were defined. Product development in the target company is organized according to product lines. This research focuses on two product lines that consist of several products that are partly developed in different sites. As it is no longer competitive to develop multiple products one at a time, the case company has adopted a product platform approach. Therefore, the product is based on a product platform where the customer-specific features are configured. The product line evolves when new versions of the products are produced containing whole new features or improvements to the existing features. The management of features and requirements is complex since even one product version can contain dozens of features that are further divided into hundreds of requirements.

In this case a query was answered by 12 product managers. Based on the query, four experienced product managers were selected for further interviews. The results of this case are presented in Paper I. The main results were the process of mining requirements management process patterns and a preliminary group of product management patterns in a GSD environment. The used query form is presented in Appendix A.

Some examples of answers from questionnaires related to *Collecting and processing product ideas.*

- "Everybody needs to have access for a tool to create idea. We need more ideas and strict filtering of these ideas. We need a systematic process for idea management."
- "We need a better process and a common tool which can be easily used inside the company."

Some examples of answers from questionnaires related to *Collecting and processing business and customer requirements.*

- "All requirements from all stakeholders need to be gathered to the same common tool as for gathering ideas"
- "Different phases need to be integrated with each other in the same tool and there is a need for bidirectional links between different artifacts"

Some examples of answers from questionnaires related to *Feature creation*

- "Analyzing of ideas are made by product managers but their need help form architects and other specialists."
- "Product managers make feature proposals after analyzing ideas and requirements"

GSD project management in Agile projects

The purpose of the third case was to explore the work of team members in Agile GSD projects. This case study was implemented in August, September and October, 2007. The geographic distribution as well as increasing complexity and efficiency demands forced the target company to seek more integrated solutions and practices to coordinate distinct project phases and to provide centralized project database applications for all project-related data.

This case focused on two product lines consisting of several products that were partly developed in different sites. The case was further focused on two teams each having 3 to 6 projects running in parallel. The projects were geographically distributed over 2 or 3 sites in two countries. Each project had typically less than 10 team members. In this case the model 3 and 4 was used. The models are described in Figure 2 on page 10. In this case the query was answered by eight participants and two of them were also interviewed. The results of this case are presented in Paper II. The main results were the process of mining agile project management process patterns and a preliminary group of agile project management patterns in a GSD environment. The used query form is presented in Appendix B.

Some examples of answers from questionnaires related to *Product Backlog.*

- "Product owner role is very important and he/she must have time to make and update product backlog."
- "There is a need for product owners or persons who can clarify specifications in every site. Also scrum masters for every site is needed."

Some examples of answers from questionnaires related to *Sprint Planning.*

- "According Scrum team members must be skilful persons which can plan a Sprint after a product owner has presented the contents of Product Backlog items."
- "Currently the Sprint planning meetings require constant travelling. "
- "Set up more advanced communication software to provide remote all-together meetings"

Some examples of answers from questionnaires related to *Sprint and Daily Scrum.*

- “Daily Scrum increases information flow between different team members. Problems can be solved or at least presented quickly in a short meeting by discussions between team members.
- “If distributed Daily scrum is practiced then adequate IT means are needed to facilitate communication.”

The research of Paper V is also about GSD patterns and based on research conducted in [Välimäki and Koskimies 2006], Paper I, and Paper II. In addition, patterns have been further evaluated in pattern review meetings and workshops. Pattern review meetings have also been organized with other companies to improve contents of GSD Pattern Language. GSD Pattern Language is also evaluated as will be presented in Chapter 5.

4 GSD PATTERN LANGUAGE

The objective of this chapter is to introduce the GSD Pattern Language. Section 4.1 gives a short introduction to the GSD patterns. Section 4.2 describes the used pattern format. Section 4.3 presents the organization of the GSD patterns based on PRINCE2 which is a project management method. This kind of organization makes it easier to understand patterns and to facilitate the usage of patterns. Finally, Section 4.4 describes mapping between ALM elements and some GSD patterns. The ALM framework describes some important aspects of GSD and mapping of ALM elements with related GSD patterns shows how an ALM element could be implemented with GSD patterns.

4.1 GSD Patterns

The purpose of the GSD Pattern Language is to enhance performance of project management work through improved global software project management practices. The GSD Pattern Language includes 18 process patterns which have been found to be important in the area of project management in GSD. Of course, there are process patterns which are suitable both in collocated and distributed project management, but in this work the main interest has been with process patterns in GSD.

Whether a plan driven or an Agile process model is used, the problems related to the nature of GSD have to be dealt with. Rather than developing a totally new GSD process that addresses these problems, a more appropriate approach is to try to come up with solutions to specific problems, and present these solutions in such a way that they can be easily integrated with existing processes. An obvious advantage of this approach is that a company need not adopt a new process model, but merely tune the existing process for GSD. In the third case study (GSD project management in Agile projects), Scrum is applied which has affected some patterns. However, the patterns do not directly refer to Scrum concepts but they are described in more general terms and can be applied even if Scrum is not used.

An attractive way to document proven solutions to specific development process problems is to use organizational patterns [Coplien and Harrison 2005] or process patterns [Ambler 1998]. A collection of such solutions can be further organized into process pattern languages [Alexander et al. 1977]. A process pattern language need not cover the entire process, but it can concentrate on a certain viewpoint of the software development process. In this work the viewpoint is GSD: I derive a pattern language for project management in GSD. The solutions in these patterns have been mined from the practices that have been found to work well in a large company operating in the field of process automation.

In general, patterns represent knowledge that is validated by previous experience. However, if patterns are mined from a limited environment, as in my case, this argument does not hold. In this work I have evaluated the resulting patterns by using a scenario-based technique introduced in Section 5.1.

The identified GSD patterns are presented in Table 10. The first column contains the name of the pattern, the second describes the problem the pattern is supposed to solve, and the last column gives a brief summary of the solution of the pattern.

Table 10. GSD patterns for project management

ID-Name	**Problem(s)**	**Solution outline**
GSD1-GSD Strategy	How to organize GSD in a company?	List the reasons and motivation to start GSD-based development in a company. Make a short and long term plan about GSD. Find out the competence of different sites and make a SWOT and risk analysis for GSD strategy. Also measure the real costs of GSD.
GSD2-Fuzzy Front End	How to gather demands (i.e. ideas, needs, requirements etc.) from external and internal customers and how to form plans and change requests from these needs for GSD development?	The needs of different customers will be gathered to a global database. It is also important to have the possibility for global access regardless of time and place as well as have the possibility to use a discussion forum inside the tool. Product managers will go through gathered needs and make decisions about them with e.g. architects. A new feature or requirement will be made if it is accepted in a decision meeting. Product managers will make a Road Map and a Business plan for a product including many features. These features will be realized in development projects.
GSD3-Collocated Kick-Off	What is the goal of a GSD project and who are the members of a project? How to build trust between team members?	Arrange kick-off meeting for all relevant members. Present common goal and motivation of this project and present release plan made by *Divide and Conquer with Iterations* pattern. Present responsibilities of each site and team members, if possible. Briefly introduce tools and repositories which are chosen for a project. Briefly present common processes in a project. Also train cultural issues for team members. Organize leisure activities for teams to improve team spirit.
GSD4-Divide and Conquer with Iterations	How to make a project plan which is manageable in a GSD project?	Plan several iterations to describe the project plan. Develop new application architecture and module structure during first iterations, if needed. Explore the biggest risks (e.g. new technologies) in the beginning of a project. The length of iteration can be e.g. 2-4 weeks to improve control and visibility. Main site can have 4 weeks iteration and other sites 2 weeks to improve visibility.
GSD5-Choose Roles in Sites	How to know who to contact in different sites with your questions?	A project manager will have negotiations with site managers or other supervisors about team members before final decisions. Also needed roles will be formed in every site (e.g. Site project Manager, Architect, IT Support, Quality assurance etc.) The main site person is in a leading position and the persons from other sites will help to take care of the issues, tasks and responsibilities in their sites. Publish the whole project organization with roles for every site to improve communication. One person can have many roles in a project.

GSD6-Communication Tools	How to choose communication methods and tools in GSD?	Have reliable and common communication methods and tools in every site. Use different tools at the same time as net meeting to show information and project data, conference phones to have good sound and chat tool to discuss in written form if there are problems to understand e.g. English language used in other sites.
GSD7- Common Repositories and Tools	How to share different artifacts between sites efficiently?	Provide a common Application Lifecycle Management (ALM) tools for all project artifacts (documents, source code, bugs, guidelines etc.) ALM provides almost real-time traceability, reporting, visualization and access to needed information etc. for all users in different sites. It can be implemented as a single tool or it can be a group of different tools which has been integrated with each other. ALM tools can include means to support operation according to the organisation's processes and development methods (state models, process templates, workflows). Use different levels (team, project, and program) reports to improve visibility of status of projects.
GSD8-Work Allocation	How work is divided between sites?	Find out what the GSD Strategy is in your company and check competence information of persons in each site with help of site managers. Make *Architectural Work Allocation* and/or make *Phase- Based Work Allocation* and/or make *Feature Based Work Allocation* and/or other allocation according to some other criteria. Make a decision about division of work between sites according to a company's GSD Strategy and the above analysis.
GSD9-Architectural Work Allocation	How work is divided between sites with architectural criteria?	Check architectural analysis of your product and plan which site will be responsible to maintain and increase knowledge in some architectural area. Architectural area can also be a whole subsystem or part of a subsystem.
GSD10-Phase-Based Work Allocation	How work is divided between sites with phase-based criteria?	Check how phase- based work allocation will be made. Also check which site is possibly responsible to maintain and increase knowledge in some phase-based area e.g. testing or requirements engineering in a certain product area.
GSD11-Feature-Based Work Allocation	How work is divided between sites with feature-based criteria?	Check the *GSD Strategy* how feature- based work allocation strategy has been described. Form a group of members from different sites to realize the features, if needed.
GSD12-Common Processes	How to implement an efficient process in a GSD project?	Choose common upper level processes and allow local processes if they do not cause problems with upper level processes.
GSD13-Iteration Planning	How to choose which features will be implemented in a GSD project?	Project manager will present prioritized features and other tasks. Team members will participate in a planning meeting either personally or by communication tools. The team members will estimate amount of work for features and tasks. If needed, more detailed discussion can be arranged in sites with participants' mother language. In the end of a planning meeting, a list of selected features and tasks are created and is visible by the common repository.

GSD14-Multi-Level Daily Meetings	How to share project status information between team members in each site? Lack of trust and long feedback loops.	Organize many daily meetings and organize another daily or weekly meeting between project managers from different sites to exchange information about the results of daily meetings. With foreigners, written logs can be one solution to ensure that communication messages are understood correctly in every site.
GSD15-Iteration Review	How to check status of a GSD project and give and get feedback frequently?	Check the project status by a demo and present results to all relevant members and stakeholders from different sites. Gather comments and change requests for further measures for both product and process. Make frequent deliveries to improve visibility of the status of the product.
GSD16-Organize Knowledge Transfer	How to transfer a huge amount of knowledge for team members in a GSD project?	Make sure that there is a product knowledge repository available for team members. Train the product and get members also to use it, if possible. Specification with use cases will be presented in the iteration planning meetings or separate meetings. Also earlier customer documentation and demo will be presented in some cases. The team members' network will be utilized by trying to find solutions for problems. Use frequent or longer visits to enhance knowledge transfer and be sure that there are good communication channels between team members.
GSD17-Manage Competence	How to know what the competence of each team member is in a GSD project?	Create a competence database for gathering information of members' competence levels at different sites. At least site manager and/or project manager knows the competence of team members. Define the areas of competence you want to monitor. Define competence levels and criteria for them. Ask site managers and or project managers to gather information about their team members.
GSD18-Notice Cultural Differences	How to notice cultural differences and increase cultural knowledge in team members in a GSD project?	Raise the awareness of your team nations' culture for team members. Use site visits, ambassadors and liaisons, if possible. Notice cultural differences when you are applying different patterns such as *GSD Strategy, Work Allocation, Common Processes,* Communication *Tools* and *Common Repositories and Tools* etc. Allow local approaches in processes, tools, meeting methods etc. to decrease problems with cultural differences, if they do not disturb common processes etc.

4.2 Pattern Format

The used pattern format is based on Appleton's pattern format, which is described in more detail in Subsection 2.4.2. The aim was to use a minimum format to make it easier to understand the idea of each pattern. The used fields are *name, problem, initial context, forces, solution, and resulting context* as presented in Table 8. These patterns have been used in internal projects according to the industrial context in Section 3.3, but information about examples or known uses have not been gathered, although there were also reviews with other companies about the goodness of patterns. These patterns have not been presented at any PLOP conferences but they are evaluated as described in Section 5.

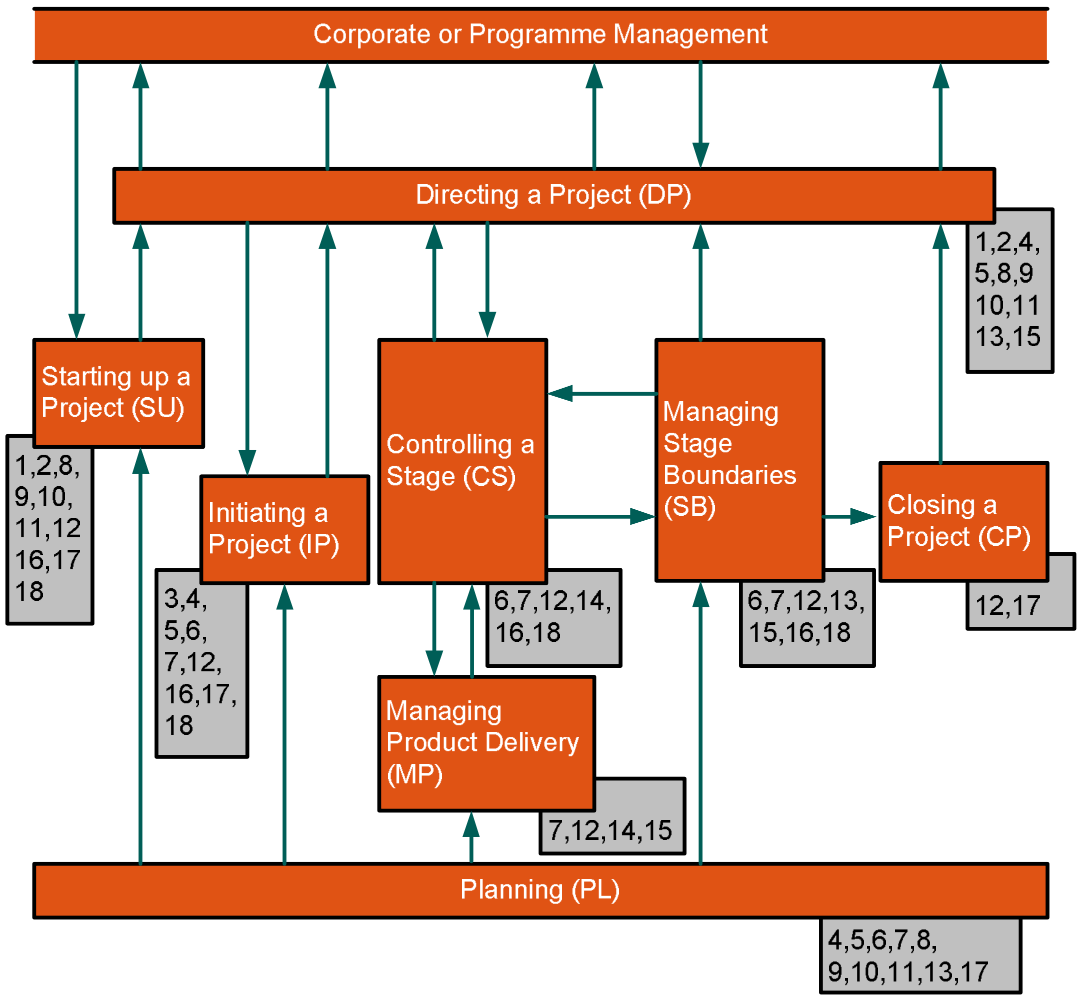
Other differences with the GSD pattern format and Appleton's pattern format are the fields *roles, rational* and *related patterns*. The field *roles* describe the roles implementing the pattern. The content of *rational* is combined in the field *resulting context*, and *related patterns* information is described in the organization of the pattern language in Section 4.3.

Table 11. An example of a GSD pattern

Name:	***GSD4- Divide and Conquer with Iterations***
Problem:	How to make a project plan which is manageable in a GSD project?
Initial context:	In the beginning of a project only the main features are known.
Roles:	Project manager.
Forces:	• A big project plan is difficult to manage in a GSD project. • Difficult to know the whole contents and the work estimations of a project in the start of a project. • Visibility of project status is poor in GSD. • Possible new application architecture, technologies etc. are unknown.
Solution:	Project manager will split a project plan into several iterations. Implement the following actions: • Plan several iterations to describe the project plan because iterations are easier to control and it is easier to make changes to a plan. • Develop new application architecture and module structure in the main site during first iterations, if needed. • Explore the biggest risks (e.g. new technologies) in the beginning of a project. • The length of iteration can be e.g. 2-4 weeks to improve control and visibility. Main site can have 4 weeks iteration and other sites 2 weeks to improve visibility.
Resulting Context:	Iterations improve the visibility of a project and motivation of team members in a GSD project. Iterations also make it easier to control a project when you split the whole project into many manageable parts. Iterations also provide feedback and the possibility to learn from earlier iterations. However, administration work is increased with several iterations.

4.3 Pattern Language Organization with PRINCE2

In this section the pattern language organization is described based on the PRINCE2 project management method. The PRINCE2 process overview was presented in Figure 3 on page 15. PRINCE2 is a process-driven project management method with a stage-based process structure which makes it easy to use as a basis for organizing process patterns. Here I organize the pattern language by attaching the patterns to the main processes of PRINCE2.



1 GSD Strategy	12 Common Processes
2 Fuzzy Front End	13 Iteration Planning
3 Collocated Kick-Off	14 Multi-Level Daily Meetings
4 Divide and Conquer with Iterations	15 Iteration Review
5 Choose Roles in Sites	16 Organize Knowledge Transfer
6 Communication Tools	17 Manage Competence
7 Common Repositories and Tools	18 Notice Cultural Differences
8 Work Allocation	
9 Architectural Work Allocation	
10 Phase- Based Work Allocation	
11 Feature- Based Work Allocation	

Figure 11. GSD Pattern Language Organization with PRINCE2

The analysis of GSD Pattern Language Organization with PRINCE2 is made by making an analysis for each pattern, one at a time.

GSD Strategy pattern can be used with Main Processes of PRINCE2 (hereafter called Main Processes) Directing a Project (DP) and Starting up a Project (SU). If this is the first GSD project it is important to list the reasons and motivation to start GSD-based development in a company and in this project. One way to make a vision of development strategy is to make a short and long term plan about GSD in a company. For that, it is needed to find out the competence of different sites and make a SWOT and risk analysis for GSD strategy. One important issue is also to measure the real costs of GSD in the short and long term. In the next projects, the results of *GSD Strategy* can be used to motivate and clarify reasons to start a new GSD project.

Fuzzy Front End pattern can also be used with Main Processes Directing a Project (DP) and Starting up a Project (SU). Before a new project is started, the needs of different customers can be gathered to a global database according to the *Common Repositories and Tools* pattern. It is also important that everyone in a company has the possibility for global access regardless of time and place as well as the possibility to use a discussion forum inside the tool. After that, product managers will go through gathered demands and make decisions about them with e.g. architects. A new feature or requirement will be made from a new demand if it is accepted in a decision meeting by product managers and architects. This new change request will be implemented in a new or existing project if it is accepted in a decision meeting by product and project managers. New features are usually difficult to implement in the middle of a current projects and that is why product managers will gather and group features with requirements together. With these new features, product managers will update current Road Maps or Business plans or make new ones. After that, the chosen features will be realized in new starting development projects which will be made according to the decisions related to Business plans. Related decision meetings can be implemented either as face to face meetings or distributed meetings with *Communication Tools.*

Collocated Kick-Off pattern can be used with Main Process Initiating a Project (IP). In the beginning of a project there are many open issues and often the team members do not know each other. Also the reason why this project will be done is not clear. A project manager will arrange a kick-off meeting for all relevant members. They will also present common goals and why this project is needed. Not only work allocation between GSD sites will be presented, but also who the members of a project are and what the responsibility of each participant with this project is. Also development tools such as *Common Repositories and Tools* and processes as *Common Processes* will be presented briefly and the schedule of the next meetings will be agreed with participants. In the beginning or in the end of meetings some team games will be arranged to increase team spirit among team members.

Divide and Conquer with Iterations pattern can be used with Main Processes Directing a Project (DP), Initiating a Project (IP), and Planning (PL). One big project plan is a risk in GSD, because the visibility of the project is usually poor. To improve visibility, a project manager can split a project into several iterations. An iteration plan can be planned according to the PL process and the plan can be presented to GSD participants according to the IP process. The iteration plan also helps to direct a project in the DP process. Finally, the iteration plan will also be updated according to the PL process.

Choose Roles in Sites pattern can be used with Main Processes Directing a Project (DP), Initiating a Project (IP), and Planning (PL). A project manager will have negotiations with site managers or other supervisors about project participants. Also, the needed roles in different sites will be decided. These project participants will be approved in a DP meeting. When a project is started the whole project organization will be published in an IP meeting. Possible changes in a project organization will be decided in PL meetings.

Communication Tools pattern can be used with Main Processes Initiating a Project (IP), Controlling a Stage (CS), Managing Stage Boundaries (SB) and Planning (PL). Communication is a key challenge in GSD and it is important that a project has as good communication tools as possible. A project manager organizes communication methods and tools and presents them briefly in an IP meeting. These tools are used in SB, CS and PL meetings when team members in other sites are participating these meetings.

Common Repositories and Tools pattern can be used with Main Processes Initiating a Project (IP), Controlling a Stage (CS), Managing Product Delivery (MP), Managing Stage Boundaries (SB) and Planning (PL). In the beginning of a project it is important to make a decision how e.g. documents, source codes and process guidelines are stored and distributed efficiently for different sites. This decision is made in the first PL meetings. A project manager presents these tools briefly in an IP meeting for project participants. These tools are used in IP, CS, MP, SB and PL meetings or other project tasks when e.g. project artifacts, reports or process guidelines are needed.

Work Allocation pattern can be used with Main Processes Directing a Project (DP), Starting up a Project (SU), and Planning (PL). Work Allocation pattern consists of three other patterns which are *Architectural Work Allocation, Phase-Based Work Allocation* and *Feature-Based Work Allocation.* All of these are related in how to allocate and divide work between each site, but the criteria of allocation is different. In the beginning of a project it is not clear how to divide responsibilities between sites. The main site manager is a key person to divide responsibilities in sites according to a company's GSD Strategy in DP meetings. A project manager will apply GSD Strategy in his/her project in SU meetings, will check GSD Competence information of persons in each site with the help of site managers and make a decision as to what kind of work allocation is used in his project. After that, a project manager will make a decision about more detailed division of work.

Common Processes pattern can be used with almost all Main Processes. In the beginning of a project there are often different processes in every site. A project manager chooses common upper level processes for the project to have e.g. same phases, concepts and document templates to improve communication. It may be useful to allow local processes if they do not cause problems with upper level processes because local processes might be efficient to use in a certain site. Processes can be tuned at the end of an iteration to make a project more effective, if a project has a meeting on how to improve or change current processes.

Iteration Planning pattern can be used with Main Processes Directing a Project (DP), Managing Stage Boundaries (SB), and Planning (PL). In the beginning of a project there are many features to be implemented. Different persons in a DP meeting have different views about which are the most important features. Also project participants do not know what kinds of features are needed for a project. The project manager is the key person to tell what the most important features to be implemented are. The project manager will present prioritized features and others in the beginning of each iteration planning meeting. Project participants will participate in planning meetings either personally or by Communication Tools. Project participants from different sites will estimate tasks which are needed to deliver selected features after discussions about their contents. If needed, more detailed discussion can be arranged in sites in the participants' mother tongue, if the native language of participants is different. In the end of planning meetings, the selected features are created and are visible by GSD Common Repositories and Tools for all project participants.

Multi-Level Daily Meetings pattern can be used with Main Processes Controlling a Stage (CS), and Managing Product Delivery (MP). After each iteration planning meeting, project groups in different sites have started to implement iteration and there is a need to communicate between groups in different sites. If everybody in the project does not speak the same language then separate daily meetings in every site can be a more efficient way of working. Another reason to have separate daily meetings can be the size of teams. If the whole team is bigger than seven participants, separate meetings can be more efficient. Other reasons to have common or separate meetings with project participants is related to companies' relationships and team setups as presented in Figure 2 on page 10. If there are separate daily meetings, then organize another daily or weekly meeting between project managers from different sites to change information about the results of daily meetings. Add more levels of meetings if needed. With foreigners, written logs can be one solution (e.g. chat logs or common documents) to ensure that communication messages are understood correctly at every site. In the meetings the status of planned tasks are checked and discussed as well as how to organize product delivery of developed products.

Iteration Review pattern can be used with PRINCE2 Main Processes Directing a Project (DP), Managing Product Delivery (MP), and Managing Stage Boundaries (SB). In a project it is difficult to know what its status is. During the project, participants have done a lot of work and there is a need to check status and give feedback about the results and products made. The project manager will organize a review meeting in which team members will show the current status of their work. It is difficult to get everyone to participate in the meeting, but all the relevant participants should attend. If all participants are not able to attend the meeting it can also be arranged by using different communication tools. In this way information can be shared with participants communicating over a remote connection. The results of the iteration will be presented and comments will be gathered from participants which will be managed later by the change management process. This pattern will bring visibility to work done and provides feedback of what has gone well and what could be done better. Also team members focus on their tasks because they have to present their work in the meetings and decisions related to product delivery of developed products can be made there.

Organize Knowledge Transfer pattern can be used with Main Processes Starting up a Project (SU), Initiating a Project (IP), Controlling a Stage (CS), and Managing Stage Boundaries (SB). In the beginning of a project it's difficult to transfer a huge amount of knowledge to often new project participants. In the beginning of a project it is important to choose common repositories and tools to manage not only information about a project but also information about domain knowledge. Training of the product to be developed as well as using the product is needed for all project participants. Specification will be presented in the Iteration Planning meeting or separate meetings before development work. Also earlier customer documentation and demos will be presented in some cases. Knowledge transfer is possible through documents and tools, but especially through people. *Choose Roles in Sites* networks will be utilized by trying to find solutions to problems. Also frequent or longer visits to enhance knowledge transfer and to make sure that there are good communication channels between team members.

Motivation to make knowledge transfer is very important in the main site because knowledge transfer puts a load on especially that site from which knowledge is to be transferred. Also, if work moves along with knowledge transfer to the other site it will reduce motivation for knowledge transfer work.

Manage Competence pattern can be used with Main Processes Starting up a Project (SU), Initiating a Project (IP), Planning (PL), and Closing Project (CP). In the beginning of a project it is difficult to know what the competence of each team member is. Site managers can create competence databases for gathering information of members' competence levels at different sites. If databases are not possible to create, then at least site managers and/or project managers know the competence of team members. Site managers or project managers define levels for competence and criteria for each competence level and define the areas of competence you want to monitor. Site managers or project managers can gather information about their team members. Site managers or project managers go through competence information with each team member and make a training plan for obtaining the needed competence levels. In the end of a project, competence updates are needed on a competence database or another document of competence.

Notice Cultural Differences pattern can be used with Main Processes Starting up a Project (SU), Initiating a Project (IP), Controlling a Stage (CS), and Managing Stage Boundaries (SB). In the beginning of a project, a nation's cultural differences and/or differences between different companies are not understood. There may be a lot of difficulties and inefficiency if cultures are very different from each other and the cultural differences are not understood. A project manager should raise the awareness of his/her team nation's culture for team members and use site visits and liaisons, if possible. A project manager can also notice cultural differences in every communication situation and try to increase awareness in all team members in different sites. With these actions team spirit, trust between team members and efficiency of teams can be improved because differences between nations are easier to understand. Cultural differences are also important and must be taken into consideration when *GSD Strategy* and *Work Allocation* are applied. Cultural differences can also affect the use of *Common Processes, Communication Tools* and *Common Repositories and Tools***.** A project manager can also use Iteration retrospective to increase cultural awareness and to improve the process if cultural differences seem to add barriers to communication, coordination and control. *Common Repositories and Tools* will improve the visibility of a project. A project manager can allow local approaches in processes, tools, meeting methods etc. to decrease problems with cultural differences if they do not disturb common processes etc. Of course cultural issues will affect more if your project teams are forming one virtual team than separate teams, as presented as two examples of team setups in Figure 2 on page 10.

In Table 9 the GSD Pattern Language organization with PRINCE2 is presented in a matrix form according to the above analysis. The eight PRINCE2 major processes are rows and the GSD patterns are columns. An x in the matrix means that the column pattern is related to the row process.

Table 12. Relations between GSD patterns and PRINCE2 major processes

	1 GSD Strategy	2 Fuzzy Front End	3 Collocated Kick-Off	4 Divide and Conquer with Iterations	5 Choose Roles in Sites	6 Communication Tools	7 Common Repositories and Tools	8 Work Allocation	9 Architectural Work Allocation	10 Phase- Based Work Allocation	11 Feature- Based Work Allocation	12 Common Processes	13 Iteration Planning	14 Multi-Level Daily Meetings	15 Iteration Review	16 Organize Knowledge Transfer	17 Manage Competence	18 Notice Cultural Differences	Number of x-marks
DP	x	x		x	x			x	x	x	x		x		x				10
SU	x	x						x	x	x	x	x				x	x	x	10
IP			x	x	x	x	x					x				x	x	x	9
CS						x	x					x		x		x		x	6
MP							x					x		x	x				4
SB						x	x					x	x		x	x		x	7
PL				x	x	x	x	x	x	x	x		x				x		10
CP												x					x		2

From Table 9 can be seen that GSD6 (Communication Tools), GSD7 (Common Repositories and Tools), GSD12 (Common Processes), GSD16 (Organize Knowledge Transfer*)*, GSD17 (Manage Competence) and GSD18 (Notice Cultural Differences) have a strong relationship with PRINCE2 processes, because these patterns have four or more x marks with PRINCE2 processes. GSD6, GSD7 and GSD12 are important when implementing an efficient global software environment. GSD6 provides tools for communication between team members. GSD7 provides almost real time access to the needed information from each team member. GSD12 provides common artifacts and methods at least for high level processes between team members to improve communication and understanding between team members. GSD16 is also important because often employees in other sites have no earlier experience about domain knowledge at all, especially if they are starting the co-operation with the main site. GSD17 is needed in order to know what the competence of each employee is, especially for the planning phase of a project. GSD18 is also a fundamental practice to achieve efficient co-operation with employees from different countries.

From Table 9 can also be seen that processes of Directing a Project (DP), Starting up a Project (SU), Initiating a Project (IP) and Planning (PL) all have nine or ten marks. This is because GSD patterns are mainly related with starting, initiating, high level directing, and planning of a GSD project. Also DP and PL are processes which continue from the start up to the closing of a project. The next highest numbers of marks have processes of Controlling a Stage (CS) and Managing Stage Boundaries (SB) which are related to iterations and are in an important role in iterative and Agile project management methods. Finally, the process Managing Product Delivery (MP) has only four marks and Closing a Project (CP) is used only twice in my analysis. The MP process has not yet had such an important role in GSD patterns in my cases because the product delivery has mainly been out of scope in my studies. MP is an important process and is one important area for further research in future. Also CP is an area to be analysed in more detail in future studies.

4.4 ALM Framework and Related GSD Patterns

ALM elements and framework are presented in Figure 6 on page 23. ALM support for GSD can be analysed by using GSD Pattern Language for Project Management which is described in Section 4.1. This GSD Pattern Language includes 18 process patterns which have been found to be important in the area of project management in GSD. If these GSD patterns are compared to ALM elements in Subsection 2.3.2, it can be noticed that some patterns were related to ALM elements as presented in Table 13 [Kääriäinen et al. 2009]. Patterns named "Communication Tools" and "Common Repositories and Tools" relate to ALM elements. Furthermore, "Common Processes" relates to an ALM element called "Process support". This shows that ALM databases could be used as solutions to meet the problems indicated in GSD patterns. Further analysis about ALM in the context of GSD is presented in Section 5.4.

Table 13. Mapping between ALM elements and related GSD patterns [Kääriäinen et al. 2009]

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

5 EVALUATION OF GSD PATTERN LANGUAGE

The objective of this chapter is to introduce the evaluation of the GSD Pattern Language. Section 5.1 discusses the coverage of GSD Pattern Language in terms of GSD challenges described in Subsection 2.1.4. Section 5.2 presents the scenario-based assessment method Q-PAM (Quality-oriented Process Assessment Method) as a whole, and then explains the individual steps in more detail. Q-PAM is used as an evaluation method in this work. Section 5.3 describes how Q-PAM was applied in the evaluation of the GSD Pattern Language. Finally, Section 5.4 presents a summary of the Q-PAM evaluation.

5.1 Relations between GSD Challenges and GSD Pattern Language

In this section the relationship between GSD challenges from literature and GSD Pattern Language are analyzed. The analysis of GSD challenges and GSD Pattern Language is created by making an analysis for each GSD challenge at a time. The aim is to analyze the coverage of GSD Pattern Language in terms of GSD challenges. Subsections 5.1.1, 5.1.2, and 5.1.3 discuss the relationship between GSD Language and GSD challenges related to temporal, geographical and socio-cultural distance, respectively. GSD challenges were presented in Table 3 on page 13. Finally, Subsection 5.1.4 summarizes this discussion.

5.1.1 GSD Challenges Related to Temporal Distances

Reduced opportunities for synchronous communication (Challenge 1). Patterns GSD6 (Communication Tools) and GSD7 (Common Repositories and Tools) give solutions by providing means for asynchronous communication, increased availability awareness by shared calendars and shared documentation which is visible to all team members. If temporal distance of projects is significant, such as eight hours, then GSD Pattern Language need updates about enlarging overlap hours by working longer ([Lane and Ågerfalk 2009], [Battin et al. 2001] and [Carmel and Tija 2005]) or to have an individual liaison role who enlarges their own working hours [Carmel and Tija 2005].

Typically increased coordination costs (2). Patterns GSD6 (Communication Tools) and GSD7 (Common Repositories and Tools) give solutions as above. Again, if temporal distance is significant, such as eight hours then Lane and Ågerfalk [2009] propose using overlap hours for remote communication and not to have local meetings during these hours.

Management of project artifacts may be subject to delays (3). Patterns GSD7 (Common Repositories and Tools) and GSD12 (Common Processes) give solutions by providing e.g. shared documentation and common processes and artifacts. Also Evaristo et al. [2004] says to enforce process and artifact standards to maintain consistency between project artifacts. Lane and Ågerfalk [2009] also remind to notice a risk of e.g. a two-week delay if there was a need for a second review meeting of an artifact for some reason.

5.1.2 GSD Challenges Related to Geographical Distances

Face-to-face meetings difficult (4). GSD6 (*Communication Tools*), GSD7 (*Common Repositories and Tools*) and GSD12 (Common Processes) give solutions by providing e.g. shared documentation and groupware tools to manage reviews and artifacts with communication tools. Also GSD3 (Collocated Kick-Off) and GSD5 (Choose Roles in Sites) give solutions by getting team members to meet face-to-face and by giving information about who is responsible for certain information.

Reduced informal contact can lead to reduced trust (5). GSD3 (Collocated Kick-Off), GSD13 (Iteration Planning), GSD14 (Multi-Level Daily Meetings) and GSD15 (Iteration Reviews) give solutions for this challenge. GSD3 (Collocated Kick-Off) can be used to organize e.g. face-to-face meetings and to understand what each team member's responsibilities are. GSD14 provides daily meetings which increase trust with each other. Also Lane and Ågerfalk [2009] suggest using daily meetings to increase trust. GSD13 and GSD14 provide meetings in order to know what each members have done and to have the opportunity to discuss with other team members.

A lack of critical task awareness (6). This challenge can be solved with the same GSD patterns as in the earlier challenge.

Difficult to convey vision and strategy (7). GSD1 (GSD Strategy), GSD3 (Collocated Kick-Off), GSD8 (Work Allocation), GSD9 (Architectural Work Allocation), GSD10 (Phase Based Work Allocation), GSD11 (Feature Based Work Allocation) and GSD13 (Iteration Planning) can be used to solve this challenge. GSD1 is used not only to make a strategy but also to communicate the strategy to team members in order to understand and make it clear what each responsibility is. GSD13 helps to update implementation of strategy in the beginning of each iteration. GSD8, GSD9, GSD10, and GSD11 are solutions for work allocation decisions and use of them is related to implementation of strategy.

Perceived threat from training low-cost "rivals" (8). GSD1 (GSD Strategy) can give a solution to this challenge. A clear and communicated strategy helps to understand the decisions of the management of a company.

5.1.3 GSD Challenges Related to Socio-Cultural Distances

Cultural misunderstandings (9). GSD18 (Notice Cultural Differences) suggest raising the awareness of project teams cultural understanding in national culture. Also company-related cultural differences need to be updated in this pattern. Remote leaders and other roles are needed as presented in GSD5 (Choose Roles in Sites) to increase co-operation. Communication and meetings with partners can increase cultural understanding as presented in patterns GSD3 (Collocated Kick-Off), GSD13 (Iteration Planning), GSD14 (Multi-Level Daily Meetings) and

GSD15 (Iteration Review). Lane and Ågerfalk [2009] also suggest using a temporary co-location of remote team members in the beginning of a project.

Inconsistent work practices can impinge on effective coordination (10). GSD12 (Common Processes) suggest decreasing the differences in process, at least at a high level. Also GSD7 (Common Repositories and Tools) with common repositories and tools make coordination more effective.

Reduced cooperation arising from misunderstandings (11). GSD16 (Organize Knowledge Transfer) at least increases common understanding about all knowledge related to a project. Also meetings, which are organized according to patterns GSD3 (*Collocated Kick-Off*), GSD13 (Iteration Planning), GSD14 (Multi-Level Daily Meetings), and GSD15 (Iteration Review), increase understanding project knowledge between team members in different sites.

Different perception of authority can undermine morale (12). GSD5 (Choose Roles in Sites) and GSD18 (Notice Cultural Differences) will help with this challenge as well as *Managers must adapt to local regulations challenges (13)*.

5.1.4 Relations Between GSD Challenges and GSD Pattern Language

This subsection presents the results of the analysis made in Subsections 5.1.1, 5.1.2, and 5.1.3. In Table 14 the analysis is presented in a matrix form. The thirteen GSD challenges are rows and the patterns of the GSD Pattern Language are columns. An x in the matrix means that a GSD Pattern provides solutions to a GSD challenge, but usually a GSD pattern does not usually solve all the issues of each challenge.

Table 14. Relations between GSD challenges and GSD Pattern Language

	1 Reduced synchronous communication	2 Typically increased coordination costs	3 Management of project artifacts	4 Face-to-face meetings difficult	5 Reduced trust	6 A lack of critical task awareness	7 Difficult to convey strategy	8 Low-cost “rivals”	9 Cultural misunderstandings	10 Inconsistent Work practices	11 Reduced cooperation	12 Different perceptions of authority	13 Managers must adapt to local regulations
GSD1							x	x					
GSD2													
GSD3				x	x	x	x		x		x		
GSD4													
GSD5				x					x			x	x
GSD6	x	x		x									
GSD7	x	x	x	x						x			
GSD8							x						
GSD9							x						
GSD10							x						
GSD11							x						
GSD12			x	x						x			
GSD13					x	x	x		x		x		
GSD14					x	x			x		x		
GSD15					x	x			x		x		
GSD16											x		
GSD17													
GSD18									x				

From the results presented in Table 14, can be concluded that the coverage of GSD Pattern Language is reasonable from the viewpoint of GSD challenges. However, GSD challenges concentrate mainly on phases which are in use when a project is ongoing. E.g. GSD2 (Fuzzy Front End), GSD4 (Divide and Conquer with Iterations) and GSD17 (Manage Competences) are mainly related to the starting of a project which is not in the scope of GSD challenges.

5.2 Q-PAM

In this section the Q-PAM evaluation method is introduced. Subsection 5.2.1 gives a short motivation for Q-PAM. Subsection 5.2.2 gives an overview of the method. Subsection 5.2.3 describes creating a quality profile for specifying the aims of the evaluation. Subsection 5.2.4 describes how to construct scenarios which are used to perform the analysis discussed in Subsection 5.2.4. The Q-PAM method has been introduced in Paper IV. The method has also been used to evaluate a knowledge-sharing pattern language [Vesiluoma 2009].

5.2.1 Motivation

How can a collection of process practices be evaluated? In many ways a process model can be compared to system architecture. The implementation of a process model is a concrete process instance, in the same way as the implementation of system architecture is the actual system. The system architecture determines the major quality attributes of the system, and the process model determines the major quality attributes of its instances realized in software development projects. The problem of determining the quality of a process model also resembles the problem of determining the quality of software architecture: in both cases, there are certain solutions supposedly contributing to some quality attributes, but the actual effect of these solutions to the quality is unclear. A further similarity is that in both cases, quality assessment is difficult on the basis of the general process or architecture model only, without considering the actual concrete realization of these models.

In the context of system architecture, a popular technique to assess the quality of software architecture is to apply scenario-based approaches, like ATAM (Architecture Tradeoff Analysis Method) [Clements et al. 2002]. In ATAM, the quality requirements are first derived from business goals and concretized using scenarios. That is, for each quality requirement (say, UI portability), a concrete situation testing the quality requirement is given, related to an imaginary implementation of the system (say, “the GUI of the system is made browser-based in a month”). Such scenarios are then analyzed against the solutions in the architecture, trying to identify those solutions which affect the realization of the scenario. If the scenario is considered realizable, the solutions contributing to this quality attribute are identified and marked as “safe”. If the scenario is considered unrealizable, the solutions making the scenario difficult or impossible are identified as “risks”. The general idea of ATAM is to create in this way links between the quality attributes and solutions in the architecture. To focus the assessment on the most important requirements, the scenarios are prioritized so that less essential scenarios can be ignored in the analysis.

I argue that a similar method can be applied to the quality assessment of software development processes as well. That is, the practices in a software process model (solutions) can be analyzed against concrete situations (scenarios) testing certain desired quality attributes in an imaginary instance of the process model. In that way, I can infer not only the overall quality level of a process model, but also get a detailed explanation about which quality attributes are weak or strong in the process model, and why. I can also make observations on “safe” and “risky” practices in general: if a certain practice often appears as a “safe” solution, this practice is obviously beneficial, if another practice is frequently labeled “risky”, the value of the practice should be clearly questioned. The assessment process can be adjusted according to chosen goals or needs in a company, and carried out as lightly as possible.

The elements describing the quality of a piece of software have usually been referred to as quality attributes. Different software quality models have introduced selected sets of these quality attributes [Miller 2001]. ISO 9126 standard [ISO/IEC 9126-1:2001] is one example. ISO 9126 includes three perspectives of software product quality: internal quality, external quality and quality in use. Internal quality can be measured during development of the product, and external quality can be measured when the product is executed. Quality in use can be seen by the user while the product is applied in the intended fashion. The quality attributes, or, as the standard calls, quality characteristics, of external and internal quality are introduced in Figure 12 as an example.

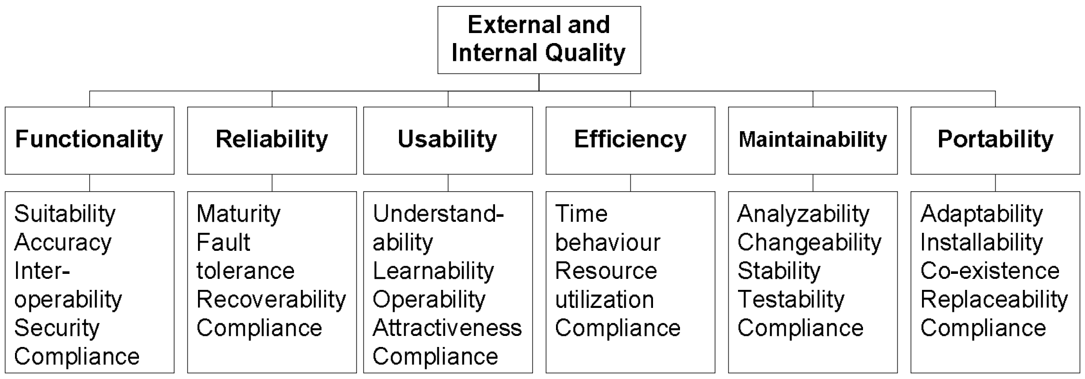

External and Internal Quality
Functionality
Reliability
Usability
Efficiency
Maintainability
Portability
Suitability Accuracy Inter-operability Security Compliance
Maturity Fault tolerance Recoverability Compliance
Understand-ability Learnability Operability Attractiveness Compliance
Time behaviour Resource utilization Compliance
Analyzability Changeability Stability Testability Compliance
Adaptability Installability Co-existence Replaceability Compliance

Figure 12. ISO 9126 quality attributes [ISO/IEC 9126-1:2001]

In this study I will exploit ISO 9126 to derive quality attributes for processes as part of the construction of quality profiles discussed in Subsection 5.2.3. However, this is only one possible technique of deriving process quality attributes, and the Q-PAM method does not take a standpoint on the technique. Indeed, a company could come up with the desired quality attributes as a result of an internal discussion on the goals of the assessment.

5.2.2 Method Overview

The first step in Q-PAM is to create a quality profile for the process (here, a process pattern language). The quality profile is a set of quality attributes considered essential in the assessment of the process. The quality profile thus depends not only on the quality requirements of the process, but also on the purpose of the assessment: the same process may be assessed with different profiles. Quality profiles are assumed to be obtained by extracting them from quality attribute lists available in standards. The construction of the quality profile is discussed in more detail in Subsection 5.2.3.

When the quality profile has been constructed, each quality attribute is associated with scenarios that serve as test cases for the quality attribute. A scenario is a concrete, desired situation in an imaginary instance of the process where the existence or non-existence of the required quality attribute can be verified. The construction of the scenarios is discussed in Subsection 5.2.4. Scenarios can be prioritized for more focused processing, if needed.

The next step is the actual quality analysis. Each (possibly prioritized) scenario is analyzed against the process patterns: which patterns (if any) support the realization of the scenario, and which patterns counteract against the scenario (if any). A tag is attached to the scenario, characterizing the extent to which the pattern language is considered to pass the scenario test, on the basis of the analysis. The analysis step is discussed in more detail in Subsection 5.2.5.

5.2.3 Creating Quality Profile

A quality profile is a (possibly hierarchically structured) set of quality attributes. The term quality profile has been used here in a similar meaning as Bosch [2000] has used it in the context of software architectures. In both cases, a profile is a means to capture a covering set of scenarios for a particular assessment purpose. A quality profile can be created on the basis of the requirements of a software development process (if such exist), a company's business goals, the purpose of the assessment, and/or a common quality framework. Here, I will use the quality attributes of ISO 9126 associated with external and internal quality as a basic source of the quality profile, interpreting and transforming the quality attributes for the context of processes. Also quality in use could be used but in my case I have not used them because they are less suitable. This is a straightforward technique that can be recommended in many cases, but I emphasize that other techniques could be used as well.

Let us consider a sample quality attribute in ISO 9126, efficient time behavior (that is, sub attribute of Time behavior under Efficiency). The standard defines this as "the capability of the software product to provide appropriate response and processing times and throughput rates when performing its function, under stated conditions". If I replace the words product and function with words process and task, respectively, this definition can be applied for processes as well, resulting in: the capability of the process to provide appropriate response and processing times and throughput rates when performing its tasks, under stated conditions. This kind of adaption is possible and reasonable to nearly all of the quality attributes in ISO 9126 regarding external and internal quality.

A quality profile obtained from a general quality model can be refined according to process-specific characteristics or purpose of the assessment. For example, Efficiency could be refined as Project manager time usage if the company is particularly interested in the efficient use of project manager resources. A more refined profile makes it easier to find scenarios related to the quality attribute.

5.2.4 Constructing Scenarios

The general idea of a scenario is to serve as a test case that can be run against the process patterns. For this purpose, a scenario should describe a concrete and measurable situation in an imaginary process instance (project). If the scenario represents a typical situation, a succeeding test suggests that the process pattern language normally supports the situation of the scenario. If the scenario represents a stress situation trying the limits of the process, a succeeding test gives an upper bound for the capacity of the process. A scenario can also test some specific part of the process that is of particular interest. Scenarios are found during a workshop by different stakeholders. I describe in Subsection 5.3.1 how the scenarios were found in our case.

Each quality attribute in the quality profile should be associated with at least one scenario. For example, assume that I am assessing a process pattern language for requirements analysis, and the quality profile contains quality attribute Changeability (as a sub attribute of Maintainability). This quality attribute could be further refined as Organizational changeability. A scenario could be then given for this quality attribute as follows:

> *Company X buys our company and wants to make our development process compatible with theirs. The requirements analysis part of our process is made compatible with X's process within half a year using nine person-months.*

Note that this kind of a change scenario requires exact time specifications to be analyzable. All the implications or assumptions need not be visible in the scenario, but they must be reasonably inferable on the basis of the scenario. In the example, company X should refer to an actual company, with known process practices.

Scenarios are as valuable assets for processes as test cases are for systems, recording important information related to the process. Thus, all scenarios given for a quality profile should be documented and preserved. However, scenarios may have different weight in an assessment project, and there may be limited resources to carry out the assessment. To be able to concentrate on the essential ones among a large set of scenarios, the scenarios can be prioritized according to their importance.

5.2.5 Analysis

During the analysis phase, each of the (highly prioritized) scenarios is considered, and the involved process patterns are identified. The involved patterns are those patterns that potentially have affect on the scenario. Essentially, the analysis means that the effect of these patterns on the scenario is studied. For each pattern involved in the scenario, a positive conclusion is that the scenario situation is supported by the pattern, so that the application of the pattern helps to realize the scenario. A negative conclusion is that the pattern either does not provide support for handling a situation that it is supposed to support, or it hinders or complicates the situation described by the scenario. A rationale explaining either a positive or negative conclusion is associated with the scenario. In the case studies I have marked positive and negative conclusions with N (non-risk) or R (risk), respectively.

Sometimes it may be difficult to conclusively argue that a scenario is realizable using the process patterns, but there are patterns that provide some assistance in the scenario. Similarly, there may be patterns which do not prohibit a scenario, but may be to some extent counteracting against it. In these cases, it would be sensible to use a more fine-grained result than just a binary tag.

After each scenario has been analyzed and tagged, the assessment data is in principle available. However, if there are several quality attributes (with analyzed scenarios), it may be difficult to present this data in a condensed, complete form. For this purpose, the quality attributes can be grouped and each group can be characterized with a ratio of succeeded and failed scenarios. In this way, it is possible to find larger “problem areas” in the process. For example, if many scenarios related to different sub-attributes of Efficiency fail, it seems reasonable to suggest that efficiency is a problem area in the process. A summary of the analysis of scenarios can also be presented by a table.

5.3 Evaluation of GSD Pattern Language

In this section evaluation of GSD Pattern Language is described. Subsection 5.3.1 discusses how to apply Q-PAM for this purpose and Subsection 5.3.2 describes analysis of the results.

5.3.1 Applying Q-PAM

Three faculty members from the Tampere University of Technology, three employees from Metso Automation and one employee from Teleca Inc. participated in the evaluation workshop along with the author. Three out of seven participants did not have any prior knowledge about the GSD Pattern Language and three out of seven participants did not have prior experience using the ATAM method in an industrial context.

The author introduced a candidate quality profile in the first evaluation session based on ISO 9126. Because of the limited time available, only a subset of quality factors from the ISO 9126 quality factors were used. A candidate quality profile was accepted with some changes after discussion. At the highest level, the chosen quality profile consisted of *Functionality*, *Efficiency* and *Portability*. *Functionality* was refined as *Suitability*, *Accuracy* and *Security*. *Efficiency* was refined as *Time Behaviour* and *Resource Utilization*. *Portability* was refined as *Adaptability*. The final quality profile is presented in Table 15.

Table 15. The quality profile used in the evaluation. The explanations are modified from ISO 9126.

Quality Factor	Explanation
Suitability (Functionality)	The capability of GSD Pattern Language to provide an appropriate set of phases for specified tasks and user objectives.
Accuracy (Functionality)	The capability of GSD Pattern Language to provide the right or agreed results or effects with the needed degree of precision.
Security (Functionality)	The capability of GSD Pattern Language to protect information and data so that unauthorized persons or systems cannot read or modify them and authorized persons or systems are not denied access to them
Time Behaviour (Efficiency)	The capability of GSD Pattern Language to provide appropriate response and processing times and throughput rates when performing its tasks, under stated conditions.
Resource Utilization (Efficiency)	The capability of GSD Pattern Language to use appropriate amounts and types of resources when the process performs its tasks under stated conditions.
Adaptability (Portability)	The capability of GSD Pattern Language to be adapted for different specified environments without applying actions or means other than those provided for this purpose for the process considered.

After choosing of the quality profile, the participants in the assessment workshop created scenarios and prioritized the most important of the scenarios. There were 57 different scenarios which were prioritized by participants by voting, resulting in 10 prioritized scenarios to be analyzed. The next subsection discusses the analysis of the results.

5.3.2 Analysis of Results

This subsection presents the results obtained from workshops of assessing GSD Pattern Language. The analysis of one of the resulting scenarios is introduced in Table 16. The patterns referred to in Table 16 are *Iteration Review, Multi-Level Daily Meetings, Common Repositories and Tools and finally, Common Processes,* which were presented in Table 10 on page 38.

Table 16. Example analysis of a scenario

Scenario	**S12**	An offshore designer decides to decrease the contents of a feature by 50%. In this way, he/she can get the feature to suit one iteration but the problem is that he/she does not talk with the product manager. This problem should be visible in two weeks.		
Response		A problem need to be solved in GSD as fast as in centralized development.		
Quality	**Main Attribute**	Accuracy (Functionality), Time Behaviour (Efficiency)		
	Pattern	**Analysis of Pattern Application**	**R**	**N**
	Iteration Review	The pattern ensures that the change can be found at the latest in the next Iteration Review.		N
	Multi-Level Daily Meetings	As a result of using this pattern, a project manager might also notice the change during daily meetings		N
	Common Repositories and Tools	Common repositories and reports will improve visibility of a project between different sites and from repositories it is possible to find task lists and reports e.g about remaining work, in which it is possible to notice the change by this pattern.		N
	Communication Tools	Communication tools make it easier to clarify change when it has been found.		N
	Common Processes	With Common processes, there can be a risk if there is not specific process guidelines to make a decision about making changes, and all project members have not been trained well.	R	
Result	**Some Support:** The implementation of the scenario S12 is supported through four patterns in the language and one pattern can have a risk.			

I illustrate the results of the analysis with a scenario-pattern matrix (Table 17), where for each scenario the involved patterns are marked with an N (non-risk) or R (risk). I have computed certain indicator values suggesting problematic scenarios or patterns. These indicators are intended only as hints; the actual conclusions can be made only after studying the seriousness of each risk separately. I have used the following indicators: IR (involvement ratio) = $(\sum N+\sum R)/\sum S$ indicating the potential applicability scope of the pattern with respect to this set of scenarios, RR (risk ratio) = $\sum R/(\sum N+\sum R)$ indicating the total degree of risk of the pattern with respect to the scenario set, and SI (support index) = $(\sum N-\sum R)/ \sum P$ indicating the level of support the pattern language provides for a scenario. Here $\sum N$ and $\sum R$ denote the number of N's and R's in a row/column, respectively, $\sum S$ denotes the number of scenarios and $\sum P$ the number of patterns. If IR is low, the pattern seems to be less relevant for the scenario set, if RR is close to 1, the pattern may cause more problems than benefits, if SI is negative the pattern language may counteract the scenario. The indicators are presented in Figure 13.

$IR = (\sum N+\sum R)/\sum S$
$RR = \sum R/(\sum N+\sum R)$
$SI = (\sum N-\sum R)/\sum P$

IR = Involvement ratio
RR = Risk ratio
SI = Support index

$\sum N$ = the number of N's in a row/column
$\sum R$ = the number of R's in a row/column
$\sum S$ = the number of Scenarious
$\sum P$ = the number of Patterns

Figure 13. Evaluation Indicators

Table 17. Summary of the analysis of scenarios for GSD Pattern Language

	S12	S3	S22	S16	S25	S31	S17	S19	S24	S28	IR	RR
GSD1												
GSD2		R									**0,1**	**1,0**
GSD3			R	N		N	R	N		N	**0,6**	**0,3**
GSD4			N								**0,1**	**0,0**
GSD5		R		R			R			R	**0,4**	**1,0**
GSD6	N	N		N	N	N	N				**0,6**	**0,0**
GSD7	N	N	N	R	N		N	N			**0,7**	**0,1**
GSD8		R		N	N		N				**0,4**	**0,3**
GSD9								N			**0,1**	**0,0**
GSD10												
GSD11												
GSD12	R			N		N		N			**0,4**	**0,3**
GSD13			N						N		**0,2**	**0,0**
GSD14	N	N	N		N	N		N	R		**0,7**	**0,1**
GSD15	N		N		N	N					**0,4**	**0,0**
GSD16								R			**0,1**	**1,0**
GSD17		N	N				R			N	**0,4**	**0,3**
GSD18		N	N			N		N	N	N	**0,6**	**0,0**
SI	**0,2**	**0,1**	**0,3**	**0,1**	**0,3**	**0,3**	**0**	**0,3**	**0,1**	**0,1**		

From Table 17 can be seen that IR (involvement ratio) was at least 0,6 with the following patterns: GSD3 (Collocated Kick-Off), GSD6 (Communication Tools), GSD7 (Common Repositories and Tools), GSD14 (Multi-Level Daily Meetings) and GSD18 (Notice Cultural Differences). These patterns seem to be the most important ones for GSD and they give good support with the set of scenarios used in this assessment.

Suspicious patterns were GSD2, GSD5 and GSD16 in which RR (risk ratio) was 1,0. GSD 02 (Fuzzy Front End) has a risk because the pattern did not include a proper change management process. GSD5 (Choose Roles in Sites) was interesting because it has only risks, but one main problem with this pattern was that there was not mentioned deputy persons to ensure communication if the main responsible person is not available and it was noticed in three scenarios. GSD16 (Organize Knowledge Transfer) has a risk because it did not include process knowledge which is also a key area to train, although training of common processes was mentioned in GSD3 (Collocated Kick-Off).

It can also be noticed that GSD1, GSD10, GSD11 do not have any marks. GSD1 is a GSD strategy pattern which is used mainly before the start of a project. GSD10 and GSD11 as well as GSD9 are patterns for different types of work allocation and the main work allocation pattern was GSD8 which was mainly used instead of GSD9, GSD10 and GSD11 in the assessment.

The index SI (support index) was from 0 to 0.3 and for five scenarios from ten this index was at least 0.2. Support index is at a relatively low level but it is understandable because a scenario is only one test case which cannot be supported by all patterns. On the other hand, if the SI is very low the result indicates that patterns need to be checked and analyzed to find possible improvement actions for them.

5.4 Summary of Q-PAM Evaluation

In this case I can conclude that although there are some suspicious patterns (GSD2, GSD5 and GSD16), as a whole the pattern language provides good support for the scenarios. For instance, patterns that relate to the application lifecycle management (GSD6 and GSD7) indicate strong support for the selected scenarios.

During the workshop, several improvement possibilities for GSD Pattern Language were found and the analysis resulted in a better understanding of the limits of the GSD Pattern Language. For example, the analysis resulted in the finding that GSD Pattern Language does not include all needed practices in critical fault management or knowledge transfer areas. GSD Pattern Language also assumes that the development environment is in a good shape and that the communication network is working at a reasonable level. Some patterns originally intended for the beginning of a project were also found useful during a project.

The results obtained from the evaluation of GSD Pattern Language also indicate important issues for GSD. One of the issues is secure shared GSD7 (Common Repositories and Tools) as an ALM solution: electronic connections (e-meetings, teleconferencing, web cameras, chat, wiki etc.) were seen as essential solutions to support a collaborative mode of work. This has also been indicated in other case studies related to global product development, for instance, in [Battin et al. 2001] and [Ramesh et al. 2006] (e.g. intranet data sharing, teleconferencing). The applicability of ALM was also studied to support the management of distributed software development projects in Paper III. The results showed that ALM supported the operation in a global development environment. The results of Q-PAM analysis support this claim, too. From all GSD process patterns presented in Subsection 5.3.2, GSD6 (Communication Tools) and

GSD7 (Common Repositories and Tools) are related to ALM. Analysis results indicate that both ALM related patterns support the selected scenarios. Only one risk issue was found to be related to the analysis of pattern GSD7 against scenario S16.

The results from earlier work in [Välimäki and Koskimies 2006], Paper I, II and V show that the most successful global software development issues have been improvements in visibility, management of features, communication, and commitment to the goals of the project. The importance of these issues for global software development has also been discussed in [Leffingwell 2007].

Communication problems have been resolved by utilizing for example GSD14 (Multi-Level Daily Meetings), GSD13 (Iteration Planning), and GSD15 (Iteration Review). These issues have also been discussed both in [Schwaber 2004] and [Schwaber 2007].

6 RELATED WORK

The objective of this chapter is to present existing work related to the main contributions in this thesis. Section 6.1 presents an analysis between GSD Pattern Language and related research and Section 6.2 discusses pattern mining and organizing pattern languages.

6.1 Analysis between GSD Pattern Language and Related Research

GSD has brought many benefits but also challenges. Solutions to these challenges have been investigated by several authors like [Coplien and Harrison 2005], [Sangwan et al. 2006], [Paasivaara et al. 2010], [Woodward et al. 2010], [Šmite et al. 2010], and [Richardson et al. 2010]. Subsection 6.1.1 discusses the similarities and differences between studies in [Coplien and Harrison 2005] and GSD Pattern Language. Respectively, Subsections 6.1.2, 6.1.3, 6.1.4, 6.1.5, and 6.1.6 discuss the similarities and differences between studies in [Sangwan et al. 2006], in [Paasivaara et al. 2010], in [Woodward et al. 2010], in [Šmite et al. 2010], in [Richardson et al. 2010], and GSD Pattern Language. Subsection 6.1.7 presents a summary of these subsections.

6.1.1 Agile Software Patterns

Coplien and Harrison [2005] include four pattern languages which are based on in-depth research of over 100 software development organizations. These four pattern languages are Project Management, Piecemeal Growth of the Organization, Organizational Style, and People and Code.

Coplien and Harrison [2005] concentrate mainly on collocated Agile projects, but it also includes some interesting patterns for GSD. Also the viewpoint of these patterns is much more detailed than in GSD Pattern Language. As an example, I discuss about the most important GSD-related patterns in the form of a patlet [Coplien and Harrison 2005, p. 26]. A patlet is a short summary of a patterns's problem and solution. Coplien and Harrison [2005] emphasize that communication is key in GSD and it is important to sustain communication in a distributed project. In Coplien and Harrison's Project Management pattern language, one example of an important communication-related pattern is *Community of Trust* which describes: *If* you are building any human organizations *Then* you must have a foundation of trust and respect for effective communication [Coplien and Harrison 2005, p. 349]. Although a trust issue is partly included in GSD Pattern Language, this pattern could be one additional GSD Pattern to emphasize the importance of trust as a foundation for co-operation between different sites. Another example of a communication-related pattern can be found from Organization Style pattern language as *Face-to-Face before Working Remotely*. This pattern describes that *If* a project is divided *Then* begin the project by inviting everyone to a meeting at a single place [Coplien and Harrison 2005, p. 358]. This pattern has the same idea as the GSD Pattern *Collocated Kick-Off*.

6.1.2 GSD Handbook

Sangwan et al. [2006] study GSD practices originating from Siemens Corporate Research (SCR). Siemens has more than 30 000 software engineers in many offices in most countries around the globe. SCR started this research program in 2003 SCR with different universities around the world. Many GSD projects from Siemens and by persons from eight schools in five nations across four continents participated in this research. It had three objectives: to find the best practices from project experiences, intensive literature analysis and experimental research which had never been done before in GSD.

Sangwan et al. [2006] introduce Model-Drive Rapid Application Development which is a high-level process framework that balances agility and discipline for GSD. The process framework includes a planning phase both for product development and for project management. The process framework describes suggestions for an organizational structure as well as how to monitor and control a project. Also some case studies are presented in a GSD environment. Sangwan et al.[2006] have quite many practices similar to GSD patterns. The format of presentation of Sangwan's practices is textual and these practices do not have any structure as a process pattern has. The organization of practices is made by a process framework and different workflows which help to find needed practices. One example of a similarity between practices of Sangwan et al. [2006] and GSD Pattern Language is *Feature Release Planning* practice in the former and *Divide and Conquer with Iterations* in the GSD Pattern Language. The goal in both is to make a feature release plan to manage development of a product. Also the implementation and result parts have similarities.

6.1.3 MaPIT Practices for GSD

Paasivaara et al. [2010] give a collection of best practices. The best practices are based on the experiences of companies, collected from research in MaPIT (Management, Processes and IT Support for Globally Distributed Software Development). The research project started in 2007 and ended in March 2010. The goals of that research were to present the challenges of GSD projects and to find solutions and useful practices for managing and working in GSD projects.

The research of Paasivaara et al. [2010] includes three parts. The first part is Planning and Practices which describes practices related to communication, collaboration and project ramp-up activities. It includes chapters Initiating GSD, Communication and Collaboration Practices and Using Scrum Practices in GSD Projects. The second part is Team and Individuals, which presents the problems and solutions behind group dynamics. It includes the chapters Trust and Distrust in GSD Projects, GSD Projects and Organizational Change, Structuring the Teamwork for GSD Projects, Commitment in GSD Projects and finally, Managing Cultural Differences in GSD Projects. The last part is Tools and Infrastructure which discusses the findings and theories behind the choice of collaboration tools. It includes the chapters Choosing Communication Media for GDS, Communication Tools for GSD Projects and Physical Workspaces and Distributed Teams. Chapters are written by different participants in a MaPIT research project.

Paasivaara et al. [2010] describe quite similar practices with GSD Pattern Language. One example of a similarity between practices of Paasivaara et al. [2010] and GSD Pattern Language is *Sprint demos* practice in the former and *Iteration Review* in the GSD Pattern Language. The goal in both of them is to check the results at the end of iteration.

6.1.4 Distributed Scrum Practices

Woodward et al. [2010] describe how to use Scrum in a globally-distributed company. This work proposes best practices which are founded in distributed projects both in IBM and other companies. The research was started in 2008 and ended in 2010. The goal of this research was to capture experience and helpful recommendations not only from IBM but also from knowledgeable Scrum Team members, coaches and consultants outside of IBM.

The work of Woodward et al. [2010] describes quite extensively the best practices for distributed Scrum. Best practices are presented for each phase of Scrum as Starting a Scrum project, Sprint Planning, Distributed Daily Scrum etc. Woodward et al. [2010] describe quite similar practices with GSD patterns. One example of a similarity between practices of Woodward et al. [2010] and GSD patterns is *Starting a Scrum project* practice in the former and *Divide and Conquer* in GSD Pattern Language. The goal in both of them is to plan the start of a project.

6.1.5 BTH Review on GSE

An investigation of empirical evidence in global software engineering (GSE) related literature has been carried out at BTH (Blekinge Institute of Technology) [Šmite et al. [2010]. Šmite et al. [2010] report their finding from investigating empirical evidence in global software engineering (GSE) related research literature. Šmite et al. [2010] report that the benefits of GSE are difficult to reach in companies. Šmite et al. [2010] also say that GSE research is still immature, because the number of empirical studies is still relatively small. The systematic review consists of a review of 59 papers which are published after the year 2000. The results of the systematic review are several descriptive classifications of the papers on empirical studies in GSE and also reports on some best practices identified from literature. Practices of Šmite et al. [2010] are partly similar to GSD patterns.

One example of a similarity between practices of Šmite et al. [2010] and GSD patterns is *Centralized project repository* in the former and *Common Repositories and Tools* in GSD Pattern Language. The goal in both of them is the possibility for global access to project data regardless of time and place.

6.1.6 GSE: A Software Process Approach

A study of three case studies in global software engineering (GSE) during nine year period has been carried out by a group of researchers at Lero research center, University of Limerick [Richardson et al. 2010]. The main result of this research is Global Teaming which is a software process for implementing GSE process efficiently in a company. Global Teaming has two specific goals which are *Define Global Project Management* and *Define Management between Locations*. These goals are further divided in practices and sub practices. These practices of Richardson et al. [2010] are quite similar to GSD patterns.

One example of a similarity between practices of Richardson et al. [2010] and GSD patterns is *Identify Communication Skills for GSE* in the former and *Communication Tools* in GSD Pattern Language. The goal in both of them is to find suitable communication methods and tools.

6.1.7 Summary of Analysis

Tables 18, 19, and 20 present an analysis between GSD Pattern and related research. In Table 18 GSD Pattern Language are compared with [Coplien and Harrison 2005] and [Sangwan et al. 2006].

Table 18. Analysis of GSD Pattern Language with [Coplien and Harrison 2005] and [Sangwan et al. 2006]

ID-Name	**[Coplien and Harrison 2005]**	**[Sangwan et al. 2006]**
GSD1-GSD Strategy	No corresponding practice	Critical success factors (p.10) Structure of a GSD project (p.113) Offshoring should be viewed as a long-term strategy in which the remote supplier develops a competence center, not as a short term cost reduction (p.233)
GSD2-Fuzzy Front End	No corresponding practice	Elicitation in Requirements engineering workflow (p.26)
GSD3-Collocated Kick-Off	Face to Face before Working Remotely (p.199) Community of Trust (p.102)	Team-building workshop (p.208)
GSD4-Divide and Conquer with Iterations	No corresponding practice	Feature release planning (p. 81)
GSD5-Choose Roles in Sites	No corresponding practice	Supplier manager is a member of the central team who manages developers at remote development (p.123) Exchange of people (p.175) Both project and local site managers (p.206) Chief architect was responsible for decision making for the application package (p. 206)

GSD6-Communication Tools	No corresponding practice	Communication and Collaboration Infrastructure (p.155)
GSD7- Common Repositories and Tools	No corresponding practice	Knowledge Management Infrastructure (p.160)
GSD8-Work Allocation	Distribute Work Evenly (p.206)	Understand dependencies (p.231)
GSD9-Architectural Work Allocation	Conway's Law (p.192), Organization Follows Location (p.194)	Architecture design workflow (p.54) For each module there will be a focus group formed, such that the remote teams work with multifunctional experts from central team (p.114) Minimize communication (p.176) Early phase activities are critical to provide understanding and divide the work package for remote teams (p.232)
GSD10-Phase- Based Work Allocation	No corresponding practice	Structure of a GSD project (p.113)
GSD11-Feature-Based Work Allocation	Feature Assignment (p. 264)	Technical activities are controlled by the functional team leaders within the central teams (p.129)
GSD12-Common Processes	No corresponding practice	Balance flexibility and rigidity (p.232)
GSD13-Iteration Planning	Distribute Work Evenly (p.206)	Planning during the Elaboration and Construction Phases (p.89) Bottom-Up Estimate (p.103) The goals of each sprint are planned centrally using the build plan. Detailed planning should be done by each development team for each sprint, with the support of the supplier manager (p.129) Frequent trips by a supplier manager will be made to the remote sites, especially for the kick-off meeting for a new iteration and to view the results of each sprint (p.130)
GSD14-Multi-Level Daily Meetings	No corresponding practice	Project status tracking was done during weekly teleconferences (p.207)
GSD15-Iteration Review	No corresponding practice	Planning during the Construction Phase (p. 89) Frequent trips by a supplier manager will be made to the remote sites, especially for the kick-off meeting for a new iteration and to view the results of each sprint (p.130)
GSD16-Organize Knowledge Transfer	No corresponding practice	The time spent at the central site is used to train the future remote team members (p.112) Delegate trusted staff to new remote sites (p.131) Technical training (p.210)
GSD17-Manage Competence	No corresponding practice	No corresponding practice
GSD18-Notice Cultural Differences	No corresponding practice	Multicultural workshop (p.208)

From Table 18 can be seen that Coplien and Harrison [2005] describe only few similar patterns with GSD patterns. One reason is that Coplien and Harrison [2005] concentrate mainly on collocated Agile projects and not the distributed ones. However, Coplien's and Harrison's more detailed patterns could be studied as candidates to increase the level of details in GSD patterns.

Sangwan et al. [2006] have quite many practices similar to GSD patterns but the level of abstraction is different. Usually practices are also not easy to identify, because they are part of textual descriptions. Some practices are described at a more detailed level than in GSD patterns. The contents of Sangwan et al. [2006] could be utilized to improve GSD patterns not only to increase details in project management patterns but also to broaden the scope of GSD Pattern Language for the development of a product as well.

In Table 19, GSD Pattern Language is compared with [Paasivaara et al. 2010], [Woodward et al. 2010], and [Šmite et al. 2010].

Table 19. Analysis of GSD Patterns with [Paasivaara et al. 2010] and [Woodward et al. 2010]

ID-Name	**[Paasivaara et al. 2010]**	**[Woodward et al. 2010]**	**[Šmite et al. 2010]**
GSD1-GSD Strategy	GSE Strategy (p.3)	No corresponding practice	No corresponding practice
GSD2-Fuzzy Front End	No corresponding practice	Starting a Scrum project (p. 39)	No corresponding practice
GSD3-Collocated Kick-Off	Kick off (p.8)	Communicate with distributed team members (p.20)	Partly: Face to Face meetings
GSD4-Divide and Conquer with Iterations	Involve the whole team in planning (p.12*)*	Release Planning (p. 39) Create the Release Plan (p.56)	Incremental short-cycle development
GSD5-Choose Roles in Sites	Define clear roles and responsibilities (p.7) Define and communicate a clear organizational structure (p.20) Frequent visits (p.48)	No corresponding practice	No corresponding practice
GSD6-Communication Tools	Agree on communication practices and tools (p.22) Multiple communication modes (p.50) Communication Tools for GSD projects (p.127)	Tools - p.26	Reliable infrastructure, rich communication media
GSD7- Common Repositories and Tools	Agree on communication practices and tools (p.22) Issue trackers (p.141)	Tools & File Sharing – p.26	Centralized project repository, common configuration management tool support
GSD8-Work Allocation	Division of work (p.5)	Starting a Scrum project (p. 39)	Task distribution based on architectural decoupling and low dependencies across remote locations
GSD9-Architectural Work Allocation	No corresponding practice	Starting a Scrum project (p. 39)	Task distribution based on architectural decoupling and low dependencies across remote locations

GSD10-Phase- Based Work Allocation	Division of work (p.5)	Starting a Scrum project (p. 39)	No corresponding practice
GSD11-Feature-Based Work Allocation	No corresponding practice	Starting a Scrum project (p. 39)	No corresponding practice
GSD12-Common Processes	Implement good working practices and processes (p.13)	No corresponding practice	No corresponding practice
GSD13-Iteration Planning	Involve the whole team in planning (p.12) Sprint planning meetings (p.42)	Sprint Planning (p 85)	Partly: Effective and frequent synchronous communication
GSD14-Multi-Level Daily Meetings	Arrange frequent status meetings (p.25) and Arrange weekly meetings across teams (p.26)	Distributed Daily Scrum Meetings (p.97)	Effective and frequent synchronous communication
GSD15-Iteration Review	Sprint demos (p.44)	End of Sprint Reviews (p. 147)	Partly: Effective and frequent synchronous communication
GSD16-Organize Knowledge Transfer	Invest in knowledge transfer (p.9) Arrange training (p.24)	No corresponding practice	Partly: Temporal collocation, Exchange visits
GSD17-Manage Competence	No corresponding practice	No corresponding practice	No corresponding practice
GSD18-Notice Cultural Differences	Cultural sensitivity training (p.103)	Cultural Differences p. 21	No corresponding practice

From Table 19 can also be seen that the practices of Paasivaara et al. [2010] describe clear correspondences with GSD patterns. Practices are analyzed and described at a more detailed level than in GSD patterns. The practices of Paasivaara et al. [2010] could be utilized to improve GSD patterns both to increase details in the current GSD patterns and to add some interesting patterns to the current GSD Pattern Language. However, there are four GSD patterns which are not in [Paasivaara et al. 2010]: *Fuzzy Front End, Architectural Work Allocation, Feature- Based Work Allocation,*and *Manage Competence.*

Also Woodward et al. [2010] describe quite similar practices with GSD patterns. Furthermore, these practices include more details than GSD patterns. However, there are some patterns which are not included in [Woodward et al. 2010], such as *GSD Strategy, Choose Roles in Sites, Organize Knowledge Transfer and Manage Competence.*

Šmite et al. [2010] describe names, benefits and constraints of practices, but these practices do not have detailed descriptions at all. All of these practices can be found from GSD Patterns. However, there are some patterns which are not included in [Šmite et al. 2010], such as *GSD Strategy, Fuzzy Front End, Choose Roles in Sites, Phase-Based Work Allocation, Common Processes, Manage Competence or Notice Cultural Differences.*

In Table 20, GSD Pattern Language is compared with [Richardson et al. 2010].

Table 20. Analysis of GSD Pattern Language with [Richardson et al. 2010]

ID-Name	[Richardson et al. 2010]
GSD1-GSD Strategy	SP 1.1 Global Task Management (1): *Determine Team and Organizational Structure Between Locations*
GSD2-Fuzzy Front End	-
GSD3-Collocated Kick-Off	SP 2.1 Operating Procedures (4): *Implement Strategy for Conducting Meetings Between Locations*
GSD4-Divide and Conquer with Iterations	SP 1.3 Global Project Management (4): *Establish Cooperation and Coordination Procedures Between Locations* SP 2.2 Collaboration Between Locations (4): *Collaboratively Develop, Communicate and Distribute Among Interfacing Teams the Commitment Lists and Work Plans that are Related to the Work Product or Team Interfaces*
GSD5-Choose Roles in Sites	SP 2.1 Operating Procedures (3): *Establish Communication Interface Points Between the Team Members*
GSD6-Communication Tools	SP 1.2 Knowledge and Skills (3): *Identify Communication Skills for GSE* SP 1.3 Global Project Management (4): *Establish Cooperation and Coordination Procedures Between Locations* SP 2.1 Operating Procedures (2): *Implement a Communication Strategy for the Team*
GSD7- Common Repositories and Tools	SP 1.3 Global Project Management (4): *Establish Cooperation and Coordination Procedures Between Locations*
GSD8-Work Allocation	SP 1.3 Global Project Management (2): *Assign Tasks to Appropriate Team Members* SP 2.2 Collaboration Between Locations (2): *Collaboratively Establish and Maintain the Work Product Ownership Boundaries Among Interfacing Locations Within the Project or Organization*
GSD9-Architectural Work Allocation	SP 1.3 Global Project Management (2): *Assign Tasks to Appropriate Team Members*
GSD10-Phase- Based Work Allocation	SP 1.1 Global Task Management (2): *Determine the Approach to task Allocation Between Locations* SP 1.3 Global Project Management (2): *Assign Tasks to Appropriate Team Members*
GSD11-Feature-Based Work Allocation	-
GSD12-Common Processes	SP 2.2 Collaboration Between Locations (3): *Collaboratively Establish and Maintain Interfaces and Processes Among Interfacing Locations for the Exchange of Inputs, Outputs, or Work Products*
GSD13-Iteration Planning	SP 1.3 Global Project Management (1): *Identify GSE Project Management Tasks* SP 2.2 Collaboration Between Locations (1): *Identify Common Goals, Objectives and Rewards for the Global Team*

GSD14-Multi-Level Daily Meetings	SP 1.3 Global Project Management (6): *Establish a Risk Management Strategy* SP 2.1 Operating Procedures (1): *Define How Conflicts and Differences of Opinion Between Locations are Addressed and Resolved*
GSD15-Iteration Review	SP 1.3 Global Project Management (1): *Identify GSE Project Management Tasks* SP 1.3 Global Project Management (5): *Establish Reporting Procedures Between Locations* SP 2.2 Collaboration Between Locations (1): *Identify Common Goals, Objectives and Rewards for the Global Team*
GSD16-Organize Knowledge Transfer	SP 1.2 Knowledge and Skills (4): *Establish Relevant Criteria for Training Teams*
GSD17-Manage Competence	SP 1.2 Knowledge and Skills (1): *Identify Business Competencies Required by Global Team Members in Each Location* SP 1.2 Knowledge and Skills (4): *Establish Relevant Criteria for Training Teams* SP 1.3 Global Project Management (1): *Identify GSE Project Management Tasks*
GSD18-Notice Cultural Differences	SP 1.2 Knowledge and Skills (2): *Identify the Cultural Requirements of Each Local Sub-team* SP 1.3 Global Project Management (3): *Ensure Awareness of Cultural Profiles by Project Managers*

From Table 20 can be seen that the practices of Richardson et al. [2010] describe clear correspondences with GSD patterns. Practices are analyzed and described at a more detailed level, especially related to cultural aspect, than in GSD patterns. The practices of Richardson et al. [2010] are also grouped according to specific goals. The practices of Richardson et al. [2010] could be utilized to improve GSD patterns to increase details in the current GSD patterns. However, there are two GSD patterns which are not in [Richardson et al. 2010]: *Fuzzy Front End,* and *Feature- Based Work Allocation.*

6.2 General Issues Related to Patterns

In this section, two areas related to patterns are discussed and mapped to my work: pattern mining, and organizing a pattern language.

6.2.1 Pattern Mining

Patterns can be roughly divided into two groups which are system and process patterns. Pattern mining has been mainly carried out for system patterns [Buschmann et al. 2007]. In the area of process patterns especially Coplien and Harrison [2005] have done this kind of research.

Pattern mining refers to the activity of finding and identifying good practices from systems or processes. Patterns are generally applied instances of solutions in systems or processes. One example of the results of pattern mining is [Gamma et al. 1995] which contains a pattern found by analyzing one sizeable banking system. Most of the patterns were generally known but Gamma [1995] discovered existing solutions and gave them a semi-formal description.

A method for pattern mining is presented in [Buschmann et al. 2007]. In this method, patterns are mined from best practices which are presented by an expert from an industry partner. This expert tries to share all knowledge with an interviewer who collects the most important information to pattern drafts which are improved in workshops with domain area experts.

A pattern mining method based on architectural analysis sessions is presented in [Leppänen et al. 2009]. The architecture analysis method is based on the architecture trade-off analysis method, ATAM [Clements et al. 2002]. According to [Leppänen et al. 2009], the ATAM method begins with architecture presentations and the proceeds to analyze architectural solutions using probable use cases and scenarios that add business value to software. If the architecture presentation or the scenario analysis shows traits of patterns, then a pattern draft is written down by the members of the assessment team. Finally, the identified patterns are finalized in pattern writers' workshops. [Leppänen et al. 2009].

A pattern mining method based on team interviewing is described in [Coplien and Harrison 2005]. This method begins with team interviews to gather data. The next step is to analyze data using social network techniques to build organizational models. After that the results will be presented to the team and the model will be adjusted if necessary. The next step is to catalog the analysis results and look for common patterns, identifying the problem, forces, and solution for each pattern. After that the patterns will be captured in pattern form. The final steps are to look for links between patterns to form sequences for applying the patterns and organize the sequences into pattern languages.

A pattern mining method to mine inspection improvement patterns is described in [Harjumaa 2005]. These patterns are mined from inspection literature, research articles related to utilizing inspections and observations during industrial experiments. After that the first version of a pattern catalogue is formed. The next step is to refine the pattern catalogue with other experiments in an industrial environment.

In this study, the mining of patterns is implemented according to a case study approach, as described in Section 3.

6.2.2 Organizing Pattern Languages

Definitions of pattern languages were presented in Subsection 2.3.4. Pattern languages can be divided into two groups which are problem and solution-centered approaches [Eloranta et al. 2009]. In problem-centered approach, the language's domain can define the structure of the language. For example in the pattern language of Alexander [Alexander 1997] the domain is building of cities, houses etc. In this case, the domain is very hierarchical which can be utilized to also make the pattern language more hierarchical. Problem-centered languages can also be organized by a multi-dimensional space. In this space, one pattern can be located according to different dimensions. For example, in the pattern language of Vesiluoma [Vesiluoma 2007] dimensions are knowledge-sharing interfaces (e.g. project, organization etc.), software engineering activity types (e.g. managing, value adding etc.) and project lifecycle phases (e.g. establishment, realization etc.) and a pattern of this language is defined according to some of these dimensions. Similarly, in this study the patterns are organized by project lifecycle phases based on PRINCE2 process description (e.g. starting up a project, initiating a project etc.).

The organization of the solution-centered languages is based on the relationships between solutions. Usually these solutions are organized according to the application order if the former pattern solution assumes the latter. [Eloranta et al. 2009] In this case, pattern organization can be presented as a graph. The organization of patterns in this study is also possible to present as a graph if I start from the beginning of the PRINCE2 process description in Figure 11 on page 42 and list all patterns as a graph when each pattern is reached for the first time. One example of this approach will be in Appendix C.

7 INTRODUCTION TO THE INCLUDED PUBLICATIONS

The following chapters describe the research contributions and key findings for each paper. In addition to the Papers included in this thesis, the author has co-authored three papers related to the topic [Välimäki and Koskimies 2006], [Kääriäinen and Välimäki 2008], and [Kääriäinen et al. 2009].

7.1 Requirements Management Practices as Patterns for Distributed Product Management

The focus of Paper I is to study and find the best practices to support distributed business requirements management during the early phases of product and project development. This paper also describes the first version of the mining process for process patterns.

The experiences and improvement ideas of requirements management have been collected from the case *GSD in the work of product managers*. The starting point of this paper was that system products need to be developed faster in a global development environment. A more efficient user requirements collection and product feature analysis becomes more important to meet strict time-to-market and quality constraints. This information is needed when the goals and features to be created in a project are planned.

The first key result of Paper I is the improved process of mining requirements management process patterns. The pattern mining method was described in more detail in Section 3 (Pattern Mining in GSD). The second key result is the first version of product management patterns which were described in more detail in Section 4 (GSD Pattern Language and Patterns). Both of the above key results are contributions of the author of this thesis.

7.2 Patterns for Distributed Scrum – a Case Study

Paper II continues the work started in Paper I. Paper II focuses on two key elements of the thesis: the pattern mining method for process patterns and GSD process patterns for Agile project management. The paper is based on data collected from one of the cases of *GSD Project management in Agile projects* and literature study of the main writer. The starting point of this paper was that one response to an ever-complicated environment is the rise of Agile methods [Abrahamsson et al. 2002]. One Agile project management, which is based on Agile values and Lean principles, is Scrum. Usually Agile methods are applied just to local development but their potential for supporting more effective GSD environments has been studied. One example of this kind of research is Paper II.

The first key result of Paper II is the second version of the pattern mining method which was described in more detail in Section 3 (Pattern Mining in GSD). The second key result is the first version of Agile project management patterns which were described in more detail in Section 4 (GSD Pattern Language and Patterns). Both of the above key results are contributions of the author of this thesis.

7.3 Get a Grip on your Distributed Software Development with Application Lifecycle Management (co-author)

Paper III continues the work started in [Kääriäinen and Välimäki 2008]. Paper III focuses on building the second version of the Application Lifecycle Management framework, analyzing the case company's ALM solutions and continuing the collection of information for GSD patterns from an ALM tools point of view.

In this paper, we study the applicability of Application Lifecycle Management (ALM) for the management of distributed software development projects. The research analyses the ALM solutions of the case company against the ALM framework and further considers on how the solutions meet the functions of the triple C-model (Communication, Cooperation and Coordination). [Anderl et al. 2008]. The research also collected distribution-related issues based on ALM framework elements. Software teams that were the subject of this study were operating in a geographically-distributed development environment and were the same teams as in [Kääriäinen and Välimäki 2008]. According to our results, ALM can support the management of a distributed project by facilitating communication, cooperation and coordination. In our case, a central ALM database with common processes and tools enabled central orchestration for the software project that operated in a global development environment.

The first key result of Paper III is the second version of the Application Lifecycle Management framework, which was described in more detail in Subsection 2.3.2 (Application Lifecycle Management Framework). The second key result is the analysis of the case company's ALM solution. The third key result is mapping how ALM framework elements support the elements of a 3C-model. In this case, 3C comes from the words Communication, Cooperation and Coordination. One result was that ALM supports GSD from a 3C-model point of view. These results were used in the elaboration of GSD Pattern Language which were presented in Section 4. This is a different 3C model used in this thesis with GSD project management. The 3C model used with GSD project management comes from the words Communication, Coordination and Control which better suits a project management approach.

These key results are mainly contributions of Mr. Jukka Kääriäinen. The author of this thesis also participated in making the second and third key result and also gathered information for GSD Pattern Language which is presented in Paper V.

7.4 Scenario-Based Assessment of Process Pattern Languages

The focus of Paper IV is to describe a more light-weight method for assessing processes and to make assessments of two process pattern languages.

Current standards and models for the quality of software development processes lead to a coarse-grained quality model which is heavy and difficult to focus for specific purposes. There is a need for a more light-weight method for assessing processes that can be expressed as process pattern languages. The method is based on imitating an existing software architecture evaluation method, ATAM, in the context of processes. The main advantages of the method are more fine-grained assessment in terms of quality attributes, the possibility to tune the assessment for a certain purpose, and a more light-weight assessment procedure. In Paper IV the assessment method is illustrated in the case of two process pattern languages.

The first key result of Paper IV is a more light-weight method for assessing processes that can be expressed as process pattern languages which is described in more detail in Section 5 (Evaluation of GSD Pattern Language). The second key result is the evaluation results of two process pattern languages. The author of this thesis participated in making an ATAM-based evaluation method and also made the evaluation of another pattern language which was an earlier version of GSD Pattern Language. The evaluation of GSD Pattern Language was presented in Section 5 (Evaluation of GSD Pattern Language).

7.5 Global Software Development Patterns for Project Management

Paper V focuses on the introduction of GSD Pattern Language and its validation using the validation method described in Paper IV.

Global software development with the Agile or plan-driven development process has been taken into use in many companies. GSD offers benefits but also new challenges without known, documented solutions. The goal of this research is to present the current best practices for GSD in the form of process patterns for project management, evaluated by using a scenario-based assessment method. The best practices have been collected from case studies *GSD Project management in plan-driven projects*, *GSD in the work of product managers* and *GSD project management in Agile projects* which were described in Section 3.3 (Industrial Context and Case Studies). It is expected that the resulting pattern language will help other companies to improve their GSD processes by incorporating the patterns in the processes.

The first key result of Paper V is the introduction of GSD Pattern Language which is described in more detail in Section 4 (GSD Pattern Language and patterns). The second key result is the evaluation results of GSD Pattern Language. Both of the above key results are contributions of the author of this thesis. This paper was granted the best paper award in EuroSPI 2009.

7.6 Applying Application Lifecycle Management for the Development of Complex Systems: Experiences from Automation Industry (co-author)

Paper VI continues the work started in [Kääriäinen and Välimäki 2008] and Paper III. Paper VI focuses on using the second version of the Application Lifecycle Management framework to analyze the main company's ALM solutions. This paper also presents the third version of the Application Lifecycle Management framework, and continued the collection of information for GSD Pattern Language from an ALM tools point of view.

In this paper we present an industrial study about the history of Application Lifecycle Management (ALM) improvement in the main company. The study is part of broader research with the aim to improve global development in the main company. The improvement of the ALM started in 2006 when the company decided to acquire a commercial ALM solution. Two software teams started to pilot the solution and after various steps ended up fairly different ALM solutions. This paper concludes the history and experiences of ALM improvement and discusses the reasons why two teams ended up to different solutions. The improvement of ALM solutions has been facilitated with the use of ALM framework.

The first key result of Paper VI is an application of the second version of an ALM framework for documenting and analyzing the case company's ALM solutions. The second key result is a description of the history of ALM development in the main company and how and why ALM solutions evolved in the context of the main company. The third key result is a description of the third version of an ALM framework which was described in more detail in Subsection 2.3.2 (Application Lifecycle Management Framework). An ALM framework was complemented with the relations between framework elements.

These key results are mainly the contributions of Mr. Jukka Kääriäinen. The author of this thesis also participated in making the second key result and also gathered information for GSD Pattern Language which was presented in Paper V and in Section 4 (GSD Pattern Language).

8 CONCLUSIONS

This section describes answers to research questions, limitations and future research needs.

8.1 Answers to Research Questions

RQ1: How should good project management practices in GSD projects be presented?

Presentation form of project practices should be easy to understand and apply. It is important to understand what the solution is, and to find out if this practice is useful in a certain project. It is important to understand what kind of problem needs to be solved, what is the context, and what are the forces which need to be taken care of before trying to apply new practices in a certain project. The format of a pattern has been found to be a useful aid to present essential parts of best practices and patterns have been used in this work.

Another view of presentation of project practices is how these practices should be organized because only a group of patterns does not help to use them as a whole. There are different ways to organize a group of patterns to a pattern language. In this work, project management patterns are organized by PRINCE2 which is a process-driven project management method. This kind of structure helps a project manager to find which patterns could be used in a certain main process in PRINCE2.

RQ2: *How can good project management practices be found in GSD projects?*

To increase the understanding of a project management process, a pattern mining method has been developed. The results clearly indicate that the pattern mining method is a technique for finding good process patterns within GSD projects. The pattern mining method is also useful because the process patterns have been observed to be good in an evaluation workshop.

RQ3: *What are the good project management practices in GSD projects?*

As a result of the pattern mining in case projects and related literature analysis, good project management practices for GSD have been discovered and described. Of course project management practices depend on project contextual factors. These factors are e.g. size, collaboration modes, type of project and number of distributed sites. That is why it is important to check which project management practices are suitable for a certain project. Project management practices for GSD have been found useful not only in researched projects but also in some other companies during different workshops. As a conclusion, good project management practices for GSD is credited for increasing the understanding of GSD project management and, as such, for supporting GSD project management analysis and providing a basis for GSD project management improvement within GSD projects.

RQ4: *How can these practices be evaluated?*

A novel scenario-based evaluation method for process patterns (Q-PAM) has been developed as part of this work. Q-PAM method is inspired by ATAM (Architecture Tradeoff Analysis Method) [Clements et al. 2002]. Q-PAM method includes three phases: creating a quality profile, constructing scenarios, and analysis. The evaluation of GSD Pattern Language gave both valuable information about limitations of GSD patterns and improvement ideas to further improve GSD patterns.

8.2 Limitations and Threats

The limitations of this work are discussed here. The main limitations are related to the generalization of the results, coverage of GSD Pattern Language, and a maturity of GSD Pattern Language.

Generalization of the results is demanding because research of this work has been conducted in one company with case studies. The results of case studies are difficult to generalize. Also, the researched projects had their own project factors which have affected the results of this research. Additionally, patterns proposals were created by the author of this thesis and the reviewers of these patterns have been persons from the same company which could affect which patterns were discovered. To overcome this limitation with GSD Pattern Language, literature is also used when GSD Pattern Language has been made. Additionally, the evaluation of GSD Pattern Language has been organised as presented in Section 5. Reviews of GSD Pattern Language have also been made by persons from different companies and this has also been a way to get validation from other companies and a way to improve GSD Pattern Language. Additionally, GSD Pattern Language was also published in an international conference.

Another limitation related to GSD Pattern Language is the *coverage of GSD Pattern Language*. The GSD Pattern Language was developed by research of case studies, reviewed with representatives from different companies and evaluated by the Q-PAM method. GSD Pattern Language cover quite a lot with respect to PRINCE2 processes, but as reviews, the Q-PAM evaluation and the analysis of related research has shown, more research is needed to improve the coverage of GSD Pattern Language.

Maturity of GSD Pattern Language is also one limitation. GSD Pattern Language is still evolving and has not been applied as a whole in any other company. To overcome this limitation, reviews of GSD Pattern Language have also been made by persons from different companies as described earlier.

The limitation of research can also be evaluated through *threats to the validity (construct, internal, and external) and reliability* [Yin 2003]. According to Runeson and Höst [Runeson and Höst 2009] Construct validity means an aspect of validity which reflects to what extent the operational measures that are studied really represent what the researcher has in mind and what is investigated according to the research questions. Runeson and Höst [Runeson and Höst 2009] also describe as an example that if questions are not interpreted in the same way by the researcher and the interviewee, there is a threat to the construct validity. To overcome this threat, questions of the questionnaire and the interviews were piloted by one project manager in a case company to ensure that used terms and questions were understandable for the case. Also, the researcher was working in the same company when the terms and concepts were understood in the same way.

Internal validity [Runeson and Höst 2009] means an aspect of validity which is of concern when causal relations are examined. When the researcher is investigating whether one factor affects an investigated factor there is a risk that the investigated factor is also affected by a third factor. If the researcher is not aware of the third factor and/or does not know to what extent it affects the investigated factor, there is a threat to the internal validity. To overcome this threat, the different patterns can be partly conducted from case data, but the affects of literature and different workshops and reviews for GSD Pattern Language are not so clearly described and written down.

External validity [Runeson and Höst 2009] means an aspect of validity which is concerned with to what extent it is possible to generalize the findings, and to what extent the findings are of interest to other people outside the investigated case. During analysis of external validity, the researcher tries to analyze to what extent the findings are of relevance to other cases. There is no population from which a statistically representative sample has been drawn. However, for case studies, the intention is to enable an analytical generalization where the results are extended to cases which have common characteristics and hence for which the findings are relevant, i.e. defining a theory. External validity is already discussed in the second paragraph of this section.

Reliability [Runeson and Höst 2009] means an aspect which is concerned with to what extent the data and the analysis are dependent on the specific researchers. Hypothetically, if another researcher conducts the same study later on, the result should be the same. Threats to this aspect of validity are, for example, that it is not clear how to code collected data or if questionnaires or interview questions are unclear. To overcome this threat, the pattern mining method is described in Section 3.1. In any case, the analysis of literature and the results of reviews might be dependent on the specific researchers and participants in reviews because different researchers and participants in reviews could have different kinds of experience and knowledge about the research area.

8.3 Future Research

Future research directions include the analysis of experiences with the current patterns in other development projects, the improvement of the patterns and the creation of new patterns according to the feedback gained from different projects or other related research. Case studies of other development projects could give new views of GSD Pattern Language because project factors can be different. One example of a project factor is team structure. In this work patterns have been found cases which are mainly organized according model 3 and 4 as described in Figure 2 on page 10. Patterns can also be mined not only from the level of a project but also from the level of a large-scale organization.

This work has also concentrated on project management practices. Another research direction is to identify patterns from other processes of GSD such as development, support processes or maintenance processes. Also, new patterns can be mined from the hardware development process.

GSD Pattern Language can also be further researched to form a maturity model of GSD Pattern Language for analysing a GSD project against this maturity model. The analysis can present an analysis and propose some best practices or patterns to make the implementation of the GSD project more efficient.

How to integrate GSD Pattern Language efficiently in used project management tools and guidelines can also be one direction of future research. It is important that GSD patterns can be used easily when a GSD project is ongoing.

Also the improvement of the pattern mining method is one possible future research direction. How can pattern mining methods be provided with a tool which supports the management of gathered data, creates summary tables, and finally, forms pattern proposals?

Finally, the evaluation method for patterns can also be one future research direction. Possible research challenges include finding methods to create scenarios more easily and supporting evaluation methods with a tool to improve the efficiency of the evaluation process.

References

Abrahamsson, P., Salo, O., Ronkainen, J., Warsta, J. (2002). Agile software development methods: Review and Analysis. Espoo, Finland: Technical Research Centre of Finland, VTT Publications 478, http://www.inf.vtt.fi/pdf/publications/2002/P478.pdf.

[AgileAlliance 2009] www.agilealliance.org, (available 24.05.2009).

Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, M. and Angel, S. (1977). A Pattern Language: Towns, Buildings, Construction. Oxford University Press, New York.

Allen, T.J. (1984). Managing the Flow of Technology: Technology Transfer and the Dissemination of Technological Information within the R&D Organization. Cambridge: The MIT Press.

[Ambler 2010] Examing the Agile manifesto. http://www.ambysoft.com/essays/agileManifesto.html (available 10.4.2010).

Ambler, S. (1998). Process Patterns – Building Large-Scale Systems Using Object Technology. Cambridge University Press/SIGS Books.

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

Appleton, B. (1997). Patterns and Software – Essential Concepts and Terminology. Object Magazine Online, Vol. 3, No. 5. http://www.cmcrossroads.com/bradapp/docs/patterns-intro.html (available 5.1.2010).

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

Beck, K. and Andres, C. (2004). Extreme Programming Explained, 2nd ed. Addison-Wesley Longman.

Bentley C. (2005). The Essence of the Prince2: Project Management Method, 2005 Revision. Protec.

Bosch, J. (2000). Design and use of software architectures. ACM Press, Addison-Wesley.

Buschmann, F., Henney, K., Schmidt, D.C. (2007). Pattern Oriented Software Architecture Volume 5: On Patterns and Pattern Languages. John Wiley & Sons.

Carmel, E. (1999). Global Software Teams, Collaborating Across Borders and Time Zones. Prentice Hall.

Carmel, E., Agarwal, R. (2001). Tactical Approaches for Alleviating Distance in Global Software Development. IEEE SOFTWARE, March/April 2001, pp.22-29.

Carmel, E., Tjia, P. (2005) Offshoring information technology, Sourcing and Outsourcing to a Global Workforce. Cambridge University Press.

Casey, V, Richardson, I. (2006) Uncovering the reality within virtual software teams. In Proceedings of the 2006 international workshop on Global software development for the practitioner (GSD '06). ACM, New York, NY, USA, 66-72.

Clements, P., Kazman, R., Klein, M. (2002). Evaluating Software Architectures: Methods and Case Studies. Addison Wesley.

Conchuir, E., Ågerfalk, P., Olsson, H., Fitzgerald, B. (2009). Global Software Development: Where are the benefits? Comm. of the ACM. 52, 8 pp.127-131.

Conchuir, E., Holmström, H., Ågerfalk, P., Fitzgerald, B. (2006). Exploring the Assumed Benefits of Global Software Development. In Proceedings of the IEEE international Conference on Global Software Engineering (October 16 - 19, 2006). ICGSE. IEEE Computer Society, Washington, DC, 159-168.

Coplien, J.O. (1996). Software Patterns. SIGS Books, New York.

Coplien, J. O., Harrison, N. B. (2005). Organizational Patterns of Agile Software Development. Lucent Technologies, Pearson Prentice Hall.

Dearle, A. (2007). Software Deployment, Past, Present and Future, Future of Software Engineering (FOSE '07), pp. 269-284.

Doyle, C. (2007). The importance of ALM for aerospace and defence (A&D). Embedded System Engineering (ESE magazine), June 2007, Volume 15, Issue 5, pp.28-29.

Doyle, C. and Lloyd, R. (2007). Application lifecycle management in embedded systems engineering. Embedded System Engineering (ESE magazine), March 2007, Volume 15, Issue 2, pp.24-25.

[Eclipse 2010] *Eclipse web-pages*, www.eclipse.org (available 19.3.2010).

Eloranta, V-P, Leppänen, M. and Koskimies, K. (2009). Using Domain Model for Structuring Pattern Language. Nw-Mode '09.

Elssamadisy, A. (2008). Agile Adoption Patterns: a roadmap to organizational success. Addison Wesley Professional.

Evaristo, J.R., Scudder, R., Desouze, K.C. and Sato, O. (2004). A dimensional analysis of geographically distributed project teams: a case study. Journal of Engineering and Technology Management, Vol. 21, No. 3, pp. 175-189.

Farmer, M, (2004). Decision Space Infrastructure: Agile Development in a Large, Distributed Team. Proceedings of the Agile Development Conference (ADC'04).

Gamma, E., Helm, R., Johnson, R., and Vlissides J. (1995). Design Patterns: Elements of Reusable Object-Oriented Software. Addison Wesley, Professional Computing Series.

Haughey, D, (2010). http://www.projectsmart.co.uk/history-of-prince2.html (available 25.11.2010).

Harjumaa, L. (2005). A Pattern Approach to Software Inspection Process Improvement, In Software Process: Improvement and Practice. Volume 10, Issue 4, pp. 455-465, 2005.

Heindl, M., Reinisch, F. and Biffl, S. (2007). Requirements Management Infrastructures in Global Software Development - Towards Application Lifecycle Management with Role-based In-time Notification. International Conference on Global Software Engineering (ICGSE), Workshop on Tool-Supported Requirements Management in Distributed Projects (REMIDI), Munich.

Herbsleb, J.D., Grinter, R.E. (1999). Splitting the organization and integrating the code: Conway’s law revisited. Proceedings of the 1999 International Conference on Software Engineering, 16-22 May 1999, pp. 85 – 95.

Herbsleb, J.D., Moitra ,D. (2001). Guest Editors’ Introduction: Global Software Development. IEEE Software, Vol. 18, No. 2, pp. 16-20.

Herbsleb, J.D., Mockus ,A. (2003). An Empirical Study of Speed and Communication in Globally Distributed Software Development. IEEE Transactions on Software Engineering, pp. 481-494.

Hevner, A. R., March, S. T., Park, J. and Ram, S. (2004). Design Science in Information Systems Research. MIS Quarterly, 28(1), 75-105.

Holmström, H., Fitzgerald, B., Ågerfalk, P. J. & Conchúir, E. (2006). Agile Practices Reduce Distance in Global Software Development. Information Systems Management, 23(3), pp. 7-18.

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.

Hossain, E., Muhammad, A.B., Verner J. (2009). How Can Agile Practices Minimize Global Software Development Co-ordination Risks? EuroSPI 2009, Springer, Berlin Heidelberg (2009). pp. 81 – 92.

International Organization for Standardization (2001). Software engineering - Product quality - Part 1: Quality model. ISO/IEC 9126-1:2001.

Kock, N. (2008). Virtual Team Leadership and Collaborative Engineering Advancements: Contemporary Issues and Implications. Idea Group Inc (IGI), Chapter XXII.

Kobitzsch W., Rombach D., and Feldmann R. L. (2001). Outsourcing in India [software development]. IEEE Software, vol.18, no.2, pp.78-86.

Kirchner, M., and Völter, M.(2007). Guest Editors' Introduction: Software Patterns. IEEE Software, 24 (4)9, pp. 28-30.

Kääriäinen, J., Eskeli, J., Teppola, S., Välimäki, A., Tuuttila, P., Piippola, M. (2009). 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.

Kääriäinen, J., (2011). Towards an Application Lifecycle Management Framework. Espoo, Finland: Technical Research Centre of Finland, VTT Publications 759.

Kääriäinen, J. and Välimäki, A. (2008). Impact of Application Lifecycle Management – a Case Study. International Conference on Interoperability of Enterprise, Software and Applications. Berlin, German. March 25th – 28th 2008. Enterprise Interoperability III - New Challenges and Industrial Approaches. Mertins K., Ruggaber R., Popplewell K., Xu X., (Eds). Springer, pp. 55 – 67.

Lane, M. T. and Ågerfalk, P. J. (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.

Leffingwell, D. (2007). Scaling Software Agility. Addison-Wesley.

Leffingwell, D., Aalto, J-M. (2009). A Lean and Scalable Requirements Information Model for the Agile Enterprise, Modern Analyst.com, posted 2 July 2009 http://www.modernanalyst.com/Resources/Articles/tabid/115/articleType/ArticleView/articleId/982/A-Lean-and-Scalable-Requirements-Information-Model-for-the-Agile-Enterprise.aspx (available 27.11.2010).

Leppänen, M., Koskinen, J., Mikkonen, T. (2009). Discovering a pattern language for embedded machine control systems using architecture evaluation methods. SPLST'09, Tampere, Finland.

Miller, D. (2001). Choice and Application of Software Quality Model. In book: Daughtrey, T. (Ed.) Fundamental Concepts for the Software Quality Engineer. American Society for Quality.

Mintzberg, H. (1989). Mintzberg on Management: Inside Our Strange World of Organizations. Free Press, New York.

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

Mockus, A., Herbsleb, J. (2001). Challenges of global software development. Software Metrics Symposium, 2001. METRICS 2001. Proceedings. Seventh Inter-national, pp. 182-184.

Moe, N. B., Šmite, D. (2008). Understanding a Lack of Trust in Global Software Teams: A Multiple-Case Study. Software Process Improvement and Practice, Volume 13, Issue 3, pp. 217-231.

Ohno, T. (1988). Toyota Production System: Beyond Large Scale Production, Productivity Press.

Paasivaara, M., Hiort af Ornäs, N., Hynninen, P., Lassenius C., Niinimäki, T. Piri, A. (2010). Practical Guide to Managing Distributed Software Development Projects. Aalto University School of Science and Technology.

Palmer S. and Felsing J. (2002). A Practical Guide to Feature-Driven Development. Upper Saddle River, New York: Prentice-Hall.

Pesola, J., Eskeli, J., Parviainen, P., Kommeren, R., 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.

Poppendieck M. and Poppendieck T. (2003). Lean Software Development: An Agile Toolkit for Software Development Managers. Boston, Massachusetts: Addison-Wesley.

Poppendieck M. and Poppendieck T. (2006). Implementing Lean Software Development: From Concept to Cash. Addison-Wesley.

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

Richardson, I., Casey, V., Burton, J., McCaffery, F. (2010). Global software engineering: A software process approach. In I. Mistrik, J. Grundy, A. van der Hoek, and J. Whitehead, editors, Collaborative Software Engineering, Springer- Verlag/Computer Science Editorial, pp. 35–56.

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

Ryan B, Scapens RW, Theobald, M (1992) Research Method and Methodology in Finance and Accounting. Academic Press. London.

Ryder, P. (2005). The Agile Manifesto Explained. CVu - The ACCU (Association of C and C++ Users) magazine. Volume 17, No 5.

Sangwan, R., Bass, M., Mullick, N., Paulish, D.J., Kazmeier, J. (2006). Global Software Development Handbook. Auerbach Publications.

Schwaber, C. (2005). The Expanding Purview Of Software Configuration Management. Forrester Research Inc., White paper.

Schwaber, C. (2006). The Changing Face of Application Life-Cycle Management'. Forrester Research Inc., White paper.

Schwaber, K. (2004). Agile Project Management with Scrum. Microsoft Press.

Schwaber K., Beedle, M. (2002). Agile Software Development with Scrum. Prentice Hall Series on Agile Software Development, Upper Saddle River, New Jersey.

Schwaber, K. (2007). Agile The Enterprise and Scrum. Microsoft Press.

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 18.10.2007).

Shroff, G., Mehta, A., Agarwal, P. and Sinha, R. (2005). Collaborative development of business applications. International Conference on Collaborative Computing: Networking, Applications and Worksharing, pp.19-21.

[Smith 2009] http://www.best-management-practice.com/Knowledge-Centre/News/Prince2-News/?DI=616423 (available 24.1.2010).

Šmite, D., Wohlin, C., Gorschek, T., and Feldt, R. (2010). Empirical Evidence in Global Software Engineering: A Systematic Review. Empirical Software Engineering: An International Journal, Vol. 15, Nr. 1, pp. 91-118, 2010.

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

Vesiluoma, S. (2007). Knowledge Sharing Pattern Language. (Eds. Berki, E., Nummenmaa, J., Sunley, I., Ross, M. and Staples, G.). Software Quality in the Knowledge Society. The British Computer Society.

Vesiluoma, S. (2009). Understanding and supporting knowledge sharing in software engineering. PhD thesis, Publication 843, Tampere University of Technology.

Välimäki, A., Koskimies K. (2006). Mining best practices of project management as patterns in distributed software development. EuroSPI 2006, Finland, Joensuu, October 2006, EuroSPI 2006 Industrial Proceedings, pp.6.27-6.35.

Weatherall, B. (2007). Application Lifecycle Management - A Look Back, CM Journal, CM Crossroads – The configuration management community. January, http://www.cmcrossroads.com/articles/cm-journal/application-lifecycle-management-%11-a-look-back.html (available 18.10.2007).

Wiredu, G. O. (2006). A framework for the analysis of coordination in global software development. In Proceedings of the 2006 international Workshop on Global Software Development For the Practitioner (Shanghai, China, May 23 - 23, 2006). GSD '06. ACM, New York, NY, pp.38-44.

Woodward, E., Surdek, S., and Ganis, M. (2010). A Practical Guide to Distributed Scrum, IBM Press.

Yin, R. K. (2003). Case study research: Design and methods. 2nd edition. Sage, Newbury Park, CA.

Ågerfalk, P.J, Fitzgerald, B., Holmstrom, H., Lings, B., Lundell, B., and Conchuir, E.O. (2005). A framework for considering opportunities and threats in distributed software development, Proc. International Workshop on Distributed Software Development, Paris, France: Austrian Computer Society, pp. 47-61.

Appendix A: Questions of a case: GSD in the work of product managers

Intro:
The goal of this questionnaire is to find the best practices, good properties of tools and artifacts for the product manager's work in a distributed environment.

This questionnaire is based on the phases of product management work related to idea and feature management.

Would you answer the following questions related to your product area, please?

Questions:

1. Collecting and processing product ideas

- What are good properties in tools and possible problems with them?
- What are good practices in processes and possible problems with them?
- What are good properties in artifacts and possible problems with them?
- What are the requirements of working in the distributed environment?
- If there are no constraints, what kind of tools, best practices, artifacts and support for working in a distributed environment would you choose?

2. Collecting and processing business and customer requirements
< the same questions as above>

3. Feature creation
< the same questions as above>

4. Feature prioritization and selection
< the same questions as above>

5. Adding features to product roadmap
< the same questions as above>

6. From product roadmap to business plan
< the same questions as above>

Appendix B: Questions of a case: GSD project management in Agile projects

Concepts in a questionnaire:

What is meant by “previous practices” and “new practices”?

- Previous practices/procedures: the practices we had prior to 2007, i.e. before Scrum and TFS.
- New practices/procedures: the practices we are now introducing/using (Scrum/TFS)

Part I

How familiar are you with practices related to Scrum?

1=Not at all 2=I have some knowledge 3=I know the basic concepts 4=I know Scrum well 5=I know Scrum thoroughly

What is your opinion of the Product Backlog procedure?

What are the pros and cons of the procedure with regard to (project) personnel? How would you improve the current procedure?

What are the pros and cons of the procedure with regard to the (project management) process? How would you improve the current procedure?

What are the pros and cons of the procedure with regard to tools? How would you improve the current procedure?

What are the pros and cons of the procedure with regard to distributed development? How would you improve the current procedure?

How well has the prioritization of requirements and features succeeded?

What is your opinion of the Sprint Planning procedure?

< the same What - questions as above>

How well has the selection of features for a sprint succeeded?

What is your opinion of the Sprint Backlog procedure?

< the same What - questions as above>

Has the team been able to split the features under development? How has the definition of feature-related subtasks succeeded?

What is your opinion of the Sprint and Daily Scrum procedures?

< the same What - questions as above>

Has the project been able to stick to the Sprint schedule?

How well has the work estimates succeeded? What is the success based on?

What problems have occurred with respect to estimating Sprint work amounts or sticking to estimates?

Has all project personnel been available for meetings?

Has the communication of realized work amounts, plans and problems succeeded as planned?

What is your opinion of the Sprint Review procedure?

< the same What - questions as above>

Has the team been able to develop what has been intended? Have there been any shortcomings?

What is your opinion of the Scrum of Scrums procedure?

< the same What - questions as above>

Has enough information concerning site-specific Scrum meetings been communicated in the Scrum of Scrums?

Have you created supplementary practices for Distributed Scrum with respect to “standard Scrum”?

What are these supplementary practices? What are the pros and cons of the practices? How would you further improve the practices?

How would you rate the previous and current procedures? (On a 1-5 scale (1=poor 2=fair 3= adequate 4=good 5=very good))

Previous procedure (prior to Scrum and TFS):
Current procedure (with Scrum and TFS):

How would you rate the used tools? (On a 1-5 scale (1=poor 2=fair 3= adequate 4=good 5=very good))

Previous procedure (prior to Scrum and TFS):

Current procedure (with Scrum and TFS):

Part II

In the next section, the effects of Distributed Scrum and related practices in your project are assessed. In this context, Distributed Scrum refers to a procedure that has been selected in your projects.

What is meant by “previous practices” and “new practices”?

- Previous practices/procedures: the practices we had prior to 2007, i.e. before Scrum and TFS.
- New practices/procedures: the practices we are now introducing/using (Scrum/TFS)

Claim:
Distributed Scrum improves the visibility of the project status in relation to the previous procedure.
(On a 1-5 scale (1= not applicable 2=I disagree 3=I partially agree 4=I mostly agree 5=I fully agree)) Add comments to clarify your answers.

Distributed Scrum accelerates the handling of changes in relation to the previous procedure.

Distributed Scrum improves the management of features through the entire project lifecycle in relation to the previous procedure.

Distributed Scrum improves the understanding of requirements in relation to the previous procedure.

Distributed Scrum improves the team communication in relation to the previous procedure.

Distributed Scrum improves the utilization of the knowledge possessed by the entire team in relation to the previous procedure.

Distributed Scrum improves the commitment to the goals of the project in relation to the previous procedure.

Any other comments?

Global Software Development Pattern Language for Project Management

CONTENTS

1. INTRODUCTION

Global Software Development (GSD) Pattern Language for Project Management includes 18 process patterns which have been found to be important in the area of project management in GSD. The GSD Patterns are presented in Figure 1. The content of each GSD Pattern is presented in the following chapters.

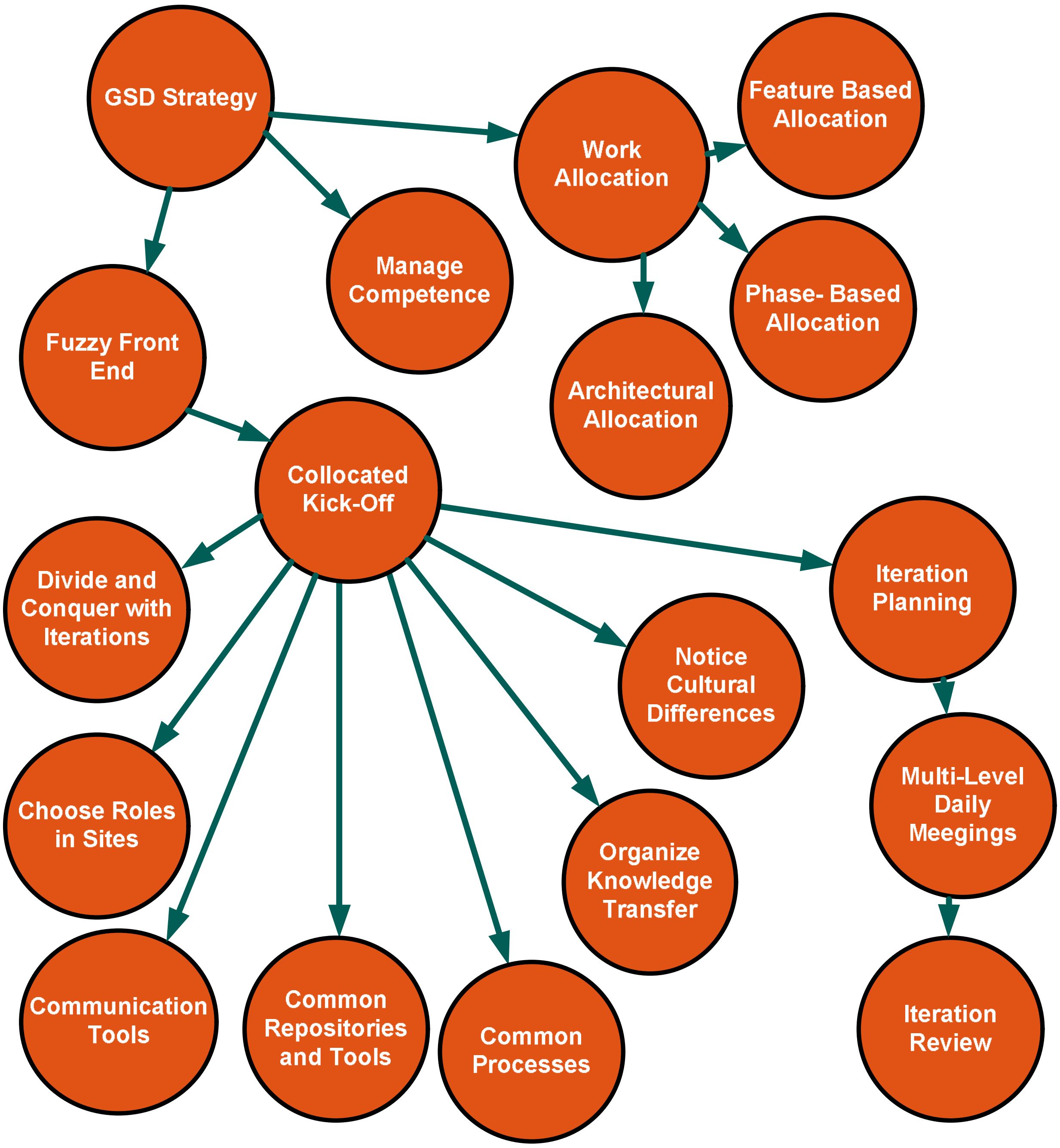


Figure 1. GSD Patterns

1.1 A Template of GSD Process Patterns

Name:	**The name of the pattern.**
Problem:	*A brief description of the problem.*
Initial context:	*The situation to which the pattern solution applies.*
Roles:	*The roles implementing the pattern.*
Forces:	*Forces that affect the situation.*
Solution:	*The required instruction to solve the problem in the context.*
Resulting Context:	*The situation/context which will result from performing the pattern solution.*

2. GSD PROCESS PATTERNS

2.1 GSD1 - GSD Strategy

Name:	**GSD Strategy**
Problem:	How to organize GSD in a company?
Initial context:	Before start of GSD in R&D department.
Roles:	Management of a company and the management of R&D department.
Forces:	• GSD benefits which can be e.g. cost savings, possibility to use talent from different countries, to use some specific knowledge, follow-the-sun development, and presence in local markets. • GSD challenges such as communication, coordination, co-operation, control breakdowns, and culture differences are easily ignored • Usually many partners are involved in GSD which requires contracts e.g. about work allocation between partners.
Solution:	Management of a company is responsible for making decisions when a GSD-based development model is taken into use. Management of a company will create a strategy to implement GSD with the management of R&D department. Implement the following actions: • List the reasons and motivation to start GSD-based development in a company. • Find out the competence and cost level of your own company, your partners and other related partners. *Manage Competence* pattern can be used with this action. • Make a SWOT (Strengths, Weaknesses, Opportunities and Threats) for a preliminary GSD strategy. • Make a short and long term plan to implement GSD strategy in your R&D, e.g. for make work allocation by *Work Allocation* pattern. • Make also a separate risk plan to manage risk with GSD. This is very important because GSD brings a lot of new risks to manage, e.g. intellectual property risk, loss of proprietary knowledge risk, product security risk, and infrastructure risk. • Plan measurements for the real costs of GSD.
Resulting Context:	Result is GSD strategy which can be used to manage different sites and e.g. high level work allocation between them.

2.2 GSD2 - Fuzzy Front End

Name:	**GSD Strategy**
Problem:	How to gather demands (i.e. ideas, needs, requirements etc.) from external and internal customers and how to form plans and change requests from these needs for GSD development?
Initial context:	Before the start of a development decisions about new projects in R&D department.
Roles:	Management of R&D department, product managers, architects and project managers.
Forces:	• It is important to gather information from both outside a company as well as inside a company. The needs of different customers are important to gather, analyze and to develop a product which will fulfill the needs of different customers.
Solution:	Director of R&D is a key person to start a new development according to company strategy and contents of Road Maps and related Business plans made by product managers. Project managers will make e.g. project plans Implement the following actions: • The needs and demands of different customers will be gathered to a common repository by product managers. For distributed demands management, it is also important to have the possibility for global access regardless of time and place as well as have the possibility to use a discussion forum inside the tool. • Product managers will go through gathered demands and they will make decisions about them with e.g. architects and project managers. A new feature or requirement will be made from a new demand if it is accepted in a decision meeting. This meeting can be distributed by communication tools. Other possible statuses for demands are e.g. cancelled or stored for later analysis. Feature size can vary from a new business idea or a new project idea to a new function in an application. • Product managers will make a Road Map for a product which will include many features. From the Road Map product managers will create Business plans which will include a group of selected features. From this Business plan one or several project plans will be made. These documents will be managed by common repositories. • Project managers will participate in making a Business plan and finally they will make a project plan to implement the group of selected features. The Project plan will include goals, resources, budget, risks analysis, work allocation etc.
Resulting Context:	R&D has Road Maps which include plans for future development. New demands and needs are gathered in a common repository from both external and internal customers around the world. Finally, project plans are made according to information and related decisions about Road Maps, Business plans, features, work allocation, risk evaluation etc.

2.3 GSD3 – Collocated Kick-Off

Name:	**Collocated Kick-Off**
Problem:	What is the goal of a GSD project and who are the members of a project? How to build trust between team members?
Initial context:	In the beginning of the project there are many open issues which need to be clarified.
Roles:	Project manager, product mangers and architects.
Forces:	• In the beginning of a project there are many open issues and often the team members do not know each other. • Also the reason why this project will be done is not clear. • Cultural differences need to be taken into consideration, too.
Solution:	Project manager is the chairman of the kick-off meeting and he/she can get help e.g. from product managers and architects. This meeting is the first step of pattern *Organize Knowledge Transfer*. Implement the following actions: • Arrange a kick-off meeting for all relevant team members to meet face to face. • Everybody presents himself / herself. • Present common goal and motivation of this project. • Present release plan which is made by *Divide and Conquer with Iterations* pattern. • Present responsibilities of each site and team members, if possible. The result of *Choose Roles in Sites* pattern can be used with this action. • Briefly introduce tools and repositories which are chosen for a project by *Communication Tools* and *Common Repositories and Tools* patterns. • Briefly present common processes in a project which are specified by *Common Processes* pattern. • Also train cultural issues for team members according *Notice Cultural Differences* pattern. • Organize leisure activities to increase trust between team members.
Resulting Context:	A common goal will be known by every relevant team member. Also common processes and tools for every site are briefly introduced. Efficient communication channels can be created between team members when they learn to know each other better. Team members are now ready to start a project.

2.4 GSD4 - Divide and Conquer with Iterations

Name:	**Divide and Conquer with Iterations**
Problem:	How to make a project plan which is manageable in a GSD project?
Initial context:	In the beginning of a project only the main features are known.
Roles:	Project manager.
Forces:	• A big project plan is difficult to manage in a GSD project. • Difficult to know the whole contents and the work estimations of a project in the start of a project. • Visibility of project status is poor in GSD. • Possible new application architecture, technologies etc. are unknown.
Solution:	Project manager will split a project plan into several iterations. Implement the following actions: • Plan several iterations to describe the project plan because iterations are easier to control and it is easier to make changes to a plan. • Develop new application architecture and module structure in the main site during first iterations, if needed. • Explore the biggest risks (e.g. new technologies) in the beginning of a project. • The length of iteration can be e.g. 2-4 weeks to improve control and visibility. Main site can have 4 weeks iteration and other sites 2 weeks to improve visibility.
Resulting Context:	Iterations improve the visibility of a project and motivation of team members in a GSD project. Iterations also make it easier to control a project when you split the whole project into many manageable parts. Iterations also provide feedback and the possibility to learn from earlier iterations. However, administration work is increased with several iterations.

2.5 GSD5 - Choose Roles in Sites

Name:	**Choose Roles in Sites**
Problem:	How to know who to contact in different sites with your questions?
Initial context:	In the beginning of the project the roles of team members are not defined.
Roles:	Project manager, site managers and other supervisors.
Forces:	• It is difficult to communicate if you do not know who to ask • GSD needs clear communication organization. • Possible turnover of team members.
Solution:	Project manager together with Site managers are key persons to choose roles of different team members in different sites.. Implement the following actions: • Project manager will have negotiations with site mangers or other supervisors about team members before final decisions. • Other needed roles will be formed in every site (e.g. Project manager, Architect, Developer, IT support, Quality assurance etc.). The needed roles in each site depend on how work is divided between sites. • E.g. usually the main site project manager is in a leading position and project managers from other sites will help to take care of the issues, tasks and responsibilities in their sites. • Publish the whole project organization with roles for every site to improve communication. • One team member can have many roles in a project. Also many persons can have the same roles. • To manage turnover of team members, deputy persons can also be chosen. Also shared code owner ship or pair programming are possible solutions for turnover of developers.
Resulting Context:	Team members who have same roles can communicate efficiently between sites because they have same responsibilities. Team members with same roles can have regular meetings to check what the results are, plans and possible problems to be solved in their responsible area. Finally, the whole project organization with roles is published in a common repository of a GSD project.

2.6 GSD6 - Communication Tools

Name:	**Communication Tools**
Problem:	How to choose communication methods and tools in GSD?
Initial context:	In the beginning of a project it is not clear which communication tools and methods should be used.
Roles:	Project manager and team members.
Forces:	• It is difficult to communicate if you do not have good communication tools which are in use with every team member. • Usually different sites have different tools and methods for communication. • Cultural differences and different native languages also affect the efficiency of communication. • For team members it is much easier to communicate with each other if they have met face to face as suggested in *Collocated Kick-Off* pattern.
Solution:	Project manager is a key person to choose communication methods and tools and how team members can use them. Of course it is important that team members can participate in decision making. Implement the following actions: • Have reliable and common communication methods and tools in every site. • Use of video conferences, web cameras, net meeting applications, chat, conference phones, Skype, mobile phones, electronic calendars, discussion tools, wiki tools etc. improves the efficiency of communication. • Use different tools at the same time as a net meeting application to show information and project data, conference phones to have good sound and chat tool to discuss in written form if there are problems to understand e.g. English language used in other sites. • Record e.g. meetings and key presentations to improve knowledge sharing. Records of meetings or presentations make it possible to listen to them once again, if needed. • Use common repositories to share artifacts, which can be implemented by *Common Repositories and Tools* pattern. • Publish availability of team members, common working times, holidays etc. to help communication and coordination of different meetings. • Also notice cultural differences which also affect communication methods and use of tools.
Resulting Context:	It costs money to establish common communication tools and common repositories for a GSD project. It saves money to use communication tools because there is no need to travel so much. Published availability information makes it easier to have net meetings and ad-hoc discussions. Communication tools changes distributed development partly back to centralized development. Finally, a project has common communication methods and tools in use.

2.7 GSD7 - Common Repositories and Tools

Name:	**Common Repositories and Tools**
Problem:	How to share different artifacts between sites efficiently?
Initial context:	In the beginning of the project it is not clear how to share documents or source code or process guidelines between team members in different sites.
Roles:	Project manager and team members.
Forces:	• Separate files are difficult to manage and synchronize between many sites. • Chosen tools should be efficient, easy to use, and integrated with each other. • To improve information security, only needed information can be visible to team members in each site.
Solution:	A project manager is a key person to organize common development, documentation and process guidelines tools and how team members can use them. Implement the following actions: • Provide a common Application Lifecycle (ALM) Management tool for o all project artifacts (e.g. Product and Sprint backlog items, source code, development documents, faults descriptions etc.), o reports, o process guidelines (workflow, guidelines), o process information (e.g. who does what and when), o traceability (e.g. which information is related with each other), o communication tools (discussion forum, chat, visualize information, notifications to users etc.). • ALM tool can be implemented as a single tool set or it can be a group of different tools which has been integrated with each other. • Use different levels (team, project, and program) reports to improve visibility of status of projects. • Effective access right control i.e. user rights methods and role-based views in use to see certain data.
Resulting Context:	ALM provides almost real-time traceability, reporting, visualization and access to needed information etc. for all users in different sites. ALM tools can include a means to support operation according to the organisation’s processes and development methods (state models, process templates, workflows). ALM tools can cost a lot of money. Time is saved when information is found more quickly. Common processes also make work more efficient. ALM also facilitated communication with communication tools. As a result, ALMbased tool is taken into use in a project. All project artifacts, which are needed, are shared by ALM among all team members.

2.8 GSD8 - Work Allocation

Name:	**Work Allocation**
Problem:	How work is divided between sites?
Initial context:	In the beginning of a project it is not clear how to divide responsibilities between sites.
Roles:	Main site manager and project manager.
Forces:	• GSD Strategy in a company has a main impact on work allocation.
Solution:	Main site manager is a key person to divide responsibilities in sites according to a company's GSD Strategy. A project manager will apply GSD Strategy in his/her project with architects. Implement the following actions: • Check competence information of team members in each site with help of site managers. • Use *Architectural Work Allocation* pattern or/and • Use *Phase- Based Work Allocation* pattern or/and • Use *Feature- Based Work Allocation* pattern or other allocation according to some other criteria. • Make a decision about division of work between sites according to a company's GSD Strategy and the above analysis.
Resulting Context:	Work is divided between sites to enable efficient development and implementation of tasks in a project according to a company's GSD Strategy.

2.9 GSD9 - Architectural Work Allocation

Name:	**Architectural Work Allocation**
Problem:	How work is divided between sites with architectural criteria?
Initial context:	In the beginning of a project it is not clear how to divide responsibilities between sites.
Roles:	Main site manager, project manager and architect.
Forces:	• One goal in GSD is to minimize communication problems between sites. Software architecture and coupling and interfaces between modules will affect the amount of communication. If coupling between modules is tight it is difficult to divide work between different sites. Also difficult and complicated interfaces between modules will make it difficult to divide work between different sites. • According to Conway's law, "Organizations which design systems are constrained to produce designs which are copies of the communication structures of these organizations." That is one reason to think carefully about how to make work divisions.
Solution:	Main site manager is a key person to divide responsibilities in sites according to a company's GSD strategy. A project manager will apply GSD strategy in his/her project with architects. Implement the following actions: • Check architectural analysis of your product and plan which site will be responsible to maintain and increase knowledge in some architectural area. Architectural area can also be a whole subsystem or part of a subsystem. • Make a decision about division of work between sites about architectural work allocation according to GSD Strategy and above analysis.
Resulting Context:	Work is divided between sites according to architectural criteria to enable efficient development and implementation of tasks in a project.

2.10 GSD10 - Phase- Based Work Allocation

Name:	**Phase- Based Work Allocation**
Problem:	How work is divided between sites with phase-based criteria?
Initial context:	In the beginning of a project it is not clear how to divide responsibilities between sites.
Roles:	Main site manager and project manager.
Forces:	• A company’s GSD Strategy can give a certain main phase responsibility or responsibilities to a certain site. • GSD Strategy describes which know-how is needed and will be created or maintained in each site and team members now and in future. • E.g. a system test laboratory can be very expensive to create and this responsibility might be nominated to a certain site.
Solution:	Main site manager is a key person to divide responsibilities in sites according to a company’s GSD Strategy. A project manager will apply GSD Strategy to his/her project. Implement the following actions: • Check the GSD Strategy how phase- based work allocation will be made. • Check also which site is possibly responsible for maintaining and increasing knowledge in some phase-based area e.g. testing or requirements engineering in a certain product area. • Make a decision about division of work between sites about phase-based work allocation according to GSD Strategy and above analysis.
Resulting Context:	Work is divided between sites according to phase -based criteria to enable efficient development and implementation of tasks in a project.

2.11 GSD11 - Feature- Based Work Allocation

Name:	**Feature- Based Work Allocation**
Problem:	How work is divided between sites with feature-based criteria?
Initial context:	In the beginning of a project it is not clear how to divide responsibilities between sites.
Roles:	Main site manager and project manager.
Forces:	• A company's GSD Strategy can give certain responsibilities to different sites. • When a system level feature affects the whole product, e.g.a security-based feature will be developed. Then there is the need to form a group of team members from different sites to implement this system level feature.
Solution:	Main site manager is a key person to divide responsibilities in sites according to a company's GSD Strategy. A project manager will apply GSD Strategy to his/her project. Implement the following actions: • Check the GSD Strategy how feature- based work allocation strategy has been described. • Form a group of members from different sites to realize the features, if needed.
Resulting Context:	Work is divided between sites according to feature-based criteria to enable efficient development and implementation of tasks in a project.

2.12 GSD12 - Common Processes

Name:	**Common Processes**
Problem:	How to implement an efficient process in a GSD project?
Initial context:	In the beginning of a project there are often different processes in every site.
Roles:	Quality manager and project manager.
Forces:	• Different processes and templates at different sites make communication inefficient. • Different sites often have different processes. • Processes often include local optimizations.
Solution:	Quality manager is a key person to specify global processes for a company or R&D department. Project manager is a key person to choose global processes for a project. Implement the following actions: • Choose common upper level processes for the project, e.g. gateway model, project management main practices (Iteration Planning, Iteration Review, etc.), main templates etc. • Allow local processes if they do not cause problems with upper level processes, e.g. project management site specific implementation and development methods etc. • Tune the processes at the end of iteration to make project work more efficient, e.g. with iteration retrospective which is used to check what went well and what to improve in processes in the next iteration.
Resulting Context:	Upper level processes are common and thus make the whole project work more efficiently. However, lower level processes might have been implemented differently in different sites to reduce cultural or language difficulties. Every change in common processes needs to be communicated and taken into use in different sites which increase work amount. That is why big changes to common processes are not so usual. Finally, common processes are taken into use in each site, at least upper level.

2.13 GSD13 - Iteration Planning

Name:	**Iteration Planning**
Problem:	How to choose which features will be implemented in a GSD project?
Initial context:	Team members are starting a new iteration to produce new features for a project.
Roles:	Project manager, product manager and team members.
Forces:	• There are many features to be implemented. • Different team members have a different view about which are the most important features. • If team members do not meet each other face to face at all, it might decrease trust between team members. • Dividing work in a GSD project is challenging because features have dependencies on each other.
Solution:	A project manager is a key person to tell what the most important features to be implemented are after he/she has discussed with a product manager. Implement the following actions: • Project manager will present prioritized features and other tasks which are made by a project manager or team members. • Team members will participate in a planning meeting either personally or by communication tools. *Communication Tools* pattern can be used with this action. • The team members will estimate amount of work for features and tasks and add more detailed level tasks. If needed, more detailed discussion can be arranged in sites in participants' mother language. • In the end of a planning meeting, a list of selected features and tasks is created and is visible by the common repository. *Common Repositories and Tools* pattern can be used with this action.
Resulting Context:	The prioritized feature and task list will help team members to concentrate on the most important features and tasks. After planning meeting, there will be estimated features and tasks which will be implemented during iteration. For community spirit and cooperation, it is important that team members meet each other as often as possible to maintain communication and confidence. Finally, the selected features and tasks are visible by common repository for GSD team members.

2.14 GSD14 - Multi-Level Daily Meetings

Name:	**Multi-Level Daily Meetings**
Problem:	How to share project status information between team members in each site? Lack of trust and long feedback loops.
Initial context:	Project groups in different sites have started to implement iteration and there is a need to communicate between groups.
Roles:	Project manager and team members.
Forces:	• Size of group should be from 3 to 7 to be efficient. • If everybody does not speak the same language then separate meetings can be a more efficient way of working.
Solution:	Project manager and team members will participate in meetings in different sites. Implement the following actions: • Organize daily meetings in every site if a common meeting with all team members is not efficient enough. • Organize another daily or weekly meeting between project managers from each site to change information about the results of team members' meeting. • Use communication tools and common repositories in meetings. *Communication Tools* and *Common Repositories and Tools* patterns can be used with this action. • With foreigners, written logs can be one solution (e.g. chat logs or common documents) to ensure that communication messages are understood correctly in every site.
Resulting Context:	Communication between sites is improved and language problems have been tried to be solved. Daily or weekly meetings between team members and project managers will increase trust and shorten long feedback loops.

2.15 GSD15 - Iteration Review

Name:	**Iteration Review**
Problem:	How to check status of a GSD project and give and get feedback frequently?
Initial context:	Team members have implemented tasks and features during iteration.
Roles:	Project manager and team members.
Forces:	• It is difficult to get everyone to participate in the meeting. • There is a need to check status and give and get feedback about the results and solutions made.
Solution:	Project manager will organize an iteration review meeting but team members will show the current status of their work. Implement the following actions: • Check the project status by a demo and present results to all relevant members and stakeholders from different sites by team members. • Gather comments and change requests for further measures for both product and process. • Make frequent deliveries to improve visibility of the status of the product. • If all team members are not able to attend the meeting it can also be arranged using communication tools. This way information can be shared with team members communicating over a remote connection. *Communication Tools* pattern can be used with this action.
Resulting Context:	This solution brings visibility to work done and provides feedback of what has gone well and what could be done better in a GSD project. Team members focus on their tasks because they have to present their work in the meeting.

2.16 GSD16 - Organize Knowledge Transfer

Name:	**Organize Knowledge Transfer**
Problem:	How to transfer a huge amount of knowledge for team members in a GSD project?
Initial context:	GSD project often has new or inexperienced team members who need to find out a lot of information about domain and the project itself.
Roles:	Project manager and team members.
Forces:	• It is difficult to find needed information • When there is a turnover of workers, a large portion of knowledge is lost from that site. • If work moves along with knowledge transfer from the main site to the other sites it reduces motivation for knowledge transfer work. • The need for knowledge transfer depends on how great portion of responsibilities is transferred to the other sites.
Solution:	Project manager will organize tools to manage information and take care that the information will be updated to information management tools. Implement the following actions: • Make sure that there is knowledge repository of project's domain available for team members. • Train the earlier version of product and get members also to use it, if possible. • Also earlier customer documentation and demo will be presented in some cases. • Specification with e.g. use cases will be presented in iteration planning meetings or separate meetings before development work. • The team members' network will be utilized by trying to find solutions for problems. • Frequent or longer visits to enhance knowledge transfer and to make sure that there are good communication tools and channels between team members.
Resulting Context:	Knowledge transfer is possible through documents and especially through people. Knowledge transfer and providing support increase work at main site and reduces time that can be used for actual development. New team members can participate faster in project work and improve the productivity of a GSD project.

2.17 GSD17 - Manage Competence

Name:	**Manage Competence**
Problem:	How to know what the competence of each team member is in a GSD project?
Initial context:	In the beginning of a project the competence of team members is not known.
Roles:	Project manager and team members.
Forces:	• Difficult to know competence level of a team member. • Competence estimation is sensitive information about a team member.
Solution:	Project manager will organize tools to manage competence information if organization does not offer support for that. Implement the following actions: • Create a competence database for gathering information of members' competence levels at different sites. At least site manager and/or project manager knows the competence of team members. • Define the areas of competence you want to monitor. • Define competence levels and criteria for them. • Ask site managers and or project managers to gather information about their team members. • Go through competence information with each team member yearly and make a training plan for obtaining the wanted competence levels.
Resulting Context:	Project manager knows competence levels of team members. After going through competence levels of each team member with a project manager or a site manager everyone will be assigned to a certain role with specific responsibilities. If needed, everyone will also have their own personal training plan.

2.18 GSD18 - Notice Cultural Differences

Name:	**Notice Cultural Differences**
Problem:	How to notice cultural differences and increase cultural knowledge in team members in a GSD project?
Initial context:	In the beginning of a project, a nation’s cultural differences are not understood.
Roles:	Project manager and team members.
Forces:	• Every nation has its own culture and culture is very difficult to change. • There may be a lot of difficulties and inefficiency if cultures are very different from each other and the cultural differences are not understood.
Solution:	Every team member should know the main differences of different sites’ cultures. Implement the following actions: • Raise the awareness of different national culture for team members. • Use site visits, ambassadors, and liaisons to improve communication and to increase the amount of cultural knowledge in project participants, if possible. • Notice cultural differences when you are applying different patterns such as *GSD Strategy*, *Work Allocation*, *Common Processes*, *Communication Tools* and *Common Repositories and Tools* etc. • Allow local approaches in processes, tools, meeting methods etc. to decrease problems with cultural differences, if they do not disturb common processes etc.
Resulting Context:	Understanding of your team members’ culture is increased. Many problems which are related to cultural differences have been avoided. Team sprit is improved because differences between nations are easier to understand. Improved trust between team members when they understand each other better.

Paper I

Requirements Management Practices as Patterns for Distributed Product Management

In: Munch, Abrahamsson,. (ed.),
Product-Focused Software Process Improvement, PROFES 2007, Riga, Latvia, July 2007, LNCS 4589, Springer (2007), pp. 188-200.

Paper II

Patterns for Distributed Scrum – a Case Study

In: 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 Approached. Springer (2008), pp. 85 – 97.

Paper III

Get a grip on your distributed software development with Application Lifecycle Management

In: International Journal of Computer Applications in Technology (IJCAT), InderScience Publishers, Vol. 40, No. 3 (2011), pp. 181-190.

Paper IV

Scenario-Based Assessment of Process Pattern Languages

In: Proceedings of Profes 2009, June 2009,
(Eds). Springer (2009), pp. 246 – 260.

Paper V

Global Software Development Patterns for Project Management

In: R.V. O'Connor et al. (Eds.):
Communications in Computer and Information Science. Vol. 42, EuroSPI 2009, Springer, Berlin - Heidelberg (2009), pp. 137 – 148.

Paper VI

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

In: R.V. O'Connor et al. (Eds.): Communications in Computer and Information Science. Vol. 42, EuroSPI 2009, Springer, Berlin - Heidelberg (2009), pp. 149 – 160.
Reprinted with permission from the publisher.