

# Methodological Support for Task Coordination in Global Software Engineering Projects at Product Software Companies

*Master Thesis*

**Author:**

Carolus Borromeus Widiyatmoko  
5590329

**Supervisors:**

Dr. Sietse J. Overbeek  
Prof. dr. Sjaak Brinkkemper

Department of Informatics and Computing Science  
Faculty of Science



**Universiteit Utrecht**

July 14, 2017



# Abstract

Demand in performing software engineering projects globally by software companies continuously grow. Companies start to acquire other companies, build remote offices, or create partnerships with other companies from other countries. By distributing their software development activities such as development and testing processes to dispersed locations, these companies aim to reduce development costs, get closer to market proximity, or recruit young talented resources. However, they also face some challenges where cultural, knowledge, and technology diversities become the barriers in coordinating tasks among distributed resources. Consequently, a well-managed coordination mechanism is required to build productive communication, better work synchronization, a same level of understanding in customer requirements and system design, which eventually, enhance project performance.

This research project proposes a method that aims to support product software companies in coordinating tasks among globally dispersed teams in software engineering projects. This is done by answering the main research question: "How can we provide methodological support for the improvement of task coordination in global software engineering projects in a product software organization?"

Following the design science framework by beginning with a problem investigation throughout a literature study and various semi-structured interviews, the "GSE Task Coordination Method" is developed through the Method Association approach. The heart of this method is the task coordination mechanism itself supported by the organizational support, and the tools support that should anticipate the organizational aspects and the GSE challenges, allowing each company to make decisions to determine different mechanisms according to its situational factors.

Five iterative in-depth expert interview sessions involving both scientific and practicing experts demonstrated that the GSE Task Coordination Method embraces both theoretical and practical aspects, which can be simply utilized by product software companies. Throughout the validation phase, some improvements were suggested and applied shown by the evolution of the method.

As the conclusion, it can be affirmed that the GSE Task Coordination Method can support management board, line managers, and team members in coordinating between the teams in globally distributed locations. However, the effectiveness of the method to support team performance enhancements in a measured way has not been fulfilled due to time constraints.

*Keywords: Task coordination, Task dependencies, Global software engineering, Product software organization, Software process improvement*



# ACKNOWLEDGEMENT

This thesis is the final result of my study in Business Informatics at Utrecht University which would not be completed without the help of many people and organizations throughout the process. I want to take this opportunity to thank them for their assistance and support.

First of all, I would like to thank Him who always stands by me and guides me with His light. I would like to thank everyone who helped me during my research. In particular, my supervisors who have supported me, and provided plenty of details and high-quality feedback: Dr. Sietse Overbeek and Prof. Dr. Sjaak Brinkkemper. I also would like to thank my colleagues, the students at Master in Business Informatics for their friendliness. A special thanks goes to Telkom Indonesia for the financial support. Also, a special gratitude I give to the participants and the companies who have contributed to this research by giving their time for interviews, providing feedback and critics, or have helped in any other way.

Last but not least, I would like to thank for the abundantly support, love, love, and love from my beloved one, Yunita Anastasia, and our "schatjes", Hayden and Nathan.

Thank you all.



# TABLE OF CONTENTS

Abstract .....	iii
Acknowledgement .....	v
Table of Contents .....	vii
Table of Figures .....	ix
Table of Tables .....	xi
<b>Part One: Research Outline .....</b>	<b>1</b>
Chapter 1 Introduction.....	3
1.1 Research Background.....	3
1.2 Problem Statement .....	4
1.3 Research Objectives .....	5
1.4 Research Questions .....	5
1.5 Research Contribution .....	6
1.6 Report Outline .....	7
Chapter 2 Research Method.....	9
2.1 Design Cycle: Research Framework.....	9
2.2 Research Approaches.....	10
2.3 Plan Validity .....	16
2.4 Research Execution .....	18
<b>Part Two State of The Art .....</b>	<b>19</b>
Chapter 3 Literature Study.....	21
3.1 Global Software Engineering.....	21
3.2 Benefits and Risks of GSE.....	22
3.3 Task Coordination Approaches to Overcome GSE Challenges.....	24
3.4 The Literature Study's Summary.....	38
Chapter 4 Coordination Practices at Product Software Companies.....	39
4.1 Product Software Company .....	39
4.2 Challenges and Practices at Product Software Companies.....	43
4.3 The Interviews' Summary .....	61
Chapter 5 Summary of State of the Art .....	63

5.1	Interdependencies in GSE .....	63
5.2	Situational Factors of Task Coordination .....	64
5.3	Task Coordination Approaches: Communication, Control, and Knowledge Sharing.....	69
5.4	Involved Tools in Task Coordination.....	75
5.5	Organizational Support for Task Coordination.....	76
<b>Part Three Solution Design and Validation .....</b>		<b>79</b>
Chapter 6 Method Design: Towards Methodological Support for Task Coordination...		81
6.1	Method Construction Preparation.....	81
6.2	Constructing Task Coordination Methodological Support.....	84
6.3	Primary Conclusion .....	91
Chapter 7 Method Validation: Evaluation and Evolution .....		93
7.1	Global Task Coordination Method Evaluation Scenario.....	93
7.2	Evaluation Results.....	95
<b>Part Four Closing.....</b>		<b>105</b>
Chapter 8 Discussion .....		107
8.1	Evaluation Summary: The Synthesized Findings.....	107
8.2	The Final Global Task Coordination Method.....	110
8.3	Limitations .....	110
Chapter 9 Conclusions.....		113
9.1	Results	113
9.2	Future Research .....	115
References.....		117
Appendices .....		124
Appendix A. Interview Protocol.....		124
Appendix B. Systematic Literature Review .....		128
Appendix C. Company Profiles.....		131
Appendix D. Appendix ICoding scheme .....		133
Appendix E. Method Association.....		134
Appendix F. Method Base .....		137
Appendix G. PDD Documentation.....		141
Appendix H. PDD Notation.....		151
Appendix I. Expert Opinion Interview Protocol.....		152



# TABLE OF FIGURES

Figure 1-1 Examples of Collaboration Model (Šmite, 2007, pp.57-68) .....	3
Figure 2-1. Design science in engineering cycle (Wieringa, 2014) .....	9
Figure 2-2 Design cycle adaptation.....	10
Figure 2-3 Number of articles found in DBLP .....	12
Figure 2-4 An example of a process delivery diagram.....	16
Figure 2-5 Method comparison approach in PDD.....	17
Figure 2-6 Project phasing in PDD .....	18
Figure 3-1 Concepts found during problem investigation.....	21
Figure 3-2 PDD of Scrum process model.....	28
Figure 3-3 PDD of Scrum product backlog grooming session .....	29
Figure 3-4 PDD of Sprint planning meeting.....	29
Figure 3-5 PDD of daily stand-up meeting.....	30
Figure 3-6 Process mapping of PMBOK® Guide and GSD practices.....	31
Figure 3-7 Knowledge process model to support task coordination .....	32
Figure 3-8 Global Teaming process area (Richardson et al., 2012, p.1184) .....	35
Figure 3-9 Global Canvas (Smirnova et al., 2014, p.88) .....	37
Figure 3-10 PDD of Global Canvas processes .....	37
Figure 4-1. Software classification (Xu & Brinkkemper, 2007) .....	40
Figure 4-2. Reference framework for software product management .....	41
Figure 4-3 PDD of task coordination approach by AlphaSoft .....	46
Figure 4-4 AlphaSoft’s Scrum Board.....	47
Figure 4-5 AlphaSoft’s burn down chart .....	48
Figure 4-6 PDD of product engineering processes at BetaSoft.....	49
Figure 4-7 Functional diagram of two coordination areas in BetaSoft .....	51
Figure 4-8 Coordination practices in GammaSoft .....	52
Figure 4-9 Segregation of tasks in GammaSoft .....	54
Figure 4-10 Service Coordinator as a communication broker.....	55
Figure 4-11 Team allocation in DeltaSoft.....	56
Figure 4-12 Task allocation for the Scrum Team.....	57
Figure 4-13 PDD of software engineering processes at DeltaSoft.....	57
Figure 4-14 PDD of designing realization plan at DeltaSoft .....	58
Figure 5-1 Organization distribution and their temporal dispersion distance.....	66
Figure 5-2 GSE challenges causal model .....	69
Figure 5-3 Organization design in communication .....	70
Figure 5-4 Knowledge coordination mechanisms.....	75
Figure 6-1 Framework for coordination mechanisms in GSE.....	85
Figure 6-2 Picture Diagram of Task Coordination Method .....	86

Figure 6-3 High-level PDD of GSE Task Coordination Method .....	86
Figure 6-4 Activity Group 1: Identify enterprise strategy .....	86
Figure 6-5 Activity Group 2: Recognizing organization profile .....	87
Figure 6-6 Activity group 3: Identifying task coordination support.....	88
Figure 6-7 Activity group 4: Determining appropriate coordination mechanisms.....	88
Figure 6-8 Select communication mechanism .....	89
Figure 6-9 Select control mechanism .....	90
Figure 6-10 Select knowledge sharing mechanism.....	90
Figure 6-11 Activity Group 5: Continuous improvement .....	91
Figure 7-1 Method acceptance variables .....	95
Figure 7-2 Merging “Knowledge Sharing” concept to “Communication”.....	96
Figure 7-3 Adjusting communication mechanisms.....	97
Figure 7-4 Elaborate other stakeholders .....	97
Figure 7-5 Reducing ”perceived distance” .....	97
Figure 7-6 Adjustment for the vertical and horizontal cultural issues .....	98
Figure 7-7 Adjusting control mechanisms.....	99
Figure 7-8 Improving task coordination preparation step .....	101
Figure 8-1 The Final Global Task Coordination Framework.....	108
Figure 9-1 PDD of the high level GTC Task Coordination Method .....	137
Figure 9-2 PDD of ”Perform Routine Activities” .....	138
Figure 9-3 PDD of ”Determine Control Mechanism” .....	138
Figure 9-4 PDD of ”Determine communication mechanism” .....	139
Figure 9-5 PDD of GSE task coordination Method (Main Method’s Final Version) .....	140

# TABLE OF TABLES

Table 2-1. Literature Sources.....	13
Table 2-2 Concepts and authors mapping .....	13
Table 2-3 An illustration of a concepts matrix.....	14
Table 3-1. Dispersion factors in Global Software Engineering .....	22
Table 3-2 Risks in global software engineering projects .....	23
Table 3-3 Task coordination approaches .....	24
Table 3-4 Practices in Distributed XP .....	31
Table 3-5 Best practices.....	34
Table 3-6 Global Canvas Elements.....	36
Table 4-1 Types of deliverables of Scrum processes in BetaSoft .....	50
Table 4-2 Task Coordination Practices by the Participating Companies .....	62
Table 5-1 Knowledge coordination mechanisms by Kotlarsky et al. (2008).....	73
Table 5-2 Tools adopted to support coordination in GSE .....	76
Table 5-3 Roles and their job functions related to task coordination in GSE .....	77
Table 6-1 Organization of the Situational Factors.....	82
Table 6-2 Coordination Mechanisms Profiles .....	83
Table 6-3 Task Coordination Experience Level.....	83
Table 6-4 Activity Group .....	84
Table 6-5 Method Association (Example) .....	85
Table 7-1 Participating experts .....	93
Table 7-2 Evaluation Cycles .....	94
Table 8-1 Method Evolution Summary .....	110
Table 9-1. Selected papers .....	128
Table 9-2 Task Coordination Concept Matrix .....	130
Table 9-3 Association Matrix for the Activities .....	134



# **PART ONE: RESEARCH OUTLINE**

**Chapter 1. Introduction**

**Chapter 2. Research Method**



# Chapter 1 INTRODUCTION

This chapter presents the problems, goals, research questions, the expected scientific contributions, and the structure of this document.

## 1.1 Research Background

In the last decade, global software engineering (GSE) has become a common practice in software development projects in many companies. Many organizations modularize software development projects and locate work packages to remote development facilities (e.g. creating development business units or acquiring software companies in other countries) or with outsourcing (Carmel & Agarwal, 2001; Herbsleb & Moitra, 2001). Engineers from other countries which have different cultures, geographic locations and time zones are involved in various stages of the software development life cycle (Olsson, Conchúir, Ågerfalk, & Fitzgerald, 2006). Šmite (2007) finds different variants of collaboration in global software engineering in which organizations share part of the product development life cycle among partners that are off-shored (Figure 1-1). As a consequence of this work division, well-managed coordination is needed.

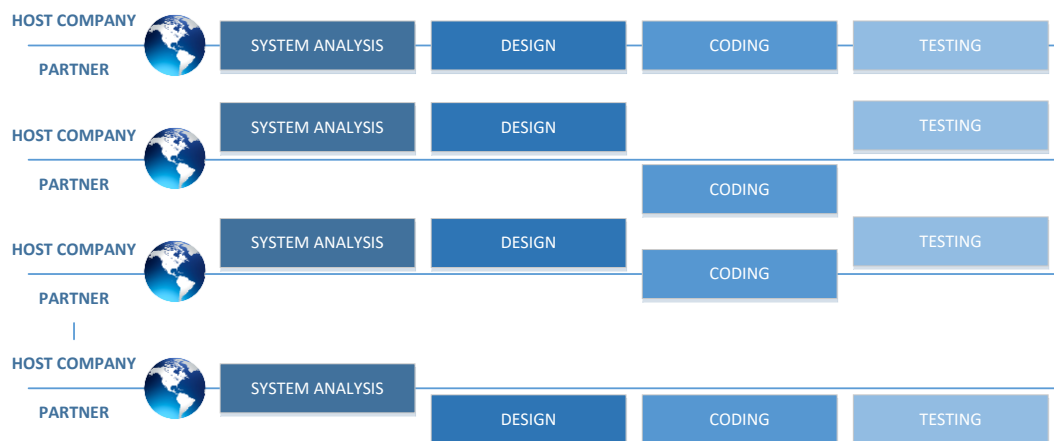


Figure 1-1 Examples of Collaboration Model (Šmite, 2007, pp.57-68)

There is an increasing amount of research in product software organizations where the software market is shifting from customized (customer-based request) software into standard software. A product software organization develops and sells mainly software as their products for a target market without customer specific modifications (Vähäniitty, 2006). Product software has a larger scale and broader target market compared to customer-based request

software. These characteristics need more complicated, expensive and slow development process to build product software. Thus, the more work units need to be done, the more resources and skills required. The lower salary scale for engineers in developing countries such as India, Malaysia, China, and Eastern Europe which offer large and highly-skilled resource pools will significantly reduce development costs (Ågerfalk, Fitzgerald, Olsson, & Conchúir, 2008). The availability of a competitive and talented resource pool becomes the main benefit for product software companies in building large scale products. For that reason, many software companies start to engage strategic partnerships with other companies (Arora & Gambardella, 2004; Bosch & Bosch-Sijtsema, 2010b). Recently, 82% of US companies employed offshore vendors to reduce their development cost (Klubnikin, 2016). Other concerns such as reduction of time-to-market, cheaper development costs, investment requirements by stakeholders, 24/7 development process, and business-to-customer proximity are the benefits that companies want to gain (Conchúir, Ågerfalk, Olsson, & Fitzgerald, 2009; Herbsleb, 2007; Setamanit, Wakeland, & Raffo, 2006).

As the software production activities are intensively interactive and complicated, the ability to communicate purposes and manage task dependencies is determinant for the organization performance. Furthermore, when the tasks become extensive and scattered, the interdependencies among tasks and teams become more complex that grows more difficulties and importance in the coordination practices compared to organizing tasks in collocated environment (Nguyen-Duc, Cruzes, & Conradi, 2012). Coordination is defined as “integration or linking together different parts of an organization to accomplish a collective set of tasks” (Van De Ven, Delbecq, & Koenig Jr., 1976, p.322). In software engineering, it can be perceived as an effort of integrating resources who are working on different tasks in a software development project. The resources should have a shared vision and agreement to a common definition of what they are building. The effort also covers managing task dependencies, to make sure that the tasks fit together and task hand-off is done without a hitch (Kraut & Streeter, 1995). Better task coordination is required to build productive communication, better work synchronization, a same level of understanding in customer requirements and system design, which are in the end, enhancing project performance (Ancona & Caldwell, 1992; Espinosa, Nan, & Carmel, 2007; Jain & Suman, 2015).

## 1.2 Problem Statement

As mentioned before, due to the increase in the intensity of product software development activities, companies are encouraged to partake the process of development to remote sites or other companies in different countries. The vast amount of related research demonstrates that task coordination in distributed collaborative software engineering is of interest for the last decades (Espinosa & Carmel, 2004; Espinosa, Slaughter, Kraut, & Herbsleb, 2007; Mak & Kruchten, 2006; Nguyen-Duc, Cruzes, & Conradi, 2015). These authors have aimed to detect challenges and risks in global software development and approaches to address distributed collaboration issues by focusing on particular aspects such as enhancing communication in distributed teams to reduce organizational silos (Olsson, Fitzgerald, Ågerfalk, & Conchúir, 2006). On the other hand, each global software engineering technique and framework has its situational demands such as the application of particular software development practices, cultural differences, fear and distrust between employees at remote sites, and the needs of knowledge development throughout the project (Jalali & Wohlin, 2010; Kotlarsky, van Fenema, & Willcocks, 2008; Piri, Niinimäki, & Lassenius, 2012; van Marrewijk, 2010). To the best of



our knowledge, there is no research which proposes methodological support that integrates both processes and artifacts to assist product software companies in understanding the situations and criteria in coordinating tasks among software development units.

Therefore, instead of competing with those existing approaches, this Master's thesis will present a methodological support to complement those studies. This raises a question on how to provide this methodological support by considering best practices and situational factors identified from what have been studied and current practices by product software companies. The purpose of this methodological support is to harmonize the current approach, provide the abstract level of definition, and help product software companies in applying the method based on their specific needs of situations (Pardo, Pino, García, Piattini, & Baldassarre, 2012).

### 1.3 Research Objectives

Two aspects of task coordination are at the core of this research, which are: the knowledge and practical aspects. So, the targeted objectives of this thesis are defined as below:

**To present methodological support for task coordination in product software companies in a global software engineering context:**

**RO1. *Knowledge aspect:* to assist organizations in understanding the specific situation and criteria that affect task coordination among development units,**

**RO2. *Practical aspect:* can contribute to the improvement of software development projects execution by improving task coordination among development units.**

### 1.4 Research Questions

Consequently, to achieve these goals, we state our main research question (MRQ) as follows:

**MRQ: “How can we provide methodological support for the improvement of task coordination in global software engineering projects in a product software organization?”**

To address the main research question, we also consider several sub-questions (SQ) as our guidelines.

**SQ1: What are the current task coordination challenges in global software engineering?**

The first phase of this thesis is framed to the current issues and practices of task coordination in global software engineering to answer our first SQ. Through this question, we provide the foundation of knowledge on the current problems and the approaches performed by organizations in synchronizing tasks among dispersed resources. We will conduct a systematic literature review to identify key factors and research artifacts of task coordination in global software engineering projects which have been studied and proposed by researchers (Section 2.2.1). Nonetheless, we will also look at some evidence or artifacts from a practical point of view. We plan to conduct interviews to elicit the challenges and approaches emerge in daily practices at several product software organizations (Section 2.2.2).

**SQ2: What are the current practices performed by product software companies in executing global software engineering projects?**

After understanding task coordination challenges and practices, we can move on to identifying the method fragments and the situational background of the existing approaches found in the literature and interviews. We use a meta-modelling technique to specify and visualize the processes, deliverables, and tool (Section 2.2.4). Meta-modelling technique is essential in a comparative review of methods and a development of situational methods (Brinkkemper, 1996). Further, the SQ2's answer will be used in the development and enrichment of our task coordination reference method.

**SQ3: What method can be designed to facilitate companies for coordinating tasks in global software engineering projects?**

To answer SQ3, we aim to develop a method based on the foundation of knowledge we gained from SQ1 and the approaches represented in SQ2. We will use method engineering approach to build a situational method that can be used by companies as a reference in coordinating task when performing software engineering globally (Section 2.2.4). We use this approach based on the understanding that no method can fit all the existing problems and engineering contexts. The complexity of each project, as well as the situational factors, brings the variabilities in the way of a project should be accomplished (van de Weerd & Brinkkemper, 2009). In the end, our reference method will be built as an optimized method which is constructed from the fragments of existing established approaches or practices. To build a reference method, identical processes and deliverables should be identified to create a specific route based on activity group that reflect their commonalities (Luinenburg, Jansen, Souer, van de Weerd, & Brinkkemper, 2008; van de Weerd, Brinkkemper, Souer, & Versendaal, 2006). Thus, to build our reference method, we need to expose and materialize the activities which specify what work to be done, deliverables or work units related to the activities, and the contexts on what these activities are performed from the information gathered in SQ1 and SQ2.

**SQ4: How to improve the developed method in task coordination after validation by considering its benefits and drawbacks?**

To validate our proposed method, we will conduct case studies to gather experts' opinions through interviews (Section 2.2.3). The experts are practitioners from product software companies who are experienced in performing software engineering projects globally. The feedback from the experts is used to assess the applicability, benefits, and drawbacks of our proposed method. Afterward, their feedback will be adopted to refine our method.

## 1.5 Research Contribution

The proposed methodological support in task allocation is expected to have the following implications:

1. Scientific contribution
  - a. Develop a comprehensive understanding of existing knowledge base of task coordination methods by elaborating and connecting methods which have been

studied and approaches by organizations on how tasks are allocated in global software engineering projects.

- b. Enhance the theoretical base in the software engineering domain by adding sources of knowledge in task coordination regarding project planning and execution management.
2. Business in practice contribution
    - a. Provide organizations a reference method for coordinating tasks that can be used in specific situational projects.
    - b. Guide organizations to coordinate tasks effectively by maintaining well-managed global software development projects.

## 1.6 Report Outline

To present how this research is operationalized, results produced, and discussion as well as rationale derived from the obtained results, the chapters are organized into three main parts as follows:

**Part 1. Research Outline.** The first part contains two chapters that provide the project management and the method of this thesis. Four approaches used in research are a systematic literature review, situational method engineering approach, semi-structured interviews, and expert validation.

**Part 2. State of the Art.** The second part of this thesis report gives the results of systematic literature review and preliminary interviews. This chapter examines the state of the art of task coordination approaches based on the literature and practical approaches based on the interviews.

**Part 3. Solution Design and Validation.** The last part of this document consists of three main elements: The design solution, the solution validation, and the summary of what have been resulted. The design solution chapter elaborates the process of method design. The concepts, framework, and processes of task coordination are presented based on the findings based on the information in practice and from the literature by using method engineering principles.

Following the design solution, the validation step explains how the proposed method was validated by elaborating several interviews with experts. The findings of the validation process (e.g. benefits, drawbacks, and trade-off) are also presented. In the end, the improvement of the framework is discussed and applied based on the findings from the interviews.

Finally, this report is ended up with the conclusions and limitations of this thesis. Some future work opportunities are presented as the suggestions for research continuation in this software process improvement and product software management topics.



# Chapter 2 RESEARCH METHOD

This section describes the research methods that are used in this project. We are following design science as our research framework. Systematic literature review, interviews, case studies, comparison analysis, and situational method engineering approaches are used to support our research in performing several research tasks such as problem investigation, gaining fundamental knowledge, developing artifacts and validating our proposed solution. Following the design science framework, the research tasks are broken down into a set of practical tasks which will be explained in the research planning section at the end of this chapter.

## 2.1 Design Cycle: Research Framework

This research will be undertaken by adopting the iterative problem-solving method proposed by Wieringa (2014). Design cycle is a subset of engineering cycle which is a continuous investigation and design processes to solve a problem by creating an artifact with the structure shown in Figure 2-1.

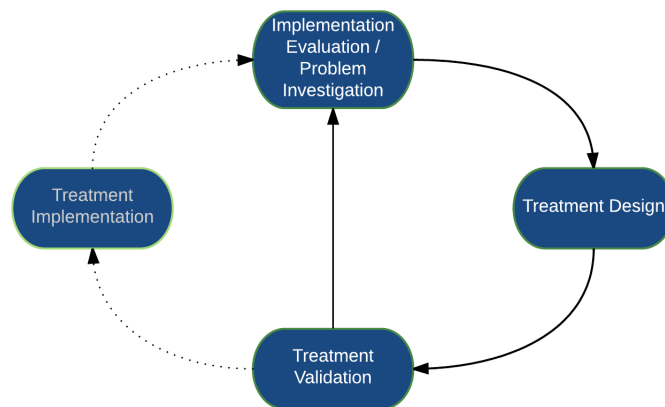


Figure 2-1. Design science in engineering cycle (Wieringa, 2014)

The engineering cycle consists of four main phases:

1. Problem investigation: investigates the stakeholders, desired goals, problems, phenomena and effects which are contributing to the goals.
2. Treatment design: specifies the requirements, identify the available treatments and design the new artifacts for the treatment.
3. Treatment validation: determines the effects, trade-offs, and requirements satisfied by the artifacts.
4. Treatment implementation: applies the artifacts in the real situation.

An artifact is something created for practical purposes. Artifacts in software engineering can be algorithms, notations, techniques, or methods. Meanwhile, the context in which the artifacts are applied can be a software engineering projects, organizations, customers, or resources (Wieringa, 2014). Therefore, the artifact produced in this thesis is a methodological support for coordinating tasks for situational purposes. Meanwhile, the project’s context is product software organizations who distribute their software project development tasks to remote sites or partner companies globally.

## 2.2 Research Approaches

In conducting our design science project, several approaches are selected and performed as can be seen in Figure 2-1. In design science, only the first three tasks of the engineering cycle are performed (Wieringa, 2014). In addition, since the engineering cycle is usually carried out in long-term research projects, our research will adapt three main parts of the engineering cycle. They are:

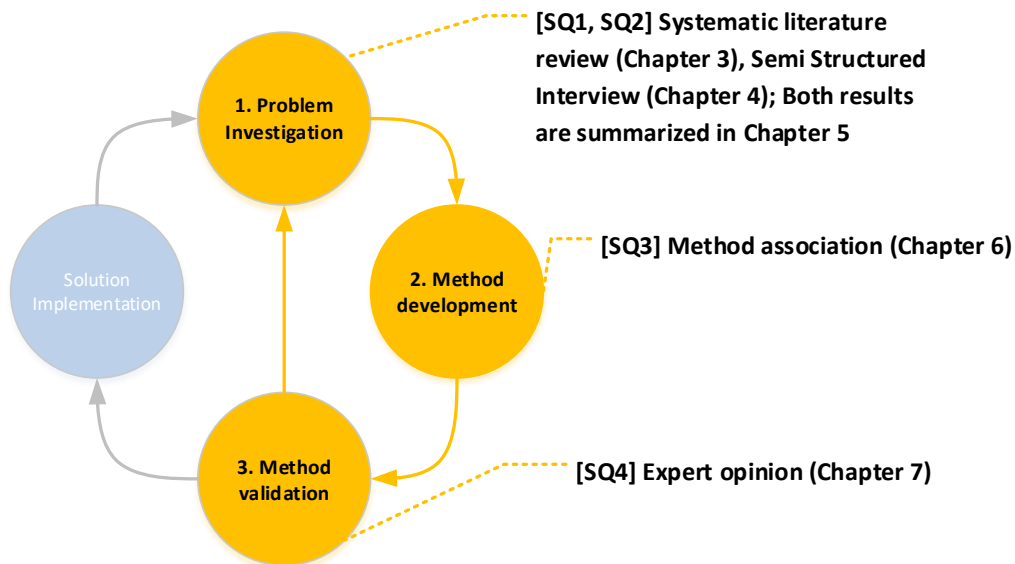


Figure 2-2 Design cycle adaptation

### 1. Problem investigation

Problem investigation is used to characterize the problem to solve. Exploratory work by conducting a backward literature review to examine on what has already researched on task coordination challenges and approaches in global software engineering (Budgen & Brereton, 2006; Webster & Watson, 2002). Semi-structured interviews with the company will also be used to get the understanding of task coordination problems and approaches in practice (Cohen & Crabtree, 2006).

A concept matrix literature review is used to examine the current task coordination approaches and propose a compilation which consists of the best fragments of existing methods to a general model of task coordination. Firstly, this framework needs a backward and forward literature approach to collect and select the source materials for the review. Secondly, concepts are determined from the literature and compiled in a concept matrix. This concept matrix will help researchers in discovering and synthesizing the key concepts of the topics (Webster & Watson, 2002).

## 2. Method design

Situational method assembly will be used for analyzing, extracting, and classifying the common fragments of the processes and concepts from established methods (Brinkkemper, Saeki, & Harmsen, 1999). This step aims to build an optimized method which can be utilized in different situations of various projects from established methods' fragments (Deneckère, Hug, Onderstal, & Brinkkemper, 2015; van de Weerd & Brinkkemper, 2009). In section 2.2.4, we will elaborate the approach in developing the task coordination reference method by using method engineering principles.

## 3. Method validation

Interviews with experts to obtain experts' opinion are conducted to validate the model to assess the benefits and drawbacks of the developed model. The feedback is used further to improve the developed model (Wieringa, 2014).

### 2.2.1 Systematic Literature Review

There were extensive studies previously performed in global software engineering, task coordination in software development, and the growth of product software development. Some previous researchers have independent topics, and some of them intersect one another. We investigated the artifacts by reviewing this contemporary literature to build a solid understanding of approaches, frameworks, or tools for task distribution in global software engineering.

#### *Systematic Literature Searching*

It is interesting to get a literature review that not only focus on top-rated literature only. Since we want to develop a methodological support, we need to get an overview of what has been previously studied by observing the literature from a higher standpoint. For that reason, we combine database searching and snowballing approaches. First, to find the primary articles, specific keywords and years limitation are used to get journals and conference proceedings on the topic of this thesis. To perform a database searching, we utilize the Computer Science Bibliography (dblp.org), Google Scholar, and ResearchGate as our search engines. Articles displayed by the search engines were selected with due regard to their scope, objectives, methods, and conclusions subjectively (Budgen & Brereton, 2006). Then, the selected articles are labeled into three main groups: global software engineering, task coordination in software engineering and software product organization.

First, to identify such studies, digital libraries such as Elsevier (ScienceDirect), ACM Digital Library and IEEE Computer Society (computer.org) and digital search engines such as DBLP are employed. Initial keywords were established, and the listed terms do not limit them since different terms with same meanings were discovered during the searching processes. The year of publications is limited from 2010 to 2016 to get a better overview of the latest researched topics.

**Keywords:** {coordination, task coordination} + {software engineering, software development, software project}, {global software engineering, global software development, distributed software engineering, distributed software development}, {software product, product software} + {company, organization}

**Year:** between 2010 to 2016

**Types:** Conferences proceedings, journals, books and book sections

As can be seen in Figure 2-3, many studies in global software engineering started to increase in the last decade (since 2006) and the figure shows that GSE is still a popular topic in this year. Based on that fact, we set 2006 as the lower limit of the searching criteria in the search engines. We only select journals, conference proceedings, theses and books from relevant areas such as Management Information Systems, Systems and Software, Information and Software Technology, Global Software Engineering, Software Product Management. We preferred articles published by ACM, Elsevier, Springer, and IEEE since they are typically publishing state-of-the-art research articles in Computer and Information Sciences and Engineering. The citations number possessed by research can be a consideration in the paper selection to be initial articles. The following criteria for initial sources were used:

- It should be a journal article, conference proceedings paper, book, or a thesis
- The paper should be written in English
- The paper should be available in digital format.

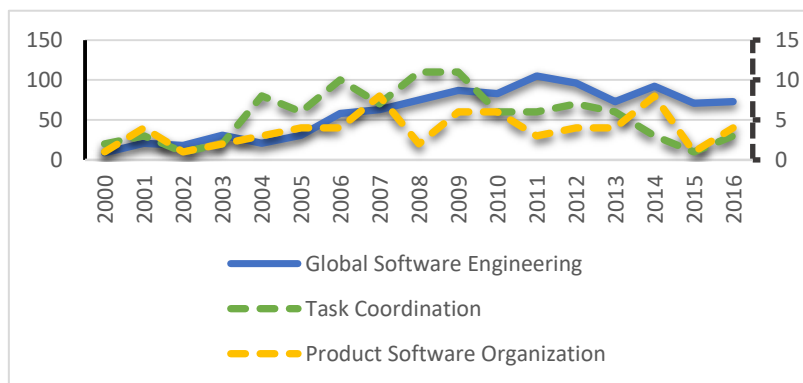


Figure 2-3 Number of articles found in DBLP

By using the search engines, we found a pile of articles which are related to coordination in software engineering, global software engineering, and product software organizations. Roughly, there are 73 articles found. The exact number of articles cannot be justified since there are some various results as well as some intersections among the search engines. We picked six articles as the starting point for the snowballing processes (Table 9-1). The remaining articles are still considered useful and were kept to be added to the references during the research progress.

The next step is searching related literature by iterating backward and forward snowballing to find more literature (Webster & Watson, 2002). Backward snowballing is performed by identifying interesting concepts and reviewing the reference lists from articles to find the meaningful discussion or other related concepts from the first step to finding prior articles. If needed, we go forward to find the more elaborate discussion by finding articles citing the articles recognized in the previous steps. This systematic search is used to ensure that we accumulatively complete census of relevant literature. These two approaches are repeated until there are no new concepts are found. After the snowball searching, we found that there are several publication sources which contribute more to our literature research (Table 2-1).

### Concept Matrix

We elicit the foundation of knowledge from systematic literature review where the authors are having different needs and background contexts. Hence, there can be widely varying jargons and terminologies for the same subject matters. The way to address this problem is by reducing and eliminating conceptual and terminological confusion and come to a shared understanding (Glaser, 1965).



Table 2-1. Literature Sources

Journals / Conference Proceedings / Books	Number of Results
ACM/IEEE International Conference on Global Software Engineering (ICGSE)	11
Information and Software Technology	7
Communications in Computer and Information Science	6
Lecture Notes in Computer Sciences	6
ACM/IEEE International Conference on Software Engineering (ICSE)	4
IEEE Software	3
Communications of the ACM	3
Journal of Software: Evolution and Process	3
Systems and Software Journal	3
Information Systems Management Journal	2
Management Information Systems Quarterly (MISQ)	2
ACM Evaluation and Assessment in Software Engineering (EASE)	2
ACM Empirical Software Engineering and Measurement (ESEM)	2
Ergonomics Journal	2
Software Quality Journal	2
Lecture Notes in Business Information Processing (LNBIP)	2
Collaborative Software Engineering	2
Management Science Journal	2
Advances in Intelligent Systems and Computing (AISC)	2
IEEE Requirements Engineering	2
Brazilian Computer Society	2
Information Systems Journal	2
Others	76
<b>Total</b>	<b>151</b>

From the selected articles, we use matrix analysis to structuring the review and summarizing the complex aspects from a higher perspective. A concept matrix can be characterized as a conceptual framework in a rectangular array of concepts. Researchers should construct the matrix based on their personal proficiency and originality to enable them to find the relationships between entries (Klopper, Lubbe, & Rugbeer, 2007). A guideline presented by Webster and Watson (2002) presents an author-to-concept-centric matrix as can be seen in Table 2-2 and Table 2-3.

Table 2-2 Concepts and authors mapping

Concept-centric	Author-centric
Concept C <sub>1</sub> ... [Author A <sub>2</sub> , ...]	Author A1 ... [Concept C <sub>2</sub> , C <sub>3</sub> , ... C <sub>n</sub> ]
Concept C <sub>2</sub> ... [Author A <sub>1</sub> , A <sub>2</sub> , ...]	Author A2 ... [Concept C <sub>1</sub> , C <sub>2</sub> ]

Table 2-3 is depicting the relationship between articles and concepts which are built by compiling the matrix after the literature is synthesized to identify the analysed concepts a proposed by the authors. The concept-centric and author-centric relationships can be presented as a matrix as can be seen in Table 2-2. All the concepts found during the literature review are inserted into this concept matrix.

Table 2-3 An illustration of a concepts matrix

Articles	Concepts				
	C1	C2	C3	...	Cn
A1		√	√		√
A2	√	√			
...					
Am			√	√	

**2.2.2 Semi-structured Interviews**

In addition to using literature review as a method for data collection, interviews are used to elicit information from the participant companies. An interview is a data gathering technique commonly used in qualitative research (Myers & Newman, 2007). There are various types of qualitative interviews which are grouped into three categories: structured interview, unstructured or semi-structured interview, and group interview. Structured interview uses a complete script that is prepared beforehand, and all questions are asked in the same order as in the pan. As well as a structured interview, the semi-structured interview also requires a list of questions before performing the interviews. The interviewer uses an incomplete script which is formulated as a general concern and interest from the interviewer to bring a room for improvisation during the interview (Myers & Newman, 2007; Runeson & Höst, 2009). Meanwhile, group interview is the interviewer where two or more people are interviewed at once by using a structured or unstructured list of questions.

In this research, we perform a semi-structured interview. A semi-structured interview is challenging because it requires openness, flexibility, and improvisation. Unlike a structured interview where there is no room for improvisation, a semi-structured interview does leave room for improvisation, which results in obtaining different results from each interviewee. Thus, the interviewer should manage the time by ensuring that there are no long pauses during the performance, but still able to cover all the questions that should be asked to the interviewees (Myers & Newman, 2007).

There are five companies which were participating in our research. One of the companies is not a product software company performing software engineering globally which helps us to contrast the findings. There are two respondents from each of the companies from various job positions who have experiences in global software engineering. The company’s names are changed with AlphaSoft, BetaSoft, GammaSoft, DeltaSoft, and ZetaSoft for the reason of confidentiality. The interviews were performed from December 2016 until February 2017.

The interviews attempted to capture several topics: company background, job roles and functions, partners or remote offices profiles, product profiles, company’s vision in GSE, challenges in performing GSE, approaches in GSE practices, and stakeholders involved in GSE projects. Each interview was performed between 45-60 minutes. The interview protocol is provided in Appendix A.

**2.2.3 Expert Opinion**

Design science allows many methods to validate design science’s artifacts, such as single-case mechanism experiments, technical action research, and expert opinion (Wieringa, 2014). Single-case mechanism experiment is a test where researchers apply stimuli to a model and explain the response regarding mechanism internal to the model. Technical action research (TAR) is

the use of an artifact prototype in a real-world problem to heal a client and to learn from this which is usually the last stages in scaling up technology from the laboratory to the real world. The difference between TAR and single-case mechanism experiment is the validation artifact in TAR is tested in a real situation with a client where the researcher also uses the artifact to help the client. The other validation method, expert opinion, is the simplest way to validate a research artifact. The proposed artifact is submitted to the experts who imagine how such an artifact will interact with problem contexts and then predict what effects that they think the artifact would have. Validation of the proposed method by expert opinion will work if the experts understand the artifacts which enables them to imagine problem contexts and predict the effects of the artifacts in the contexts. Unlike single-case mechanism experiment, TAR and expert opinion closely conform to conditions of practice because of the involvement of experts or clients from the real situations.

Since expert opinion is an effective way to validate new artifact designs, we decide to use this method to validate our proposed artifacts. We will present our artifacts to business practitioners in global software engineering from products software companies as our experts and ask them to give feedback to our framework and reference method. Thereafter, we improve our artifacts based on the feedback. The improvement of the framework will also lead to the enhancement of the reference method because the framework is utilized by the reference method. Critical feedback is useful than a positive one because it gives indications of improvement opportunities for the artifact. Negative feedback can indicate situations in real practices which are not thought of by the researcher. Therefore, expert opinion is useful to weed out bad design ideas early. The expert opinion interview protocol is provided in Appendix I.

#### **2.2.4 Situational Method Engineering**

For the analysis of the existing frameworks, techniques, and methods in global software engineering, we use the method engineering approach proposed by Brinkkemper (1996). Where software engineering pays attention to all aspects pertained to software production, method engineering focuses on the construction of method that fall into software engineering domain. Therefore, Brinkkemper (1996) defined method engineering as “the engineering discipline to design, construct and adapt methods, techniques, and tools”.

It is obvious that task coordination practices found in the literature and companies are diverse because of the complexity in which the approaches and tools are utilized as well as the situational factors that can influence a project (Kraut & Streeter, 1995; Li & Maedche, 2012; van de Weerd, Brinkkemper, & Versendaal, 2010). The variations are found because of the need to achieve better software engineering’ productivity and quality. For that reason, we need to construct a reference method that can be derived to adapt methods to the project situation at hand to support task coordination. The reference method is constructed from the fragments of the existing approaches. It can be a combination of methods with route maps or a high-level method scenario.

#### ***Metamodeling Techniques***

To support our method engineering approach, we use Process Delivery Diagram (PDD) as our meta-modeling technique. From the example of PDD in Figure 2-4 , PDD consists of two main parts: process view and deliverable view. The description of the PDD notations is available in Appendix H.

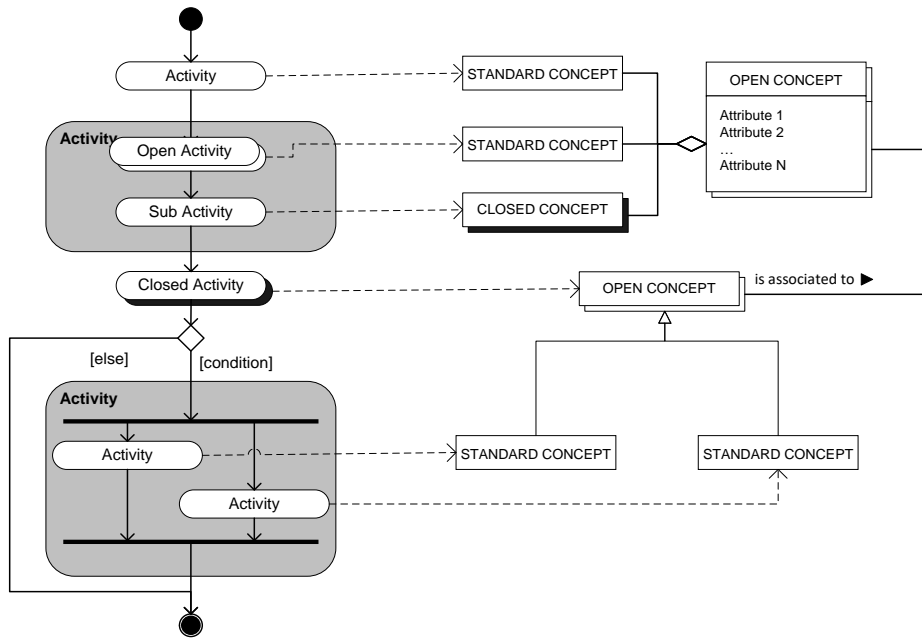


Figure 2-4 An example of a process delivery diagram

Process view adopts UML activity diagram to depict the activities and the transitions that show the control flow from activity to the next activities. Meanwhile, the deliverable view adopts the UML class diagram to depict the concepts which are involved, in, or created, or used by the activities or by other concepts (van de Weerd & Brinkkemper, 2009).

To build a task coordination reference method, we adopt the situational method engineering approach which is used by van de Weerd, Brinkkemper, Souer, et al., (2006) and method association approach by Luinenburg et al. (2008). The approach as can be seen in Figure 2-5 can be followed as below:

1. Perform preliminary study by conducting scientific literature review to identify established methods for this research.
2. Identify situational factors in established task coordination approaches
3. Identify activity groups from the preliminary study.
4. Choose a candidate method from established method.
5. Model method fragments of the chosen method.
6. Associate the method fragments to the activity groups.
7. Assemble situational task coordination method
8. Validate situational task coordination method

### 2.3 Plan Validity

As this research involves contemplating the developed method in an exploratory research, Yin (2013) suggests to take into account three types of validity: construct validity, external validity, and reliability. Internal validity is establishing a causal relationship which does not become the concern of our relationship.

1. Construct validity  
This validity test is establishing correct operational measures for the concepts being studied. We use multiple sources of data such as literature and interviews. We will also perform several follow-up interviews with our key respondents to validate our solution.

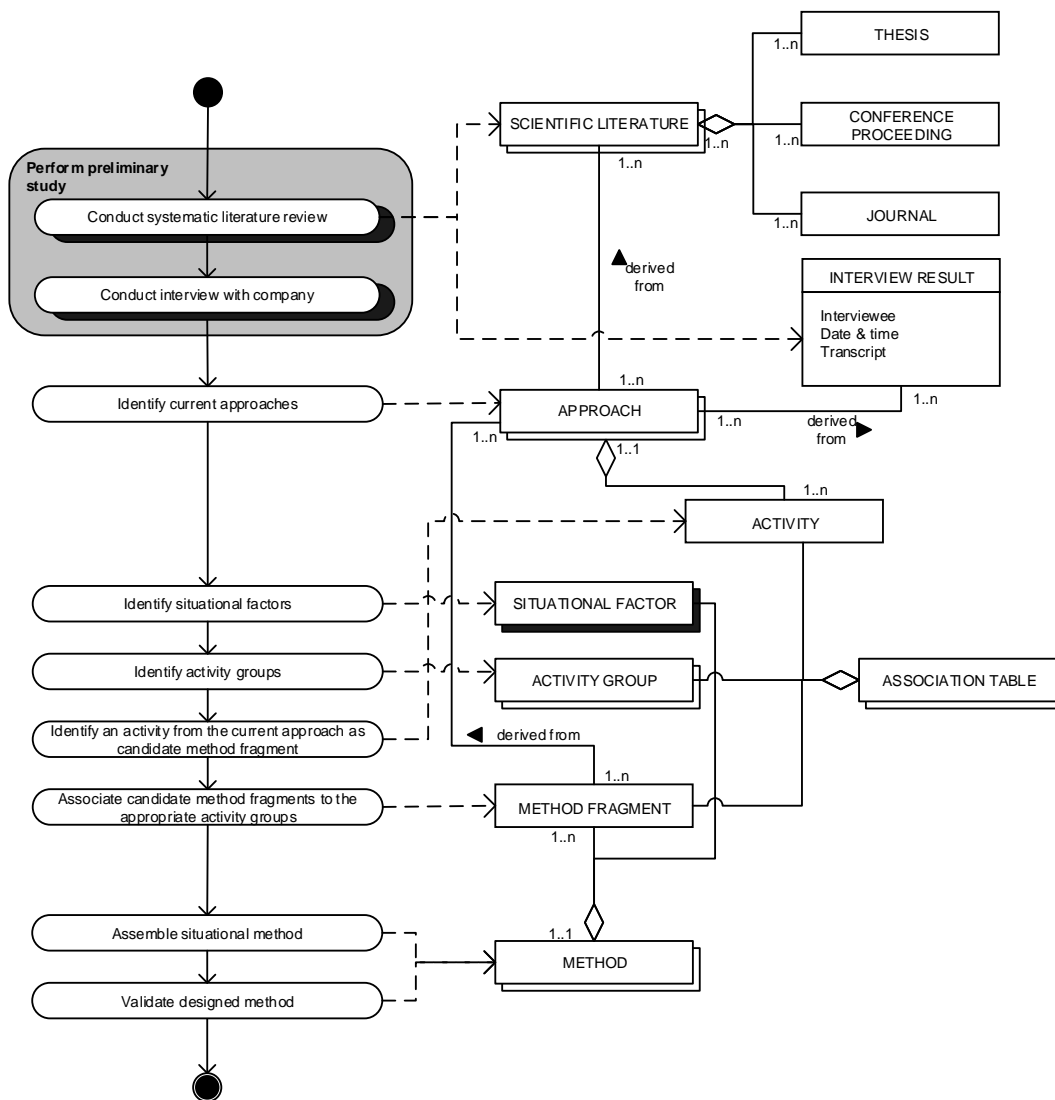


Figure 2-5 Method comparison approach in PDD

Both of these approaches are used to avoid subjectivity and bias of data (Yin, 2013, p.34).

2. External validity

The second test deals with the problem of knowing whether our findings can be generalized. We would not say that our finding and solution are applicable to our research contexts. Moreover, we argue that the external validity test is satisfied to a sufficient extent in this research by covering broader issues and through validating our model across different organizations (Lee, Baskerville, Lee, & Baskerville, 2017; Yin, 2013).

3. Reliability

The last test is to ensure that if later researchers followed the same procedures as described by an earlier researcher, they could replicate the same findings and conclusions. The common way of approaching the reliability issue, we develop and maintain our case study protocol to be followed and all data during this research (Yin, 2013, p.36).

## 2.4 Research Execution

This research is stipulated to be conducted within 8 (eight) months. To scale and manage the research, each task are grouped into the following main phases (Figure 2-6):

1. First Phase.  
During this phase, we planned the research management and built the understanding of the topic. This phase is ended up in a milestone where the first colloquium is presented.
2. Second Phase.  
This phase concerned on the research operationalization and finalization based on the planned schedule in First Phase. Two milestones of this phase are the 2<sup>nd</sup> colloquium presentation and the thesis defense. If possible, a scientific research report is produced to be submitted to a scientific conference in software engineering or IT project management domains.
3. Report Writing  
Since this research topic is selected, documentation process is performed throughout the planning and execution phases.

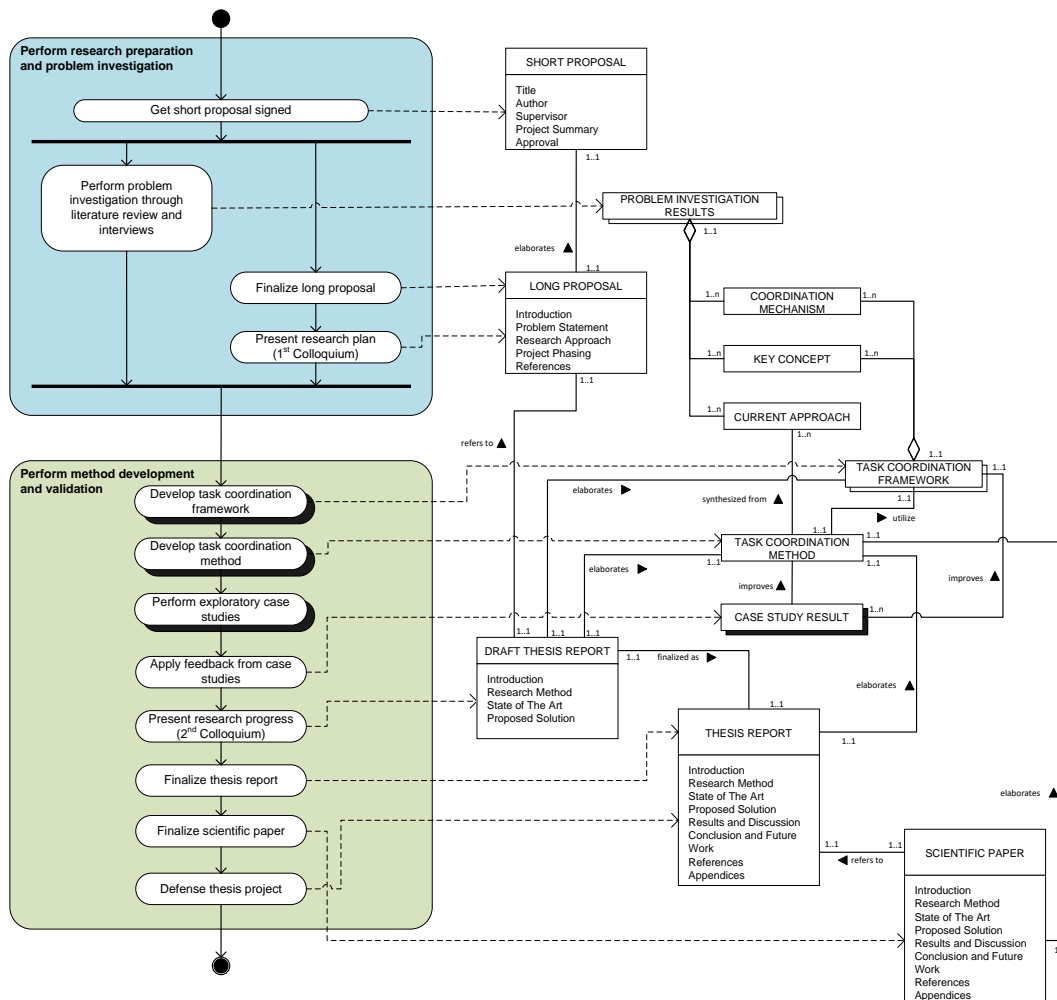


Figure 2-6 Project phasing in PDD

## **PART TWO      STATE OF THE ART**

**Chapter 3. Literature Study**

**Chapter 4. Coordination Practices at Product Software Companies: The Interviews**

**Chapter 5. Summary of State of the Art**





# Chapter 3 LITERATURE STUDY

Reviewing relevant studies which are previously researched is a starting point for a design science project. This chapter presents the results of problem investigation phase of our design science project which completed by performing a systematic literature review.

From the literature review phase, we identified several concepts related to task coordination in global software engineering projects at product software companies as can be seen in Figure 3-1 which are: Communication, control, knowledge, stakeholder, and tool.

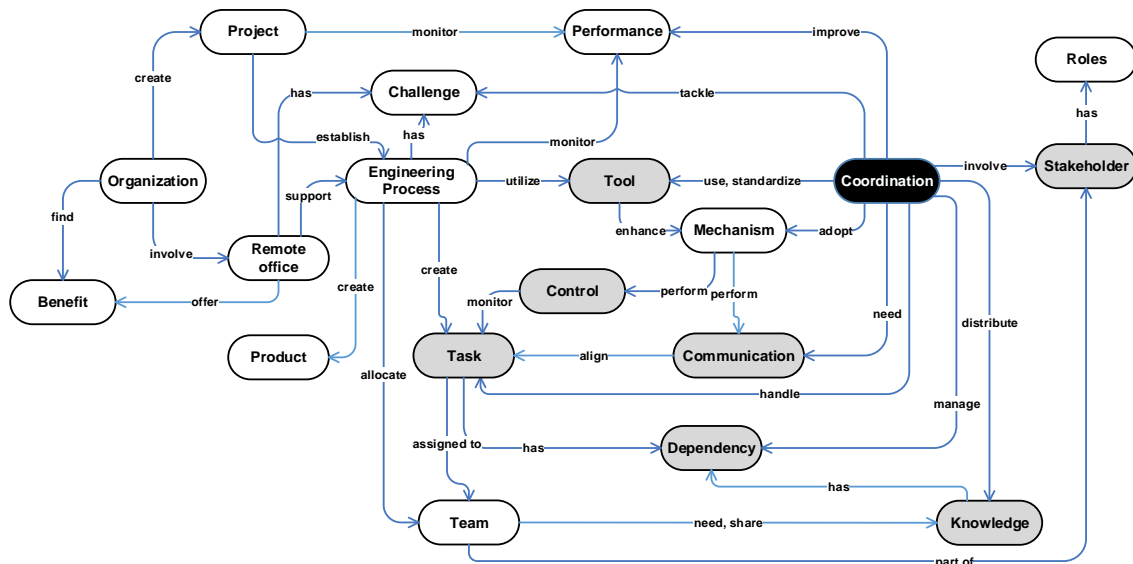


Figure 3-1 Concepts found during problem investigation

The literature that represents the evidence of the concepts can be seen in Table 9-2.

## 3.1 Global Software Engineering

As software engineering is defined as “an engineering discipline that is concerned with all aspects of software production” (Sommerville, 2010, p.7), it includes software project management and development tools, methods, and theories to support software production. To produce software, engineers should take into account practical cost, schedule, dependency issues, and the needs of software customers and producers. Software engineering uses software process as a systematic approach which leads to the production of a software product. Software process consists of four primary activities which are:

1. Software specification: customers and engineers define the scope of the software that will be developed and its features and limitations of its operations

2. Software development: the software is designed and developed
3. Software validation: the software is evaluated to ensure that it conforms to software requirements
4. Software evolution: the software is modified to adopt the changes required by customers or markets

Due to some reasons such as the increase in development costs and to get closer to market and customers, many companies practice global software development by carrying out parts of their engineering and development processes in various countries (Ågerfalk et al., 2008; Conchúir et al., 2009).

Niazi et al. (2016, p.1) defined global software development (GSD) as “the process whereby software is developed in different teams located in various parts of the globe”. Hence, global software engineering (GSE) can be discerned as an engineering discipline of software production where parts of the engineering process are dispersed in various locations. Ramasubbu et al. (2011) and Šmite (2007) propose numerous major distinguishing factors that describe dispersion characteristics in global software engineering as listed in Table 3-1.

*Table 3-1. Dispersion factors in Global Software Engineering*

<b>Factors</b>	<b>Description</b>	<b>Sources<sup>*)</sup></b>
Configurational	There is unevenness in distribution across sites. Multiple distributed member participation in a virtual team that develops software by joint effort is characterized by the number of collaborating partners.	R, S
Spatial / geographical distribution.	The geographical distance between the team members involved in the project.	R, S
Time-zone / temporal diversity	It is characterized by the level of working hours overlay, which most frequently differs from time zone differences.	R, S
Socio-cultural diversity	The level of social, ethnic, and cultural fit can differ even between the teams from one national location. Difference in mother tongue language that characterize the level of the common language skill (such as English) of the distributed team members also part of the socio-cultural diversity.	S
Knowledge Gap	The difference level of knowledge and expertise as well as the availability of the access to the required knowledge	K
Contextual diversity	The level of organizational fit or heterogeneity are characterized by diversity in experiences, process maturity, and inconsistency in work practices	R, S

<sup>\*)</sup>R : Ramasubbu et al. (2011); S : Šmite (2007); K: Kotlarsky, van Fenema, and Willcocks (2008)

### 3.2 Benefits and Risks of GSE

There are enormous potential benefits in distributing the engineering process globally. Cost savings is perceived as the most sought-after benefit of distributing software process across

countries (Ågerfalk et al., 2008). Companies share their development activities to leverage development costs from other countries such as India and China. Besides providing engineers with lower salaries, these countries also offer another benefit which is a larger developer pool with highly-skilled engineers (Conchúir et al., 2009). By acquiring subsidiaries or developing remote sites in other countries where the companies' clients are located, expanding markets and achieving customers closeness becomes possible (Herbsleb & Moitra, 2001; Jain & Suman, 2015). Performing software engineering globally also reduces time to market which is still a controversial benefit. The time zone differences are the degree to which companies can maximize productivity. Companies are managing resources in multiple time-zones by reducing the hand-over process to increase the number of hours in the 24-hour day during the development activities (Carmel, Espinosa, & Dubinsky, 2010; Herbsleb & Moitra, 2001).

In addition to the provided benefits, companies also face challenges because of the dispersed resources. The diversities as described in Table 3-1 implicitly imply that organizations that are performing GSE could face numerous problems in coordinating tasks among team members (Table 3-2).

*Table 3-2 Risks in global software engineering projects*

<b>Risks</b>	<b>Description</b>
Insufficient direct communication	Spatial distribution complicates team members' ability in having face-to-face communication with their colleagues when they need to discuss problems that eventually could extend the problem-solving time (Nguyen-Duc & Cruzes, 2013)
Process dependency problem	Ineffective handover when a team or individuals in should delay in performing their tasks because they use the same resources or need the result of tasks undertaken by the others can slow the project (Jain & Suman, 2015).
Inadequate collaboration	A lack of overlapping working hours limits coworkers in collaborative activities can cause the development process less efficient and delays in the project (Ågerfalk et al., 2008).
Distorted information	The development team at the remote office can obtain incomplete or distorted information about product requirements from product management team. The difference in knowledge also can cause information misinterpretation (Jain & Suman, 2015; Nguyen-Duc et al., 2012).
Traveling cost	To recover from insufficient face-to-face communication, maintain social contacts, and to build more trust, managers from host office need to do regular site visits which increase the travel budgets (Ågerfalk et al., 2008; Jain & Suman, 2015).
Lack of common understanding	The lack of knowledge sharing due to technological differences (such as different collaboration tool) can lead to an imbalance of common understanding among team members which leads to misunderstandings during discussions (Jain & Suman, 2015; Schneider, Torkar, & Gorschek, 2013)
Weak control in project management	Obviously, it is harder to manage interdependencies among tasks that are performed in different sites compared to collocated ones. Geographical differences complicates the managers' ability to monitor team members and task progress (Jain & Suman, 2015; Verner, Brereton, Kitchenham, Turner, & Niazi, 2014)

### 3.3 Task Coordination Approaches to Overcome GSE Challenges

Malone and Crowston (1994, p.90) define coordination as “managing dependencies among activities”. To face challenges and to achieve a desirable level of coordination effectiveness, many researchers have been formulated coordination strategies to coordinate tasks which can be grouped into two types of approaches: principles and framework. Principles is a basic idea or rule that explains or controls how something happens or works<sup>1</sup>. Meanwhile, the framework is a structure to make the conceptual distinction and organize idea by providing a network of concepts that together provide a comprehensive understanding of a system (George et al., 2011). The approaches identified during the systematic literature review are summarized in Table 3-3.

Table 3-3 Task coordination approaches

Researchers	Type of Artifacts	Proposed Approaches
(Olsson, Conchúir, et al., 2006)	Principles	Best practices such as buddy system, regular traveling, providing norms of messaging and optimizing asynchronous communication
Kircher, Jain, Levine, and Corsaro (2001); Li and Maedche (2012); Strode, Huff, Hope, and Link (2012)	Principles	Adopting agile practices (Scrum and XP process model) can optimize direct communication and build teams.
Kotlarsky, van Fenema, and Willcocks (2008)	Framework	Knowledge-based coordination mechanisms
Deshpande et al. (2011)	Principles	PMBOK® guidelines in GSE
Richardson, Casey, Burton, and McCaffery (2010); Richardson, Casey, McCaffery, Burton, and Beecham (2012)	Framework	Global Teaming provides two major key areas in starting and operating global software engineering.
Smirnova, Münch, and Stupperich (2014)	Framework	Global Canvas defines the roadmap of global collaboration projects
Wen (2016)	Principles	Providing a liaison officer or a broker to bridge the communication and knowledge transfer

#### 3.3.1 Overcoming Challenges Through Best Practices [L1]

Olsson et al. (2006) conducted an empirical investigation by performing interviews at three global software development companies. They classify issues related to work dispersion into three constraints: temporal distance, geographical distance, and socio-cultural distance. For each constraint, they distinguished several approaches performed by the companies in addressing those challenges from interviews and concluded recommendation actions.

**Temporal distance challenge.** Temporal distance is very close related to overlapping working hour management. When the time-zone becomes the biggest problem to organize the different teams in projects, it is necessary to consider moving remote teams to the

<sup>1</sup> Principles [Def. 1]. (n.d.). *Cambridge Dictionary*. Retrieved March 7, 2017, from <http://dictionary.cambridge.org/dictionary/english/principle>.

possible nearest area. Companies must avoid offshoring and choose nearshoring if they could not manage small overlapping or even no overlapping working hours. The temporal distance affects to daily communication within and between teams. Delay or responses is a frustrating situation for both sides especially when the coordination is related to time-critical tasks. This challenge impacts companies which have non-native English speakers at the remote office. Even though asynchronous tools are valuable to facilitate coordination, team members who are not native English speakers usually need more time to reflect before answering a question which then increases the time for the sender to receive a response.

Based on the best practices from other companies such as HP and Fidelity, 'follow-the-sun' approach is a solution for companies that support other teams at later time-zone. Another mechanism chosen by Intel entails considering the number of locations involved in the project. They decide to divide and distribute tasks only to two sites to make time-zone differences manageable. These mechanisms should be enabled by the use of technologies to support asynchronous communication and work collaboration (Sarker & Sahay, 2004). Companies also need to develop norms of messaging to avoid concurrent discussion of several topics where questions, responses, and comments can be directed and produced serially. Collaboration through technology still must be based on strong social relationships to help in tolerating the coordination complexity and increase the ability to handle multiple jumbled threads of conversations simultaneously.

**Geographical Distance.** A major challenge caused by physical distance is how to create a feeling of 'teamness' among distributed team members. The physical meeting is believed to establish a sense of trust and belonging. However, it is a common situation that cross-site relationships mostly exist at higher levels. Other research also found that employees at different sites sometimes do not feel like being part of the same team (Herbsleb & Mockus, 2003). To create a higher level of teamness, some managers consider having developers from remote locations meet each other to establish face-to-face contact.

To overcome the challenges caused by geographical distance, building team cohesion through periodical site-visit, co-located team building activities, and additional physical meeting especially during project definition session. Sharing team members' profiles through the online portal is also can be used that makes each of the team members can know each other and know to whom they should talk to for specific questions. Nearshoring also can be assumed as a better choice instead of offshoring to minimize communication issues associated with undertaking IT work at a distance. Also, a "bridgehead" or "liaison" officer can act as a mediator between sites to help in connecting the boundaries encountered in daily work.

**Socio-cultural Distance.** Socio-cultural difference is a complex dimension where language fluency, work ethic, process maturity, and culture at the level of national and organizational as well as political and legal aspects are involved (Šmite, 2007). This dimension is experienced by organizations which consist of heterogeneous team members from multiple countries. The team members who are not native English speakers found difficulties in understanding and interpreting requirements or assignments. Especially, conversations that focus on technical issues and involve rigor vocabulary are found to be hard to understand by all team members, which leads to misunderstandings.

In many cases, the language capability is not the biggest problem. Since asynchronous communication does not deliver the emotion and expression, team members at different countries have different assumptions regarding what to say, how to say it and when to say it. Team members from different countries also have different cultures in giving responses. Japanese developers need more time to provide the responses because they want to give complete information. On the contrary, Indian developers usually reply immediately with less information because their colleagues at the head office prefer some acknowledgment of their questions.

Some companies use asynchronous communication to overcome the language problem to adhere socio-cultural challenges. This solution lets team members take the time to rethink and evaluate the assignments or questions. However, they also must consider providing immediate acknowledgment as soon as possible when they need time to accomplish the task or to provide the complete answers.

Companies also can use 'buddy system' where team members at remote sites are buddied up with team members from the head office as their mentors. Occasional traveling and face-to-face meeting to the remote office are necessary to share information, build trust, and influence team members at the remote office with people at the host office's way of working.

### **3.3.2 Adopting Agile in GSE [L2]**

Agile software engineering (Agile) is not a methodology. Agile combines a philosophy and a set of development guidelines to encourage customer satisfaction by providing early and incremental delivery of software. In Agile, software stakeholders work together as a self-organized team and control the project by themselves, so they have the same view. An agile team is characterized by its intensive communication and collaboration among all team members to provide an operational and incremental software on the appropriate commitment date (Pressman, 2010).

Agile philosophy is perceived as a revolutionary change to overcome the limitations of plan-driven and traditional heavyweight approaches, such as difficulties to predict requirements in advance, the needs of providing proven design before the construction phase, and challenges in predicting the ideal engineering processes from a planning point of view. Engineering processes are designed incrementally to adapt changes and uncertainties. Hence, Agile development brings the following values (Pressman, 2010) :

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

These items are elaborated by the following 12 principles (Pressman, 2010):

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.

5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity, the art of maximizing the amount of work not done—is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Many Agile process models are developed, such as Scrum, Extreme Programming (XP), Adaptive Software Development, Dynamic System Development Method (DSDM), Crystal Programming, Feature Drive Development (FDD), Lean Software Development, Agile Modelling, and Agile Unified Process. In a global software engineering context, XP and Scrum are the two widely process models practiced by companies (Strode et al., 2012). As discussed in a study performed by Paasivaara and Lassenius (2006), Agile methods and GSE could seem incompatible because many studies report communication as the biggest problem of distributed software engineering. However, by extending the process models, such as distributed XP and distributed Scrum, Agile might help to resolve this issue by suggesting communication practices that could be used to satisfy the communication needs of distributed engineering situation.

### ***Scrum in Global Software Engineering***

Scrum focuses on managing iterative and incremental development approach that moves project control from a central scheduling to dispatching authority and responsibility to the team members working on the tasks (Schwaber, 2004). Scrum provides the management framework for software engineering projects that consists of three main stages (Schwaber, 1997, 2004; Sommerville, 2010):

1. Outline planning. Product Owner leads a team of customers or business users from various disciplines such as marketing and product management to initiate a list of features. Then, the Product Owner should prioritize features list of the product and document the results into the product backlog.
2. Sprint cycles. These cycles are the innovative part of Scrum that consists of several activities, which are:
  - a. Planning Meeting
  - b. Daily Stand-up
  - c. Review meeting
  - d. Retrospective meeting

At the end of the iteration, the deliverables can be released as a new incremental release.

Scrum process model in step by step activities as depicted by using Process Delivery Diagram (PDD) in Figure 3-2 consists of two main phases: Developing product backlog phase and Scrum iteration phase. The developing product backlog phase is used to manage the requirements and to design product features. Meanwhile, the scrum iteration phase is the product realization process through iterative sprint activities (Schwaber, 1997, 2004; Sommerville, 2010).

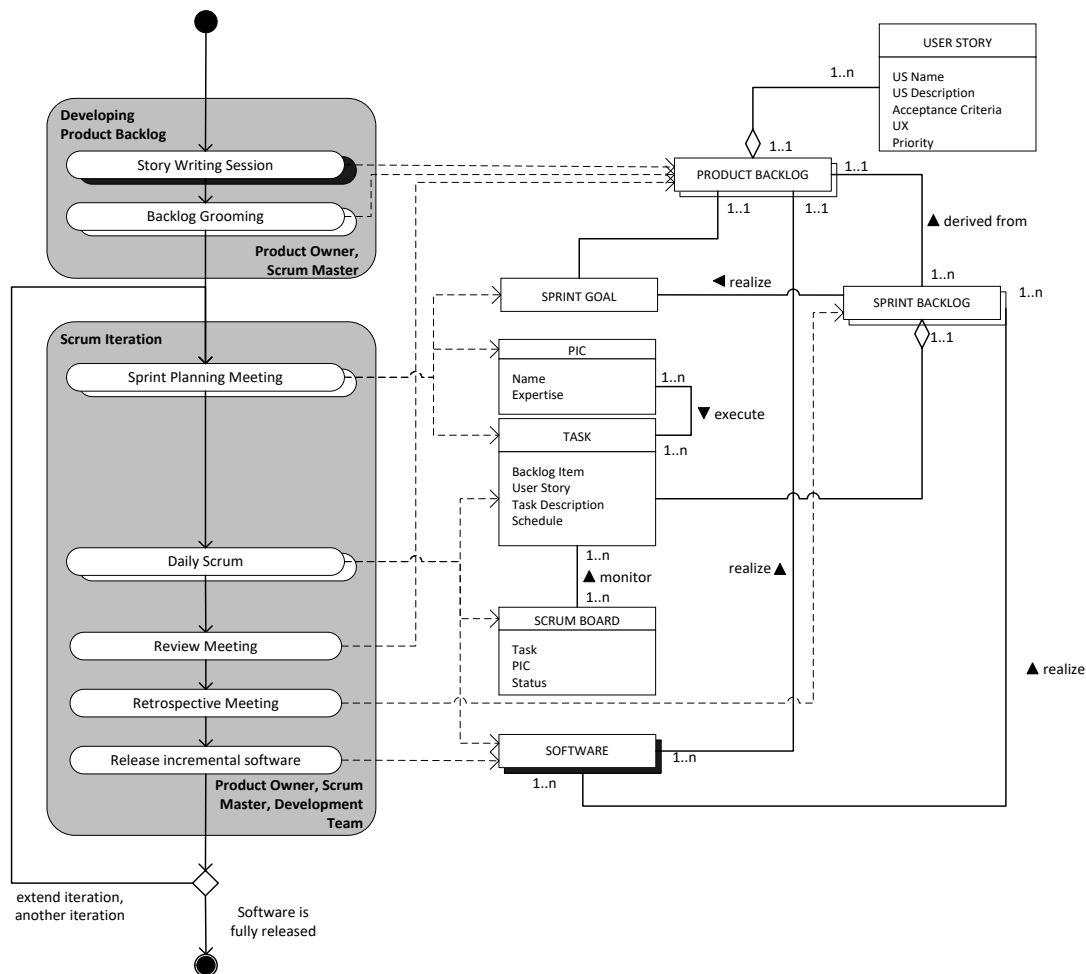


Figure 3-2 PDD of Scrum process model

Based on the Agile principles and the process depicted in Figure 3-2, it is obvious that Scrum is well suited when stakeholders are collocated, and there is an intensive and frequent interaction among them. In the real situation, many projects require more effort and involve multiple scrum teams which are possibly located in distributed locations. The teams work in parallel through a variety of coordination mechanisms. An appropriate infrastructure such as high-bandwidth technology for source code sharing and synchronized builds, and alternative communications such as instant messaging should be put in place to implement frequent work synchronization and coordination among distributed scrum teams (Paasivaara & Lassenius, 2006; Schwaber, 2004). Therefore, Schwaber (2004) suggested adding a staging phase, where non-functional requirements are defined and prioritized to Product Backlog to support the collaboration, which are:

1. Decompose business architecture to support clean-interface multi-team development.
2. Decompose system architecture to support clean-interface multi-team development.
3. If necessary, define and implement a development environment to help multi-team collocated or distributed environments.

To facilitate the staging step mentioned by Schwaber (2004) are added as non-functional requirements in the product backlog grooming session as depicted in Figure 3-3.



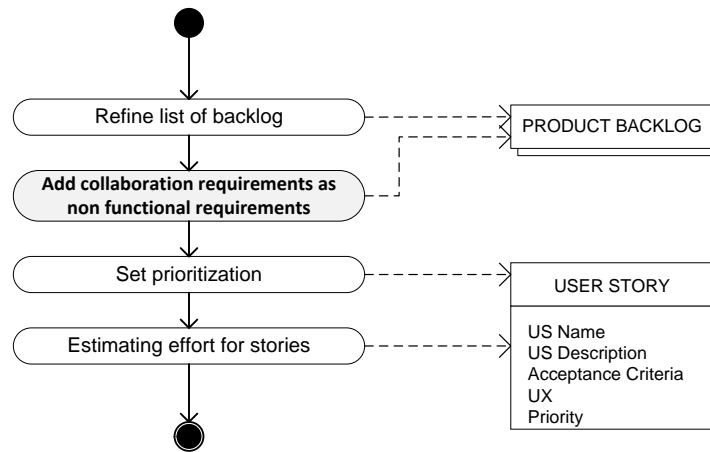


Figure 3-3 PDD of Scrum product backlog grooming session

Sprint Planning Meeting (Figure 3-4) is a meeting that initiates an iteration in Scrum where Product Owner and team members get together to collaborate about what will be done for the next Sprint iteration (Schwaber, 2004). In this meeting, the knowledge of how to do the tasks should be already explicit, or at least all team members know where the expertise is located and know where the expertise is needed. It becomes necessary for an effective coordination during sprint execution where entire team members have a comprehensive understanding of the goal as well as the tasks priorities, and how each team members work fits in with other team members' work (Strode, Hope, Huff, & Link, 2011). Coordination effectiveness will be achieved when the entire agile software development team has a comprehensive understanding of, project goal, project priorities, what is going on and when, what they as individuals need to do and when, who is doing what, and how each team member work fits in with other team members work (Strode et al., 2011, p.15).

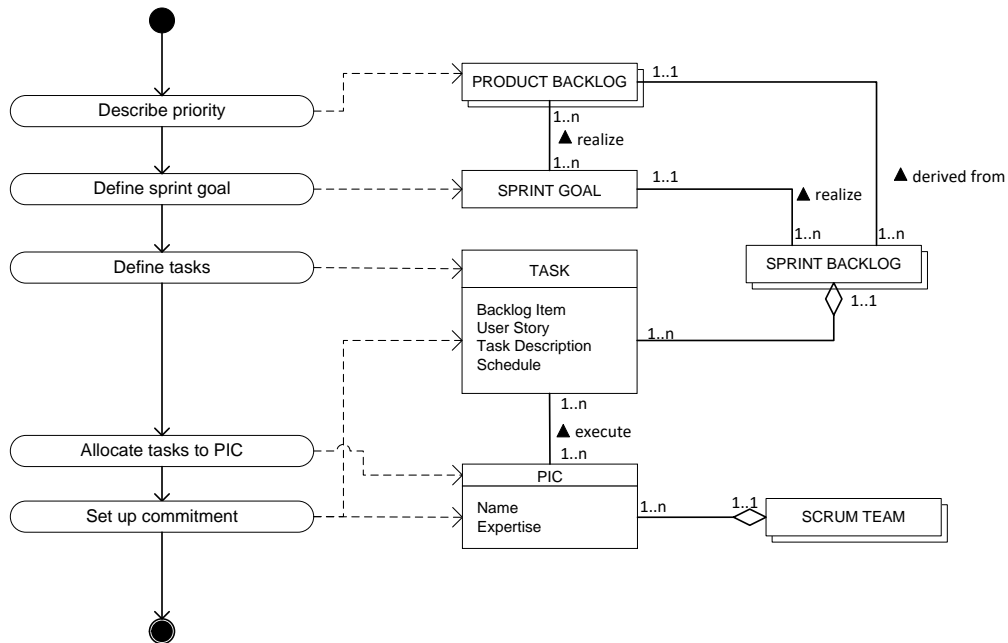


Figure 3-4 PDD of Sprint planning meeting

During the iteration, all team members get into a 15 minutes' stand-up meeting namely Daily Scrum Meeting which is led by a scrum master (Figure 3-5). The stand-up meeting can

be perceived as a synchronization activity where each team member being better informed about who is performing what task on the project on that day, this contributes to the implicit component of coordination effectiveness (Strode et al., 2012). The use of sprint backlog and scrum board as project monitoring tools in daily meetings helps team members to express and visualize what value has been delivered and where all attendees can quickly see whether the project is on track. The daily stand-up meeting also can be considered as a control mechanism that serves common milestones to team members, provides quick feedback, and updates progress reports (Pries-Heje & Pries-Heje, 2011).

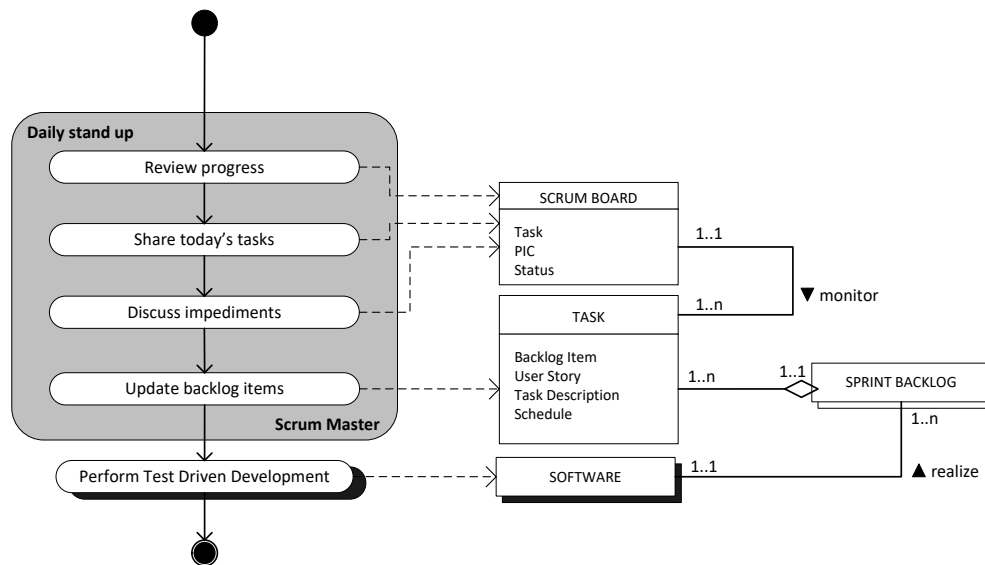


Figure 3-5 PDD of daily stand-up meeting

### *XP in Global Software Engineering*

Similar with Scrum, in Extreme Programming (XP), requirements are developed incrementally by breaking down the requirements (user stories) into several tasks. Two main differences between Scrum and XP are: XP has shorter iteration (XP has 1-2 weeks, Scrum has 2 or more weeks), and task prioritization in XP is done by the Product Owner (customer) where the teams are required to work on the tasks in that prioritization order. In the meantime in Scrum, the tasks prioritization is done by the Scrum team (Cohn, 2007).

There are important things which become the characteristics of XP: Planning game, pair programming, collective ownership, and continuous integration. Pair programming means developers work in pairs and check each other's work. Collective ownership allows the pairs of developers work on all areas of the systems and all the developers take responsibility for all the code. Meanwhile, continuous integration means all the new finished deliverables should be integrated as soon as possible to the whole system and continued by testing the system (Sommerville, 2010, p.66). Those practices need an intensive communication and expertise to coordinate the tasks among the engineers. Since the developers take the same responsibility for the code, they must have an equal level of knowledge about the code. However, in a circumstance where projects with teams residing in other locations, the projects cannot be done as in a collocated location. Thus, XP should be improved to address the challenges in coordination caused by the team distribution such as communication and availability. For that reason, Kircher, Jain, Levine, and Corsaro (2001) suggested Distributed eXtreme Programming (DXP) to address this problem (Table 3-4).

Table 3-4 Practices in Distributed XP

Processes	Practices
Planning game	For release planning and iteration planning with customers being remote, video sharing with application sharing support can be utilized.
Pair programming	Developers can use IDE that supports remote pair programming.
Continuous integration	There should be at least a team or individual who becomes central role. A central role can invite other team members to do common integration on the development machine.
On-site customers	Again, with the use of video conference system, remote customers can be treated as “virtual on-site customers”

### 3.3.3 Adopting PMBOK® in GSE [L3]

Project Management Body of Knowledge (PMBOK®) is a project management standard from a managerial perspective (PMI, 2000). Project management is defined as the application of knowledge, skills, tools, and techniques to projects activities to meet project requirement (PMI, 2000, p.6). PMBOK® can be used in many types of projects including software engineering projects. As a body of knowledge, PMBOK® covers several areas of processes, such as communication management, integration management, and human resource management.

Deshpande et al. (2011) identified that GSD practices in coordinating tasks from literature and performed an empirical research study to investigate GSD practices with vendor companies in India. They compared the results with PMBOK® Guide processes as the basis and established both common and the unique processes to both GSD and PMBOK® Guide. The result is a set of GSD coordination processes which support project managers in overcoming GSD coordination challenges and issues.

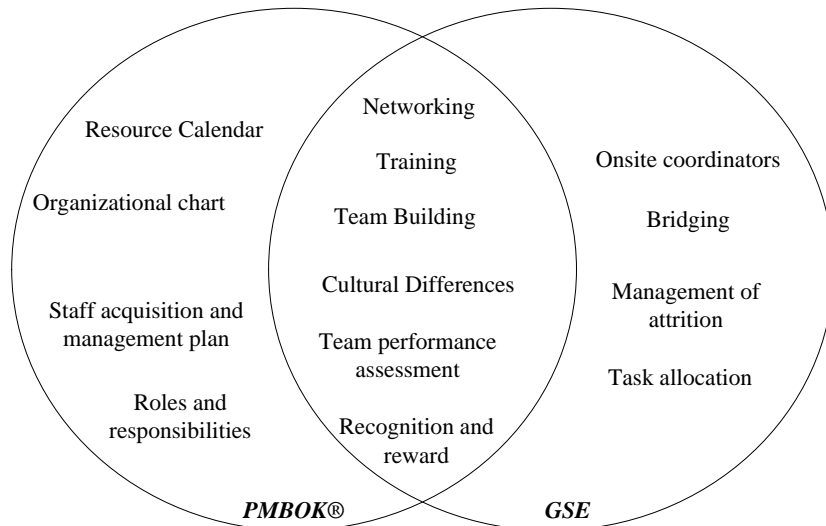


Figure 3-6 Process mapping of PMBOK® Guide and GSD practices

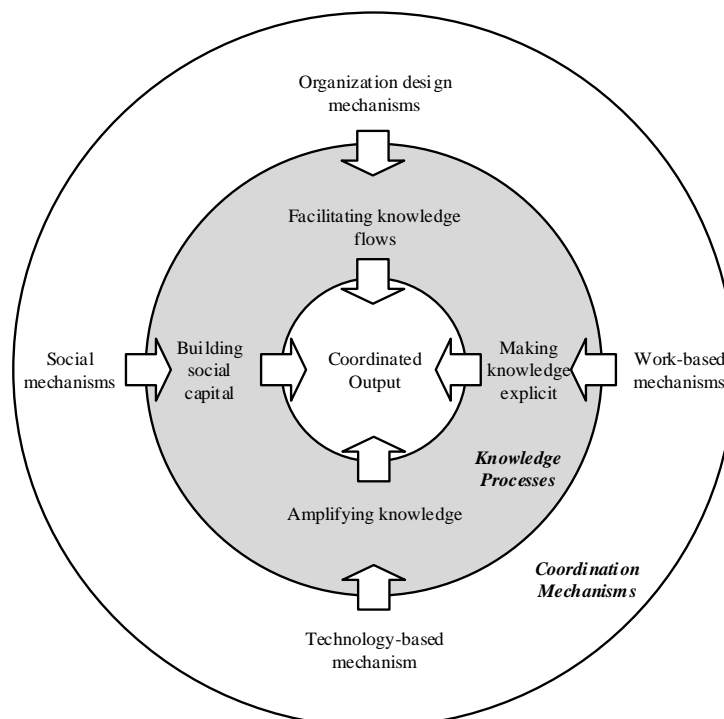
They identified 14 processes in GSE practices from the literature and interviews and Project Human Resources Management chapter in PMBOK® Guide by performing a comparative analysis of coordination processes. From this map, PMBOK® Guide can be used as a starting point for companies to identify and prepare the processes that should be performed on global

software engineering projects. Nonetheless, there are additional processes which are not covered by PMBOK® Guide. On-site coordinators, bridging, management of attrition and task allocation are essentials since they are found from the previous studies and experienced by the interviewees. On-site coordinators are the ones that can facilitate communication better between sites by bridging the sites to manage cultural, linguistic and knowledge differences.

Companies and they remote sites or global partners should understand the triggers that can cause attrition so that they can take necessary steps to overcome it. Companies also should consider the strengths, weaknesses, and interests of their resources or global partners to be able to allocate tasks to the best resources. Thus, these processes should be acknowledged by companies and included to be the part of common practices for a successful global software engineering projects. Unfortunately, no stepwise guidelines provided on how to apply the process mapping in daily practices.

### 3.3.4 Managing Knowledge Processes in GSE [L4]

Kotlarsky et al. (2008) see that coordination as the achievement of concerted actions. Organizations should arrange activities across dispersed units to facilitate knowledge flows by providing a structure through which expertise and information can be interchanged. The goal of coordination mechanisms is to build the coherence of knowledge processes in achieving a coordinated outcome. Therefore, they developed a knowledge-based perspective on coordination and demonstrated its applicability in the context of globally distributed software projects. They suggest that categories of coordination mechanisms should facilitate knowledge processes. For example, work-based mechanisms make knowledge explicit and accessible, while social mechanisms are needed to build social capital and to exchange knowledge and ideas.



*Figure 3-7 Knowledge process model to support task coordination*

They performed interviews with successful and unsuccessful global software engineering projects from two companies. By using the framework, they identified some best practices from

global software organizations as can be seen in Table 3-5. By referring to the knowledge process model as depicted in Figure 3-7, there are several steps that should be considered by organizations to achieve a coordinated knowledge:

1. Identify situational factors from the organization artifacts such as processes, structure, technology, and social activities. Organizations could recognize one or more factors that usually occur in their information processing properties, by:
  - a. Probing on how organization design defines the roles and the cooperation practice that constitute learning and value creating processes.
  - b. Observing on how tasks are structured that encourage individuals coordinate activities
  - c. Finding tools or platforms that support individuals in managing resources and interacting with their environment synchronously or asynchronously.
  - d. Investigating working relationships and social cognition among individuals in the organization where they try to build a shared understanding of new circumstances or to adapt one to another.
2. Define knowledge processes based on the situational factors found in the organization. The chosen mechanisms should be aimed to encourage organizations in coordinating knowledge and optimizing sharing to manage dependencies and produce effective team operations (Table 3-5).

### **3.3.5 Managing Virtual Teaming [L5]**

Another research by Richardson et al. (2010) found that many companies are struggling with the successful implementation of GSE because of temporal, cultural, and geographical distance. They proposed Global Teaming (GT), a software process model which includes specific practices to ensure that requirements for successful GSE are stipulated.

Global Teams have the same goals and objectives with traditional teams. Traditional team can be described as a social group of individuals who are collocated and interdependent in their tasks. The main difference point is that Global Teams operate across time, geographical locations and organizational boundaries. The main objective of Global Teams is to function as a single team with the same goals as if they are localized in one place. Global Teaming focuses on two goals:

- defining a well-managed global project management (more related to the project management perspective), and
- defining management between locations (more specific in communication and collaboration strategies).

Global Teams Process Area Framework is built by reflecting CMMI® structure and identifying explicit and implicit GSE factors in CMMI® (Figure 3-8).

Table 3-5 Best practices

Coordination Mechanisms	Approaches or Best Practices
<p><i>Organizational design mechanisms:</i> facilitate knowledge flows to reduce existing gaps and prevent knowledge and information gaps in the future.</p>	<ul style="list-style-type: none"> <li>• Creating cross-continental mini-teams was helpful in shaping communication patterns, providing clarity and thus facilitating knowledge-sharing processes between organizations and their remote sites.</li> <li>• A clear division between technical and social supervision (management of local teams) in which local development manager is responsible for ensuring the quality of the product and effective team operations.</li> <li>• Direct communications were encouraged in the knowledge collaboration group to facilitate knowledge sharing.</li> </ul>
<p><i>Work-based mechanisms:</i> capture knowledge and make it explicit and accessible to all team members despite their geographical location.</p>	<ul style="list-style-type: none"> <li>• Dividing works by feature provides dispersed teams with full ownership of and responsibility for the entire block of functionalities to reduce knowledge dependencies which eventually reduce misunderstandings and conflicts</li> <li>• Standardize tools and methods used by dispersed teams will ensure consistency and facilitate a common understanding of the products. A sharing of knowledge embedded in the standards aided coordination across the locations, as people performed interrelated tasks coherently.</li> </ul>
<p><i>Technology-based mechanism:</i> amplifying knowledge sharing of the team by using technologies to communicate, coordinate and share knowledge will allow remote team members to share explicit knowledge resources and increase the speed and flexibility of knowledge sharing independent of place and time.</p>	<ul style="list-style-type: none"> <li>• Facilitating the reuse of knowledge and software components across locations will reduce time-to-market of new product version.</li> <li>• Centralizing technologies by utilizing Internet and web technology under a single environment accessible from all remote locations is important to ensure everybody works on most up-to-date versions and at the same time and allow remote counterparts to update his or her knowledge about on the situation, including plans and the progress.</li> <li>• The use of application sharing and video conference tools can help counterparts from dispersed locations learn to know the composition of a remote team and knew whom to contact. These tools also can be prioritized to be used in high priority and urgent situations. Meanwhile, email can be used for low priority tasks and issues, and tasks that could not be completed in real-time because of time-zone differences.</li> </ul>
<p><i>Social mechanism:</i> create social capital for the global team by building up shared experiences, team building, and creating memory transactions among team members to reduce knowledge gaps, build relationships and maintain team atmosphere are considered important to ensure effective coordination over distance.</p>	<ul style="list-style-type: none"> <li>• The transactive memory in the group started with the project initiation can influence the information that had to be shared will bring an impact on the efficiency of communication.</li> <li>• Organizing a team-building exercise bridges the knowledge gap and facilitate knowledge sharing between the teams in the early stages of the project and gives an opportunity for major members to meet, learn about areas of expertise and cultural differences of remote counterparts, and create space for social interaction.</li> <li>• Mutual adjustment included setting up rules of communications helps people adjust to communication styles and reduces the misunderstandings and confusions that typically happened as a result of different cultural backgrounds.</li> <li>• Organizing frequent distant interactions through regular teleconferences and face-to-face interactions facilitates interactions between remote counterparts which eventually helps to keep the knowledge of all parties up to date.</li> </ul>

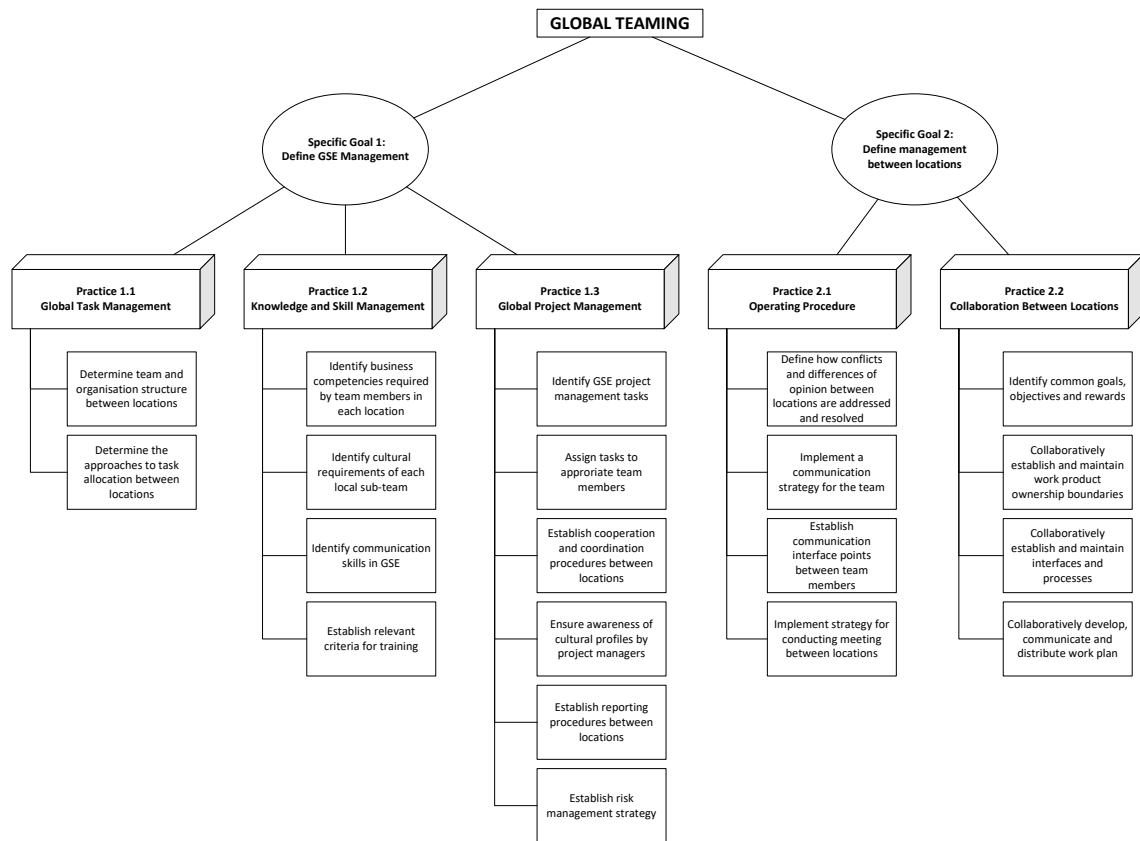


Figure 3-8 Global Teaming process area (Richardson et al., 2012, p.1184)

### 3.3.6 GSE Canvas Model [L6]

A canvas model usually is used to describe process chain, the interaction between functions and information transferred in a layered design. Smirnova et al. (2014) utilize canvas model to provide a holistic approach that synthesizes knowledge and guide companies to set up global collaborations for software-based products and services in a systematic way. They investigated important aspects and practices which are needed when starting global collaborations in software development from literature study and advice taken from industrial partners. Then, they prioritized and aggregated the aspects and practices to provide credible and helpful activity roadmap for practitioners. They propose Global Canvas as a model that visualize the structure of activity roadmaps for organizations intending to establish global development collaborations. The activity roadmap incorporates nine elements (Table 3-6). Because the canvas aggregates all the main necessary aspects and presents the activities as feasible roadmaps, it also can be used as an assessment scheme .

The proposed activity roadmaps for organizations who want to establish global software development collaborations are described as in Figure 3-9 that shows that there are five stages that must be passed so that organizations can embrace the ninth element. Organizations can follow four main phases as shown in the diagram in Figure 3-10.

Table 3-6 Global Canvas Elements

<b>Element</b>	<b>Description</b>
Strategy	Organizations should ask themselves the reasons behind on why they collaborate globally to identify their current situations, what is the expected situation, and how are they going to do to achieve their goals.
Collaboration Structure	Collaboration Structure is aimed at determining the approach of development task allocation between locations based on collaboration goals, creating roles and responsibilities along with the way of distributing them, and defining an organizational structure and peer-to-peer connections between sites.
Product Structure	Product structure addresses how product architecture could be adapted for global software engineering compared to centralized development where the Product ownership boundaries between locations, and how modifications to the product part at one location can affect work at other locations.
Coordination	Coordination holds an important role where the resources availability and capabilities need to be effectively managed for the collaboration goals.
Development Process	Development process aims at defining the model for software development activities between the collaboration sites by defining the processes at the interfaces between the collaborating sites without aiming at the unification of all processes at all sites, especially when the sites belong to different organizations.
Communication	Communication addresses all kinds of communication activities between the different development sites. It becomes crucial and needs to be considered early on because global collaboration is a large degree human-based interactions.
Social Aspects	Social aspects refer to the process through which team members gain the knowledge on behavioral and communication norms, attitudes, cultural and social patterns of each other to work together in cooperation.
Infrastructure	Infrastructure refers here to all tools, platforms, and other technical means that support technical, organizational, and managerial activities in the context of distributed software development, maintenance, and operation.
Organizational Change Process	There is typically a period when team members learn to know each other and better understand the ways of working together at the first stages of a global collaboration. In this phase, the software development efficiency is usually recovering gradually. By a change management process, global collaborations will perform scaling effects gradually on efficiency that goes beyond the efficiency of centralized development.



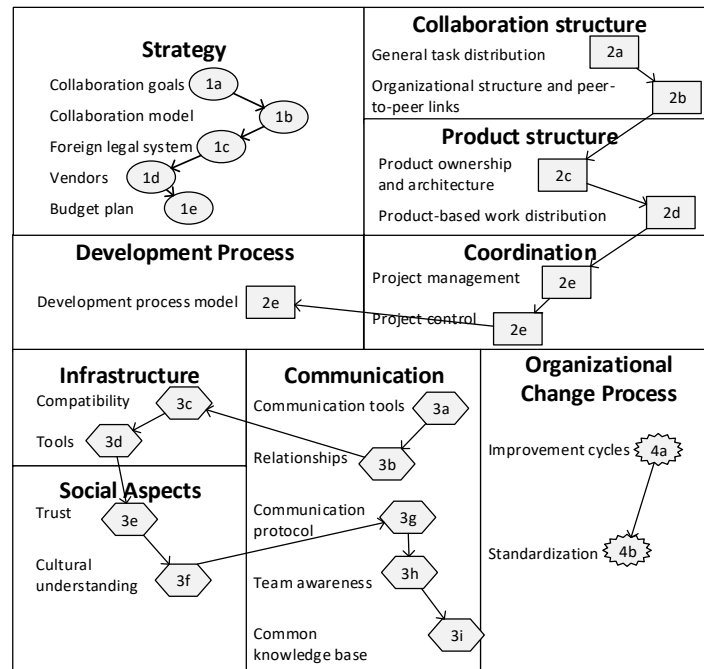


Figure 3-9 Global Canvas (Smirnova et al., 2014, p.88)

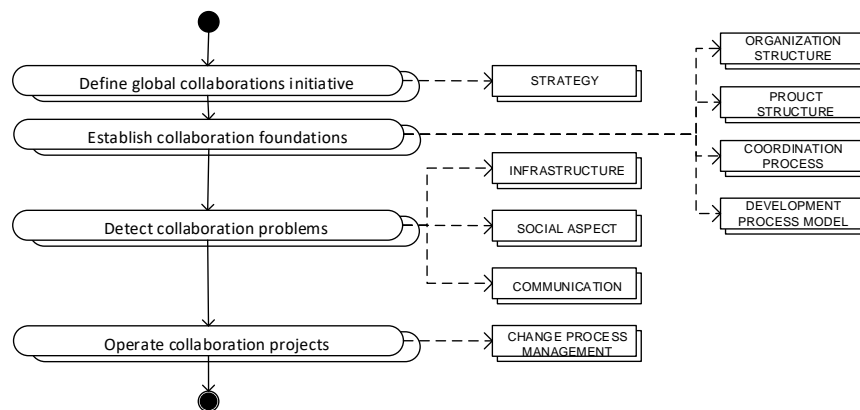


Figure 3-10 PDD of Global Canvas processes

### 3.3.7 Assigning Liaison Officer as Broker [L7]

Wen (2016) identified that many GSD projects suffer from a communication barrier which is caused by language, cultural, and time-zone differences between the stakeholders. To deal with the barrier, Wen proposed a new role called Global Software Development Broker (GSDB). A GSDB should possess several requirements such as having excellent communications skills and good knowledge of commercial law systems for multiple served countries. The requirements are needed to help GSDB to perform several functions, which are:

1. Identify suitable partners, either a host company to global partners and global vendors to find a project from prospective host companies.
2. Provide legal service by helping companies negotiating and developing a mutually beneficial contract regarding the different commercial laws at different countries.

3. Communication service to reduce communication cost, to improve communication quality between different stakeholders, and to help communication flow among team members in global projects.

Regarding task coordination, a GSDB should be able to help the company to simplify the communication network structure. Assigning or recruiting someone as GSDB may increase the total communication amount. However, if the GSDB can streamline the communication structure and overall communication quality, it will reduce the complexity and manage the communication traffic which at the end will reduce the communication cost. That is why, a GSDB should be able to profound the communication skills with the local cultures of both locations to facilitate information exchange, identify expertise, and mediate cultural differences (Verner et al., 2014).

In coordinating knowledge, it is also important to assign communication brokers depends on the needs of knowledge. Kristjánsson, Helms, and Brinkkemper (2014) distinguished two main types of knowledge: Functional and technical. Functional knowledge is associated with the desired or implemented functionalities of the product. Meanwhile, technical knowledge is associated with the implementation of the functionalities. They found that a boundary-spanning knowledge broker who is the expert from the customer side can supports knowledge development at the vendor side and transfers new knowledge back to the organization. This role is responsible for the technical liaison and cultural liaison to reduce the knowledge imbalance and cultural difference. The liaison officer can also be an engineer from the remote office or vendor side who is temporarily assigned to the host office to facilitate communication between sites. He/she is also assigned in gathering knowledge from head office, responsible for transferring and creating knowledge shared space at his/her organization through internalization or socialization (Espinosa, Slaughter, et al., 2007).

### **3.4 The Literature Study's Summary**

From the literature, we summarize the dimensions that describe how the teams are distributed in product software companies into five basic measurements: spatial, temporal, knowledge, socio-cultural and contextual. We found several best practices for task coordination adapted from GSE guidelines such as the Global Teaming, the GSE Canvas Model, and Kotlarsky's Knowledge Process Model that report coordination approaches as communication, project management, and knowledge sharing practices.

In the next chapter, we try to dig deeper into the findings we get in this chapter more in depth on the product software companies that participated in the research. Through interviews, we will confirm the latest issues and best practices of these companies as updates and complement of our literature study results..

# Chapter 4      COORDINATION

## PRACTICES AT PRODUCT

### SOFTWARE COMPANIES

In this chapter, we present the results of our preliminary case studies in GSE projects at product software companies. The goal of this study is two-fold: (1) to capture the state of the art of the challenges and companies' best practices in performing GSE, and (2) to clarify the of the theories with the current practices. There are several primary concepts we identified in the literature study as presented in Table 9-2: control, communication, stakeholder, dependency, knowledge, tool, and project performance. Due to this exploratory nature, we held semi-structured interviews to allow participants to discuss the key topics freely to clarify these concepts such as "*How do you manage communication among distributed teams*" and "*What are the problems in controlling dependencies?*". The five participating companies are headquartered in the Netherlands, but their experience in performing GSE, the size of distributed teams, and the way of the participating units are distributed differentiate to ensure some degree of heterogeneity in the results.

#### 4.1 Product Software Company

Further to our introduction section, the software market is shifting from tailor-made software into product software. Xu and Brinkkemper (2007) classified software into four categories (Figure 4-1). They defined product software as "a packaged configuration of software components or a software-based service, with auxiliary materials, which is released for and traded in a specific market." Hence, a product software organization can be described as a company that develops and sells mainly software as its products for a target market without customer-specific modifications (Vähäniitty, 2006).

Bekkers, van de Weerd, Spruit, and Brinkkemper (2010) present a software product management (SPM) competence model as a comprehensive overview of all important areas of software product management (Figure 4-2). The model was developed from an SPM reference framework by van de Weerd, Brinkkemper, Nieuwenhuis, Versendaal, and Bijlsma, 2006. The competence model depicts the interaction and information flows among stakeholders through four business functions: requirements management, release planning, product road mapping, and portfolio management. Each business function is elaborated with several focus areas, which.

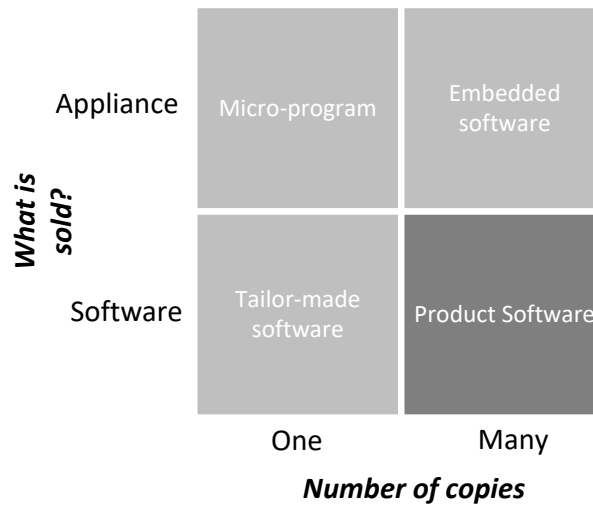


Figure 4-1. Software classification (Xu & Brinkkemper, 2007)

represent a coherent group of capabilities within a business function. Overall, the model explains the key competencies that should be fulfilled by stakeholders in software product management

Bekkers, van de Weerd, Spruit, and Brinkkemper (2010) present a software product management (SPM) competence model as a comprehensive overview of all important areas of software product management (Figure 4-2). The model was developed from an SPM reference framework by van de Weerd, Brinkkemper, Nieuwenhuis, Versendaal, and Bijlsma, 2006. The competence model depicts the interaction and information flows among stakeholders through four business functions: requirements management, release planning, product road mapping, and portfolio management. Each business function is elaborated with several focus areas, which represent a coherent group of capabilities within a business function. Overall, the model explains the key competencies that should be fulfilled by stakeholders in software product management.

In this research, we used the SPM competence model to help us identify the situational factors (SFs) in global software engineering, which could influence product management focus areas and stakeholders in a product software organization seen in the key concepts found during the literature review. A situational factor is “any factor relevant to product development and product services” (Bekkers, Spruit, Van de Weerd, Van Vliet, & Mahieu, 2010, p.43). A situational factor can be a particular method fragment that is not based on an established software engineering approach. We explore the situational factors that affect task coordination in global software engineering processes performed by software companies within the framework of software product management.

There are several characteristics of product software based on the definition provided by Xu and Brinkkemper (2007).

**Packaged Components.** “Packaged components” denotes software code, binaries, and executables. Product software, as the integration of modules and components, can also be perceived as a packaged set of standard components that can be configured to satisfy particular needs of customers without having to change the line of codes (Bertram, Schaarschmidt, & Von Kortzfleisch, 2012; Mantyla & Vanhanen, 2011).

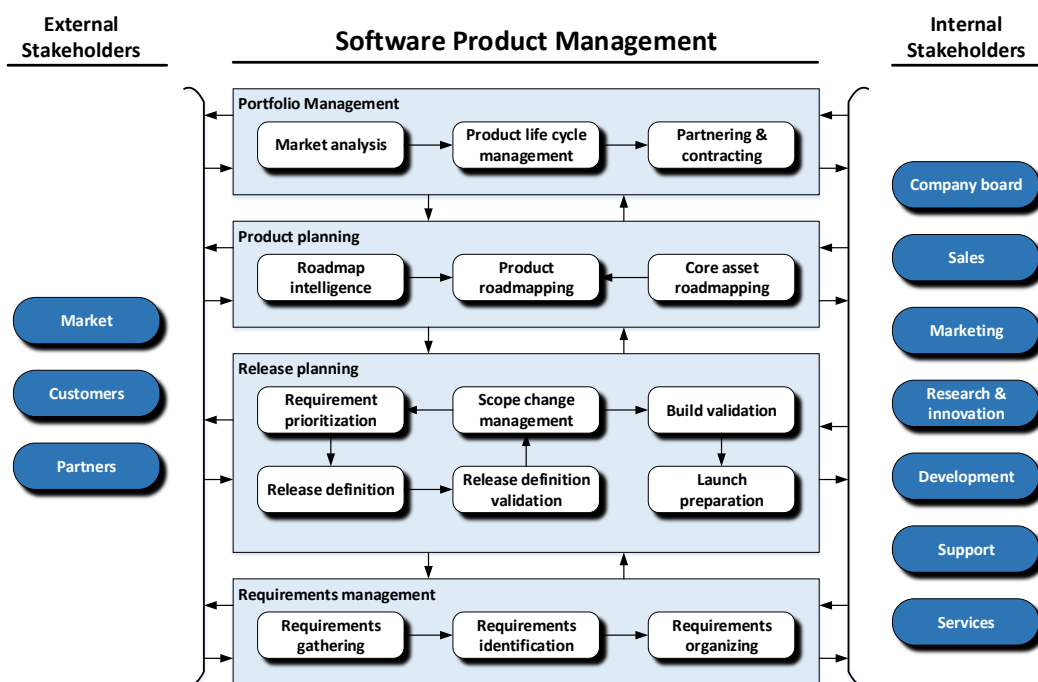


Figure 4-2. Reference framework for software product management (Bekkers et al., 2010, p.4)

**Auxiliary materials.** Auxiliary materials refer to product documentation, web pages, user manuals, training materials, and brochures in which they are owned, managed, updated, and released by the vendors together with product updates (Xu & Brinkkemper, 2007).

**Software-based services.** As software that is created for a market, product software can be on-premises software or software as a service (SaaS). On-premises software is installed and run on the premises (building or hardware) of the customers who are using the software, which refers to software as packaged components. Meanwhile, SaaS, or what was earlier known as an application service provider (ASP), is “s time and location independent online access to a remotely managed server application, that permits concurrent utilization of the same application installation by a large number of independent users (customers), offers attractive payment logic compared to the customer value received, and makes a continuous flow of new and innovative software possible” (Sääksjärvi, Lassila, & Nordström, 2005).

The key element of SaaS is the online characteristic, which enables consumers of the service to use the software application anytime and anywhere around the world, as long as they have a computer with a web browser and Internet access. This characteristic is revolutionizing many aspects of product software, such as business models, deployment models, cost structures, revenue logic and licensing schemes, the focus of clients as well as service providers, architecture, and competencies. It does not require complicated hardware or software on the customers' side. As a service, software is used on demand through a time subscription or a pay-as-you-go model. Service providers are also challenged to provide secure and reliable architecture because they provide only one instance to be accessed by many customers concurrently. Software can also be updated and upgraded at the same time, which reduces the time to release,

deliver, and deploy patches or updates (D'souza, Kabbedijk, Seo, Jansen, & Brinkkemper, 2012; Reuwer, Jansen, & Brinkkemper, 2013; Sääksjärvi et al., 2005).

**Product line and release management.** “Product line and release” is related to a product software’s commercial value. It encompasses the activities of product release to the market, deployment activities (including integration with other applications and customizations), and routine maintenance services. The origins of requirements clearly become the differences between tailor-made software and product software. Requirements for the first release of product software are usually not as clear as tailor-made software since the requirements are derived from perceived deficiencies in the marketplace, market trends, or potential customer interviews. Thus, product software requires careful release planning and prioritization of requirements (van de Weerd, Brinkkemper, Nieuwenhuis, et al., 2006; Xu & Brinkkemper, 2007).

**Market driven.** Vähäniitty (2006), as well as Xu and Brinkkemper (2007), mentioned that product software is a market-driven project undertaken to produce standardized software. To make a successful product, delivering the right product at the right time to the right market is essential (Artz, van de Weerd, Brinkkemper, & Fieggen, 2010). Product software can be an evolution of tailor-made software. When companies consider entering broader markets by transforming their tailor-made software into product software, specific activities should be performed. They should elicit and analyze the broader requirements information by monitoring the market trends, deploying a product management strategy, defining the product lifecycle and release planning, and establishing partnerships and contracts (van de Weerd et al., 2006). Small to medium-sized companies should be grateful for the presence of the Internet, which makes it possible for them to enter broader markets by exporting their products as Software as a Service (SaaS).

The study performed by Reuwer et al. (2013) showed that even though internationalization brings greater opportunities, companies should establish a strong domestic market position as a solid base. They also should consider that physical and cultural differences should be accommodated for internationalization processes such as understanding targeted countries’ behavior, providing multi-language packs, and ensuring legal compliance in targeted countries. Companies can engage in partnerships with local companies or build remote offices as their organizational strategies to perform these activities (Bosch & Bosch-Sijtsema, 2010b; Reuwer et al., 2013).

1. Platform. As already mentioned, product software can be either on-premises software or SaaS, which extends the capabilities of product software and allows companies to provide services not only as ready-to-use software such as Microsoft Office. SaaS can also be used to provide infrastructure that, in turn, provides computing resources like Amazon Web Services. In addition, SaaS can also be used to provide an application platform to serve specific business processes that can be configured to accommodate particular business processes, such as Salesforce.com (D'souza et al., 2012; Kang et al., 2010).
2. A software platform and its architecture should be defined in its software product line (SPL). Software product line engineering defines the software core, reusable assets, and the development process of software as the actual products. The reusable assets can be modeled as a set of features that represent the commonalities and variabilities of the products to satisfy stakeholders’ technical and non-technical requirements alike.

Therefore, software variability management should be established to maintain product platform features, product functional and technological architectures, and product line configuration and its derivation processes (Brisaboa, Cortiñas, Luaces, & Pol'la, 2015).

3. **Business model.** The last concept that we found during the literature review is “business model and licencing.” D’souza et al. (2012) revealed that on-premises and SaaS products need different kinds of partnerships. On-premises products need partnerships for distribution channels, including consultancy, sales extensions, and training services. Meanwhile, SaaS products need partners as their co-creators who can add value to the products, such as add-ons and third party connectors, or provide them with infrastructure as a service (IaaS).

For customers, there are also different types of licensing. In on-premises licensing, software companies usually sell their products with a pay-per-user licensing mode, which can also be per instance, such as for database products, or per CPU core in appliances software. In SaaS, software companies can provide licenses in more different ways, such as pay-per-user, pay-per-use, or pay-per-feature. In terms of software and data ownership, in on-premises software, customers are the owners of the software and the data. Meanwhile, in SaaS, companies, as the service providers, are responsible for the software reliability and data security (D’souza et al., 2012). These situations imply that different competencies for the software companies and their vendors are needed, based on the choices of product software technology and business models. Thus, communication between software companies and their vendors is expected to encourage more peer-to-peer and partnership-oriented collaborations, which can bring a shared, collective understanding of the domain knowledge, technology, and business needs (Smirnova et al., 2014).

Based on those characteristics, compared to tailor-made software, product software development processes become more complicated, need more competencies, have a larger development scale and a broader target market. Thus, many companies consider acquiring more resources and competencies by using global software engineering (Ågerfalk et al., 2008).

## 4.2 Challenges and Practices at Product Software Companies

In this section, we present the results of case study interviews at five product software companies in the Netherlands. Four of them are performing global software engineering by having remote offices or partners in other countries; one of them is not, due to several reasons. The company profiles are introduced in Appendix B and reference for the coding scheme is provided in Appendix D.

### 4.2.1 AlphaSoft [CA]

The interviews at AlphaSoft were conducted with two different roles, which are the Scrum Master and the Unit Manager. A scrum master and a unit manager in AlphaSoft are organized to be responsible for only specific types of products, such as retail and wholesale software as well as point of sales software.

AlphaSoft has been performing global software engineering for around six years. The remote office is responsible for development and testing. They started their first nearshoring by acquiring a company in Belgium. They started to build an office in Romania under the Belgium office’s management when they found that Romania offered a large number of high-tech savvy and enthusiastic fresh graduates with lower salary grades compared to Belgium and the Netherlands. This country was selected because people in Romania also have good English

skills and a similar working culture. Now, the remote offices have around 40 engineers who are responsible for development and testing. Related to software production, the Netherlands office oversees the product design, project management, and legacy DevOps activities. The needs of young engineers are also considered by the fact that software technology changes rapidly. That is why AlphaSoft has a clear segregation of expertise for the remote offices. The engineers in the Netherlands have the skills and knowledge for the maintenance and continuing development of the core modules, which are built with old technology. Meanwhile, engineers in Romania work with the latest technology to adopt the changes among the clients, such as enhancing the UI/UX, applying a new platform, providing the mobile capabilities, and expanding the product as a service.

### **Challenges**

Having the benefits of only one hour of time difference and the ease of communicating where the English proficiency and the dialect are similar does not mean that this company is not facing any coordination challenges, as expressed by the Scrum Master in the interview data (iv-a-1):

*“...every distributed work—this is the challenge; you don’t see each other in person often, so you hardly know each other. The language is not a native language for both sides, the Romanians talk English and the Dutch talk English as well but it’s not native, so that’s always a challenge (iv-a-1).”*

Another challenge occurred when the Romania office was opened. Some doubts and distrust in terms of the ability of new employees appeared. Employees in Belgium also felt their positions threatened by the presence of new, younger employees. With the passage of time, team cohesion began to emerge by itself.

Between Romanian and Dutch employees, there are slight cultural differences. Dutch people are mostly more outspoken, Belgians are not very talkative, and Romanians are somewhere in between. The historical background of Romania as a country also affects the behavior, where employees in Romania have more respect for organizational hierarchy, while the Dutch have more open and horizontal communication to anyone beyond the organizational hierarchy, whatever the level of their position as can be inferred from the comment from the Scrum Master as below:

*“Then, we have a cultural problem. Maybe you have seen it well. Maybe the Netherlands is more outspoken, bolder, more aggressive, and other countries are more teammates and respect the hierarchy more (iv-a-1).”*

In addition to the cultural differences, the Scrum Master still feels that the distance becomes another challenge since she prefers a direct and intensive communication. The sense of not working in the same location and the inability to see people’s expressions directly during communication is perceived as a less convenient situation for team members.

Before the remote office in Romania was built, AlphaSoft also established a partnership with a company from India to share the development and testing phases. Unfortunately, the partnership did not work well. The language problems, working culture, and especially the huge time difference (five hours) raised concerns at AlphaSoft that could not be compensated by the cheaper labor supplied by the company from India.



### *Coordination Practices*

Regarding the product software engineering processes, as can be seen by the existence of the Scrum Master role, AlphaSoft adopted scrum methodology to oversee the software development process. Obviously, as stated by the Scrum Master, the scrum process model helps the company address communication problems by making the communication chain more concise.

*“...normally, we waited for the development after the design was finished as a whole, but the design never finished. So, now, because we have everybody in the team, the communication is shorter (iv-a-1).”*

The Scrum Master can work with more than one scrum team. There are no specific additional processes in the scrum experienced by this company related to working with distributed teams. Since the distributed tasks occur in the development and the testing stages, the daily stand-up meeting becomes the most important event where the coordination activities occur. The daily stand-up is arranged to be done in 15 minutes and should be attended by all development team members and led by the Scrum Master. The daily stand-up starts with a check of the progress of tasks. A scrum board and a burn down chart are presented so that all the attendees have the same view of the current state of each of the tasks. Each team member explains their progress and reports if any impediments occurred. Other team members can give feedback or share their thoughts to help other members. After the stand-up meeting, the sprint backlog should be updated by the team, including the product owner. The product owner is responsible for ensuring that the information in the product backlog is up to date. The product backlog centralizes all the project information and task statuses, which all the team members should review regularly (iv-a-1). The Scrum Master elaborates her experience as below:

*“This is done by the team including product owners through Skype or Lync. Product Owners need to have the backlog up to date, so, therefore, they’ll walk through the list regularly. After the planning meeting, preferably, the sprint backlog items do not change. The Scrum Master will see to this. In TFS (Microsoft Team Foundation Server), the team provides tasks per product backlog item. We use SharePoint to save our documentation. In the PBI, we make sure the link to SharePoint is saved in the PBI (iv-a-1).”*

Other mandatory meetings, such as the sprint review meeting and retrospective meeting, are also used to discuss any issues raised during the current iteration process. For example, people might discuss the needs of specific knowledge, skills, or training sessions for the team members to work on their tasks; or, they might talk about the needs of mentoring or knowledge elicitation. The product owner leads these meetings with the help of the Scrum Master.

As the Unit Manager has mentioned in the interview, the position of unit managers becomes important since they should bridge the gap between the head office and the remote offices.

*“I bridge the gap with more senior management to the team to make sure the teams who are working at NL, Belgium, or Romania should have the same information. That’s why I should travel a lot to Romania to get some information from them and share some information so they get the same information equally (iv-a-2).”*

To build team cohesion, the Unit Manager also travels (from Belgium) to and stays in Romania for a week every month. The purposes of this proactive site visit are (iv-a-2):

1. improving team cohesion by increasing informal communication with the remote team members,
2. revealing and solving untold issues in the daily stand-up or retrospective meeting,
3. reviewing remote office situations to be reported to the head office,
4. updating information and knowledge from the head office,
5. bridging communication between remote offices and the head office, and
6. influencing the team members at remote offices with the working culture and behavior of the head office.

This mechanism is proven to blend the interaction between the head office and remote offices over the cultural and behavior issues, expand boundaries, and accelerate the communication and information transactions between remote locations. The coordination practices by the Scrum Master and the Unit Manager is then depicted in Figure 4-3.

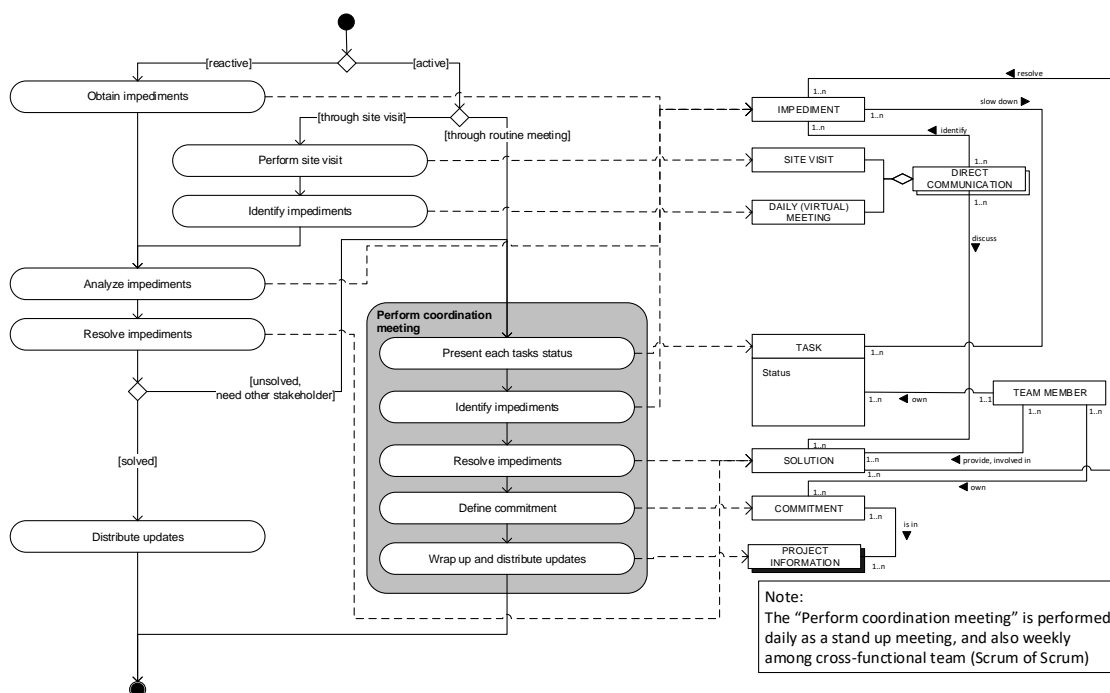


Figure 4-3 PDD of task coordination approach by AlphaSoft

Based on the coordination mechanism by Mintzberg (1980), AlphaSoft uses a mutual adjustment mechanism where the product owner and scrum master facilitate the team members with information and intensive meetings to facilitate communication between team members. As expressed by the Scrum Master, each team member should self-organize him or herself when discussing topics with others, including with the Scrum Master (iv-a-1).

1. *“Scrum has mandatory meetings like sprint review, sprint retrospective, sprint planning, sprint refinement and daily scrum every day for a maximum of 15 minutes. Furthermore, the team has the ability to use Skype or Lync for further communication if needed. The team is responsible for talk to each other if functionality needs to be discussed.”*
2. *“The Scrum Master is the office manager or team leader. If team members don’t attend, this will be discussed either during the retrospective or personally.”*

Since they are not experiencing challenges in terms of time differences, communication is done mostly in a direct way, which helps them avoid misunderstandings and delays in communication. It is not clear how knowledge is transferred among team members during project execution. Considering that they have smooth and direct communication, it is believed that tacit knowledge is exchanged by means of socialization. Team members can learn by practicing, or they become “socialized” into a specific way of doing things from peers (Smith, 2001). The stored knowledge is information that is explicitly formalized as project documentation and stored in TFS and SharePoint, which is accessible by all team members.

**Supporting Infrastructure and Tools**

AlphaSoft did not have to prepare the specific infrastructure because the existing infrastructure has already been supporting the communication and collaboration among team members. The communication mostly is done through a synchronous mechanism. They use Lync (Skype for Business) for direct communication, SharePoint to save and share project documentation, and TFS to support collaboration and project management. These tools are able to be integrated each other.

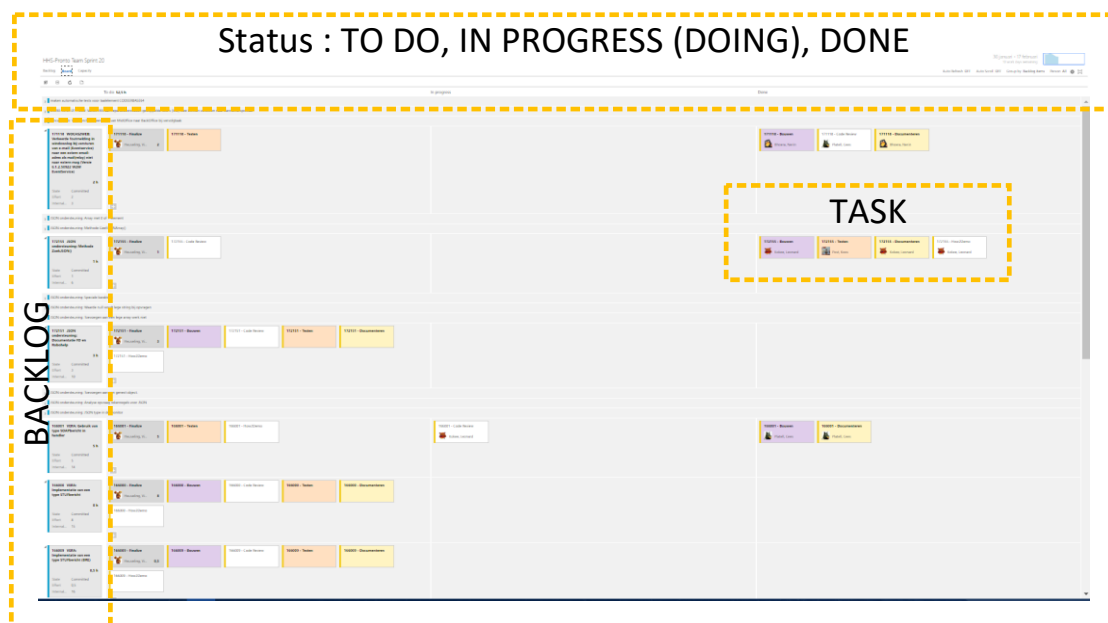


Figure 4-4 AlphaSoft’s Scrum Board

They use product backlog, a burn down chart (Figure 4-4) and scrum board (Figure 4-5) to provide a single view of project status. As already mentioned before, the product owner is the one who has the responsibility to make sure that the information provided is up to date. When a team from Romania is involved in a project, they use video conferencing software in the daily stand-up and the retrospective meetings, so everybody can see and meet each other.

**Important Roles and Functions**

The product owner, the Scrum Master, and the Unit Manager are the most important roles in managing coordination when performing software engineering projects globally at AlphaSoft. They hold crucial functions differently, which are as follows (iv-a-1, iv-a-2):

1. Product Owner
  - a. Facilitate knowledge sharing by providing up-to-date, explicit information.

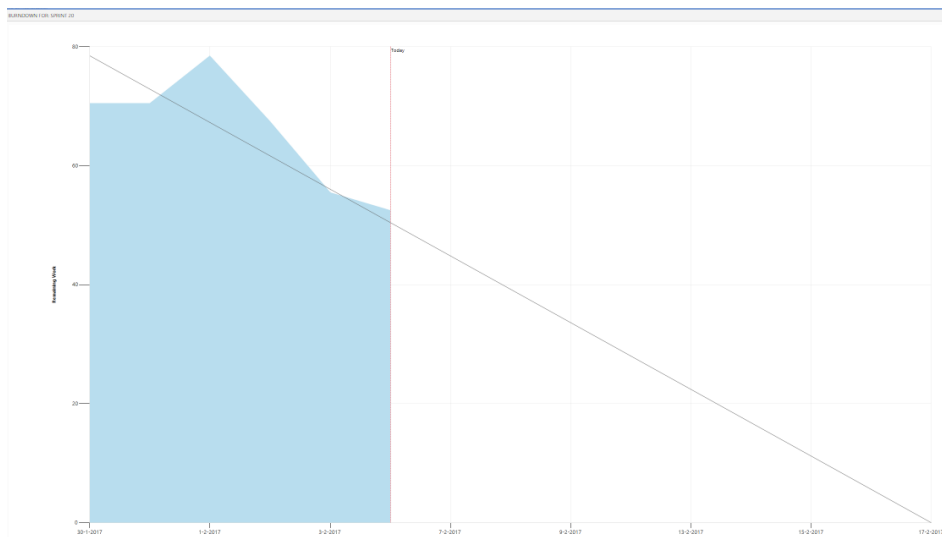


Figure 4-5 AlphaSoft's burn down chart

- b. Articulate the business needs into requirements in technical language that can be understood by the engineers, as well as translating the technical concerns and situations to the business units and management.
  2. Product Owner
    - a. Facilitate knowledge sharing by providing up-to-date, explicit information.
    - b. Articulate the business needs into requirements in technical language that can be understood by the engineers, as well as translating the technical concerns and situations to the business units and management.
  3. Scrum Master
    - a. Facilitate direct communication between team members in the periodic meeting.
    - b. Help the team in self-organizing the tasks.
    - c. Promote collaboration among teams and between teams for the product owner.
    - d. Shield the team from direct interruptions during the sprint.
    - e. Be the first officer to be found if the engineers are facing problems or encountering impediments.
    - f. Help the product owner in the project review and analysis at the planning and review meetings.
  4. Unit Manager
    - a. Become the boundary spanner who bridges the communication and knowledge transfer between the head office and the remote offices.
    - b. Be the representative of the product owner and the Scrum Master to make sure that the employees at the remote offices able to accomplish the tasks within the agreed time.

#### 4.2.2 BetaSoft [CB]

BetaSoft is a multinational company that has a network of business units in many countries. Our research focuses on an organization under BetaSoft that manages the research and development of global ERP solutions. Two interviews were performed with the Technology Director and the Product Manager. The Technology Director was assigned as the CEO of the remote office in Kuala Lumpur (KL), Malaysia. The KL office has been operating for almost

14 years, providing most of the engineering processes for the global solution product, such as product design, product realization, and product testing (iv-b-1).

BetaSoft has established its product engineering processes and has adopted scrum of scrum approach to define the product roadmap (Figure 4-6, iv-b-1). The process starts with a management team meeting, which is attended by all the discipline directors and the managing director. The management team discusses each discipline performance within the corporate strategy. The corporate strategy itself is defined every five years. Currently, the company has three main strategies: to become a cloud software company, to provide platform digitalization, and to create value-added services in business intelligence and data analytics. Then, a technology board meeting led by the Technology Director and attended by the product marketing director, the managing director, and sometimes the customer service director is held to discuss the roadmap preparation. The roadmap preparation meeting provides a chance to set yearly objectives, which define the subjective and objective directions for the technical team.

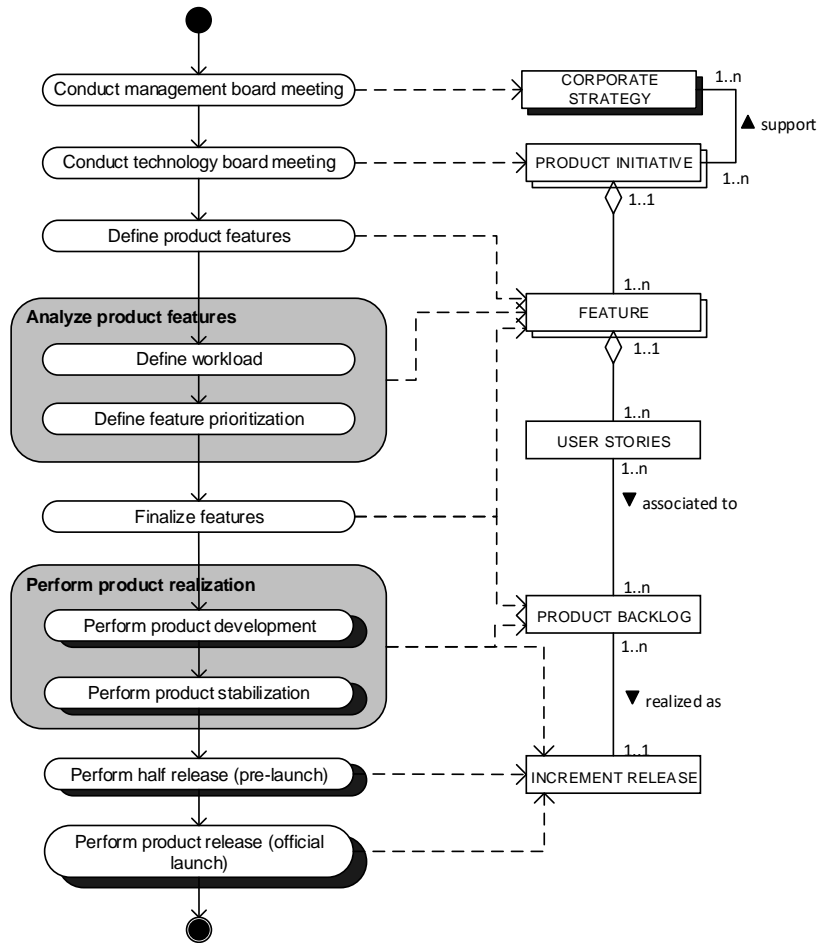


Figure 4-6 PDD of product engineering processes at BetaSoft

Thereafter, the technology board will call the Product Manager and the feature owners to prepare and engage with the development team. A feature owner is associated with a feature and should be involved in the feature development such as planning, scrum meetings, and demos (iv-b-1). Table 4-1 present the hierarchy of deliverables of the requirements definition (iv-b-1). For example, a set of features is represented as an epic which should be delivered in a product release. The development team helps the feature owners define the complexity of the

requirements and estimate the workload. These inputs from the development team are used by the Product Manager and the feature owners to maintain the priorities.

*Table 4-1 Types of deliverables of Scrum processes in BetaSoft*

<b>Deliverables</b>	<b>Description</b>
Initiative	Initiatives can be perceived as the high-level (strategic) requirements from business related discipline units.
Epic	Epic defines the workspace as a set of features that should be delivered in a release that represents several user requirements.
Feature	Feature is a collection of related user stories which cannot be completed in a single sprint.
User Story	User story is a work unit that can be completed in single sprint.
Task	The smallest unit of work that should be performed by the developer(s)

The feature owners and the Product Manager present the analysis results. The board then discusses technical problems and finalizes the roadmap. Next, the finalized features are put in the backlog as a sign that the development team can start doing their jobs. The technical implementation is designed to be performed within six sprints, where each sprint is run within three weeks. At the end of the development, they have another sprint for the product stabilization, which consists of alpha testing (performance and security testing), formal stabilization, and formal handover from the development team to the product management team.

Before the full release, the go-to-market team, which consists of the product management team, the customer service team, and the development team, performs a control release (half release) to check the product entrance readiness by testing the release with several customers from various types of industries for six weeks. At the end of the half release, the team presents the results and shares advice to the technology board. If the technology board is satisfied with the results, the product can be fully released to the market.

### ***Challenges***

BetaSoft has also standardized all the documents structure stored in the TFS; it is easier for the Product Manager to check and analyze the results that reduce the information ambiguity. The good level of fluency in English of engineers in Kuala Lumpur makes communication easier. BetaSoft has arranged the segregation of expertise to minimize the task dependencies between offices. The direct communication is complemented using asynchronous communication tools, such as email, which are used to eliminate the effect of time zone differences. That is why BetaSoft has not had any communication problems. Based on the interviews, only the quality of the communication tools, such as unclear connections, becomes a communication problem (iv-b-2).

Early on, organizational and local cultural differences became obstacles that limited communication flows (iv-b-1). Employees at the Kuala Lumpur office, as well as people in Asia in general, need to think before answering, and they choose to be quiet. This culture, sometimes, was not acknowledged by their colleagues from the head office. Also, technical people usually are not as extroverted as sales people, which made it harder for people from the head office to communicate with their colleagues at the KL office.

**Coordination Practices**

BetaSoft’s software engineering processes are started by the meeting of different stakeholders to discuss the product roadmap. The meeting is held three times a year to provide a list of recommendations and feedback about features. The information shared with the stakeholders is stored as PowerPoint files and saved in Microsoft TFS. Product managers and their team members then determine priorities for new developments to put on the roadmap with the consideration of the size and the complexity of the features. The team then prepares the more detailed requirements and then verifies the requirements with the stakeholders. The verified requirements then are sent to the development teams in Kuala Lumpur to be executed or implemented. Currently, BetaSoft is considering streamlining these processes and involving all the stakeholders in particular processes so they know the current situation and how to proceed (iv-b-2).

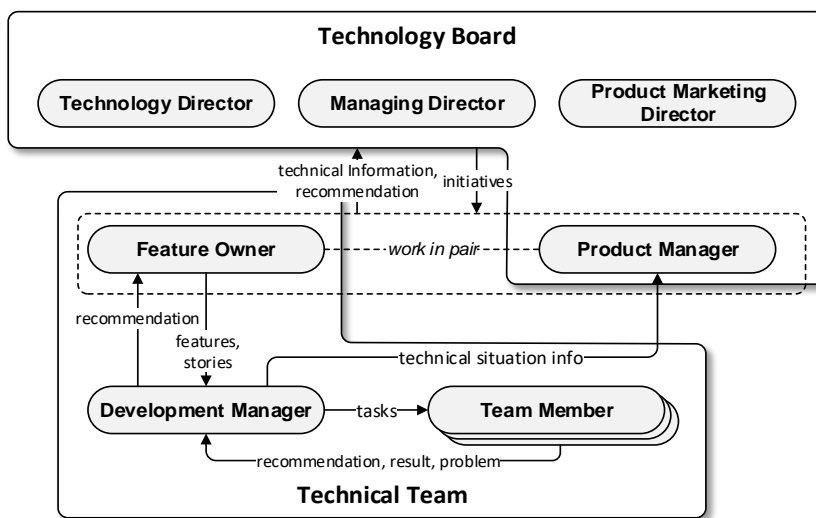


Figure 4-7 Functional diagram of two coordination areas in BetaSoft

To manage coordination between the product management team and the development team, BetaSoft uses scrum of scrum to accommodate the segregation of the business coordination stage and the technical coordination stage. The business coordination stage is the coordination between the technical board and the Product Manager to decide the feature design and release planning. The technical stage is when the Product Manager and the features owner discuss issues with the development manager to decide the development plan, the deliverables, and when the deliverables should be provided. The development team, led by the development manager, consists of product designers, developers, and testers. All the technical decisions, such as architectural design, are decided by the development team internally with the supervision of the Product Manager and the features owner. It makes clear that individuals in the roles of product manager and feature owner connect these two main coordination areas.

The coordination practices are shown where the technical team, led by the feature owner and the Product Manager, discusses feedback and recommendations from the business stakeholders. They share inputs from the technical team, clarify the requirements, and negotiate the urgency of the requirements based on the situation of the technical teams. By having a regular meeting, business stakeholders and technical teams can align their work (Figure 4-8).

To update all the team members with the newest information, all the documents in the form of slides, diagrams, worksheets, and documents are stored in the TFS. Each process owner

is responsible for the information updates in the TFS; for example, the Product Manager should update all the requirements documentation in the TFS.

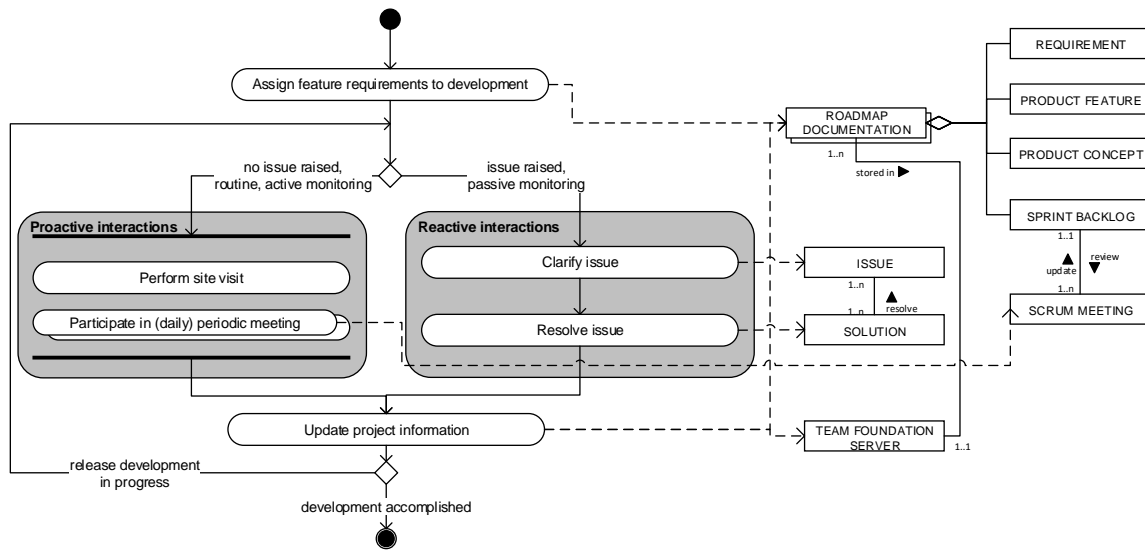


Figure 4-8 Coordination practices in GammaSoft

The TFS can also facilitate the communication process between the development team in KL and the customer service team in other countries. In daily activities, the development team works for the new features development and bug fixing. The customer service team can create a new work item to report a bug in the TFS, which the development team follows up on. The development team can either accept or reject the work item. The rejection is not caused by the inability to understand the description (language problem) but mostly because of the lack of information provided by the customer service team (iv-b-2).

During the development stage, the product manager does not interfere with the development processes (iv-b-2). Both teams, the technical team and the product management team, have discussed the deliverables and when the agreed tasks should be accomplished. In the meantime, when the development team finds difficulties in performing or accomplishing the tasks, the product manager should be notified by the development manager.

To minimize the cultural challenges, people from the head office often travel to the KL office. They can see and feel the hospitality of their colleagues, which influences how they communicate. This face-to-face interaction can help cultural socialization, which cannot be done remotely by using WebEx<sup>2</sup>. At the same time, the managing director of KL office pushed the culture socialization continuously to help team members in Kuala Lumpur adopt the cultural values at the head office, such as openness, freedom of expression, and horizontal interaction over the organizational hierarchy (iv-b-1).

### Supporting Tools Infrastructure

We can see that the roles of product manager, feature owner, and development manager are important in the coordination processes in this company.

<sup>2</sup> WebEx is an online meeting tool provided by Cisco. It provides virtual meeting with video support. But in our case study, WebEx is mostly used for teleconferences (voice-based).



1. Product manager. The Product Manager manages the long-term roadmap of the product and holds the central position between the technology board and the development team. More specifically, the Product Manager leads the collaboration with non-technical disciplines such as marketing, product marketing, and customer service.
2. Feature owner. Similarly, with the Product Manager, the feature owner provides the function of media-broker between management and technical teams. The main difference between the feature owner and the Product Manager is the engagement with the development process (shorter roadmap). The feature owner is engaged with the all the scrum meetings such as planning, stand-ups, and retrospectives. The feature owner also responsible for the development of user stories based on the defined features and for putting them in the backlog.
3. Development manager. The main function of the development manager is managing the execution of the tasks during the development process to realize the requirements within the agreed time. Related to the collaboration between the KL office and the head office, the development manager has a function to facilitate the communication between the development team, the feature owners, and the Product Manager.

#### 4.2.3 GammaSoft [CC]

We conducted two interviews with different business units in GammaSoft: the service delivery (SD) department, the organization that manages software used in internal GammaSoft, and GammaSoft itself as a holding company. The first interview was conducted with the SD department manager, who represents the SD department, and the second interview was held with the global data architect, who represents GammaSoft as a holding company.

Both have a commonality because GammaSoft and the SD department particularly perform software engineering by involving a partner company from India in product realization. More specifically, in producing software for clients, GammaSoft as holding company also assigns a partner from Poland to design its products. GammaSoft itself holds the responsibility of defining the mission and the vision of the business, which is realized as product requirements and standardization of the work methods.

#### *Challenges*

Compared to other companies that we have interviewed, GammaSoft is the largest company, based on the number of remote offices under its holding organization. There are 40 subsidiaries spread across various countries, and they are largely are local companies that were acquired by GammaSoft. Each subsidiary can develop and sell its own products. GammaSoft provides the standardization of work methods, especially for marketing and sales activities.

As a holding company, GammaSoft launched a flagship product named Business World, a cloud-based ERP service built through partnerships with two companies in Poland and India (iv-c-2). The partnerships were built to help them design and realize their products because of the following considerations.

1. GammaSoft needs to focus on the business to adapt to the rapid changes in customer needs and behaviors caused by rapid changes in technology.
2. Partner companies offer productivity in terms of resource availability and resource specialization with a competitive economy value.

Unfortunately, between the subsidiaries, there is no collaboration such as sharing the reusable software modules from subsidiaries in different countries (iv-v-2). Engineers in subsidiaries spread across various countries could be a potential resource for GammaSoft. In the long term,

engineers with diverse experiences, knowledge, as well as access to a wider market, can be used to build systems that are more flexible and powerful. To that end, there needs to be a synergy between subsidiaries to unify the opportunities and the resources. There is still a problem reflected in the interview where the organizational silos occur and limit the organization’s cohesion.

GammaSoft also faces a difficulty in managing the work with their product designer partner and developer partner (iv-c-2). The problem also started internally at GammaSoft itself; sometimes the company is not able to give clear requirements to its product designers. It has become apparent that unclear information causes the dependency on the main office to become high, which, in turn, increases the intensity of communication.

Another problem occurs with the developer partners. They do not experience considerable challenges related to communication and English language skills of employees of partners in India. However, cultural differences, technical experience, and process maturity levels resulted in the developer partner being unable to finish the tasks at the required time and quality. The developer partner is from a country where people are not used to saying “No.” The developer partner is also sometimes unable to guarantee the quality of the work as expected by GammaSoft. This problem is exacerbated by the small chance of synchronized communication because of the small overlapping working hours caused by the time zone difference. As the designer partner, these issues cause communication problems. Those challenges mentioned above have led to a gap in terms of communication and information transaction that interferes with the coordination between GammaSoft and its partners.

**Coordination Practices**

As GammaSoft wants to enforce segregation of duties by the expertise and capacities of each organization, this company does not attempt to monitor every detail of the activities carried out by its partners directly. Any review of the work is done according to the terms of the agreed time and deliverables unless the partners imply or communicate directly to the product manager or related staff in GammaSoft where there is a problem in the task execution. GammaSoft tries to localize task dependencies in the same area of competence: GammaSoft, as the product owner that manages the product portfolio (Figure 4-9). The product designer company’s primary duty is translating the business requirements and the vision of GammaSoft’s board of management into a product and features roadmap. Meanwhile, the product developer company is in charge of transforming all these designs into the desired product software and is responsible for the operation and maintenance of the software (iv-c-2).

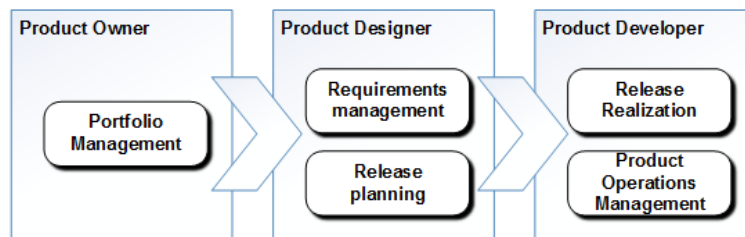


Figure 4-9 Segregation of tasks in GammaSoft

In the same way, for internal projects, GammaSoft also involves its developer partner for DevOps processes (iv-c-1). To have a good communication flow, the developer partner has an initiative to provide a service coordinator as a liaison officer in the Netherlands. The partner company also provides a small group of engineers to help service coordinator at the first level of problem-solving. The service coordinator becomes the communication broker between the

development team and service delivery manager, who represents the business users (Figure 4-10). Sometimes, business users want their queries to be executed as soon as possible, regardless of the planning done by the service delivery manager. They often bypass the service delivery manager and try to speak directly with the development team. Thus, the service coordinator also becomes the patron who protects the development team from direct intervention from business users.

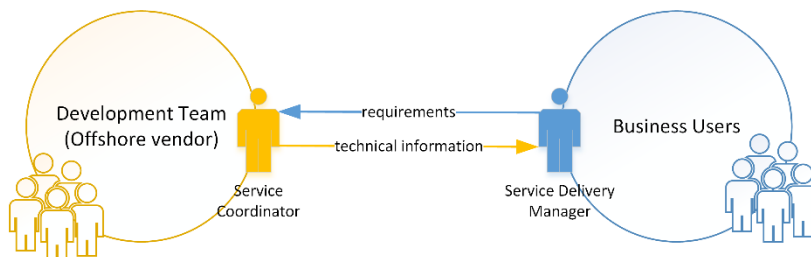


Figure 4-10 Service Coordinator as a communication broker

As presented in the introduction of this report, vendor software offshore generally offers cheap engineers and newly recruited graduates. They are usually hard workers, but they do not have the amount of experience in the real world sufficient to provide work results with the expected quality. Some senior engineers at GammaSoft have these qualities, but they could not be allocated for a particular system development task because it does not correspond to their job description. For that reason, the service delivery manager was a willing to conduct a knowledge transfer session for young developers to influence them by seeing the best practices demonstrated by senior engineers and applying these practices in their daily work (iv-c-1).

### Supporting Infrastructure

GammaSoft does not have any specific software engineering process model for managing the development process. It can be assumed that GammaSoft is following a traditional software engineering process model, which commonly consists of requirements, design, implementation (including development, testing, and deployment), and evolution. Thus, it is difficult to infer what kind of artifacts used to monitor the engineering activities, such as burn down charts in scrum or S-Curve Gantt Chart in PMBOK.

In terms of IT tools, GammaSoft also uses Microsoft SharePoint for the collaboration activities with their partner and OneVision for project management and internal collaboration. The reason behind the separation of these tools is the differentiation of the main feature of these tools (iv-c-1, iv-c-2).

### Important Roles and Functions

From this discussion, we can derive two important roles at GammaSoft that have a close relationship with global software engineering projects.

1. Service Coordinator
  - a. Acts as the liaison officer who provides the function of communication broker between the development team and the business users.
  - b. Covers the development team from direct intervention from business users.
  - c. Communicates the impediments from the technical side to the service delivery manager (or other line managers).
2. Line managers, such as service delivery manager and product manager

- a. Communicate the business requirements in technical language to the development team through the service coordinator.
- b. Communicate the technical considerations to the business users to balance the business pressure and the technical capacities

#### 4.2.4 DeltaSoft [CD]

The interview at DeltaSoft was performed with the research and development manager and one of the team leaders. The R&D department (R&D) is responsible for the product road mapping, research, development, and the operation and maintenance. The involvement of a software development partner in Romania was encouraged by the needs of additional human resources for the development process. The nearshoring partner has been working together with DeltaSoft for almost three years. The partner company was chosen by DeltaSoft because of the small time zone differences, the availability of resources, and their competencies. There is also a small consideration of the lower salary grades offered in Romania, but competency and the small time zone difference were the main considerations compared to the price (iv-d-1).

The R&D department consists of five teams: product owners, technical, business logic, configuration and testing, and documentation. DeltaSoft adopts a scrum process model for product engineering activities. Every iteration is supposed to be accomplished within two weeks. There are two types of scrum team: portfolio teams and architectural teams. Currently, there are two portfolio teams and three architectural teams. Each scrum team is a composition of one member from the product owner and four to six other members from different units. For example, Portfolio 1 (P1) consists of a product owner, two or three members from business unit, and two or three members from configuration and testing; Architectural 1 (A1) consists of a product owner, two or three members from technical and two or three members from configuration and testing. Specifically, Architectural 3 (A3) is the team that is working with the developer partner (Figure 4-11 and Figure 4-12).

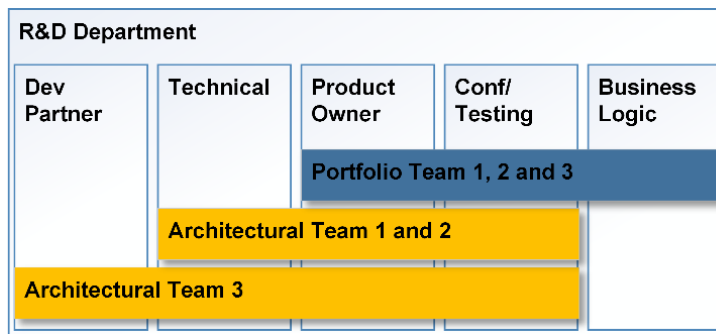


Figure 4-11 Team allocation in DeltaSoft

The R&D department develops one product, but it has the flexibility and capability to be delivered as various types of ERP systems, such as maintenance management and fleet management. They create a platform that enables portfolio teams to work in a domain-specific language to create business logics as modules within product software. Meanwhile, the architectural teams are responsible for creating the platform, system core functionalities, and integration with other software.

In producing the product software, DeltaSoft has defined a set of engineering processes: market analysis, requirements processes, feature design, and software realization (Figure 4-13, Figure 4-14, iv-d-1). Twice a year, DeltaSoft performs market trend identification where all the

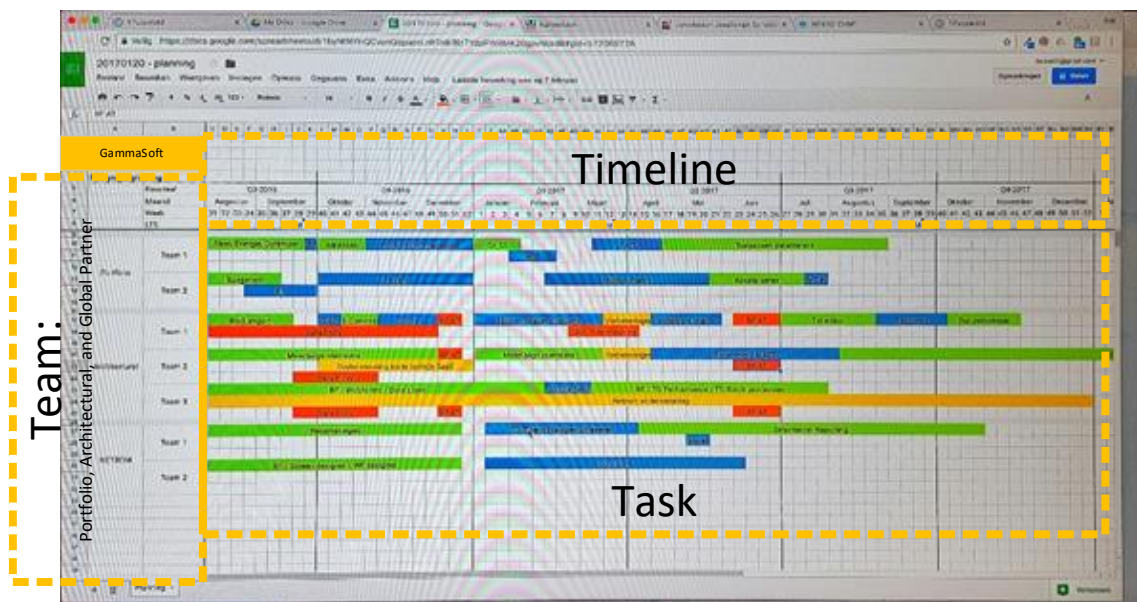


Figure 4-12 Task allocation for the Scrum Team

sales teams from the Netherlands, Belgium, and Germany hold a meeting to discuss competition trends, market trends, win–loss analysis, and competitor analysis. From this meeting, a list of market requirements is produced by using the terminologies based on customers’ context. Then, the market requirements are translated into product requirements that use the company’s own terminology. The product requirements describe the conceptual solution, which consists of features’ descriptions using a user story format. Afterwards, these conceptual solutions are discussed by the feature team to select and prioritize the requirements and then arrange them as product roadmap. The product roadmap is used by the R&D team as the reference to design, develop, and release the software.

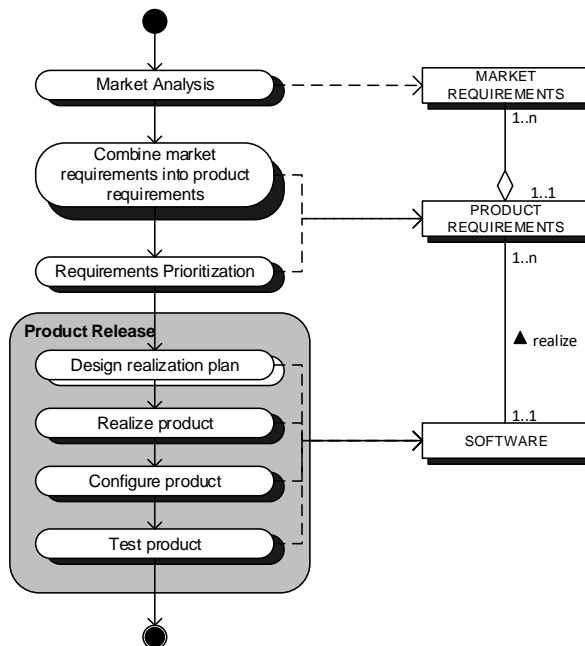


Figure 4-13 PDD of software engineering processes at DeltaSoft

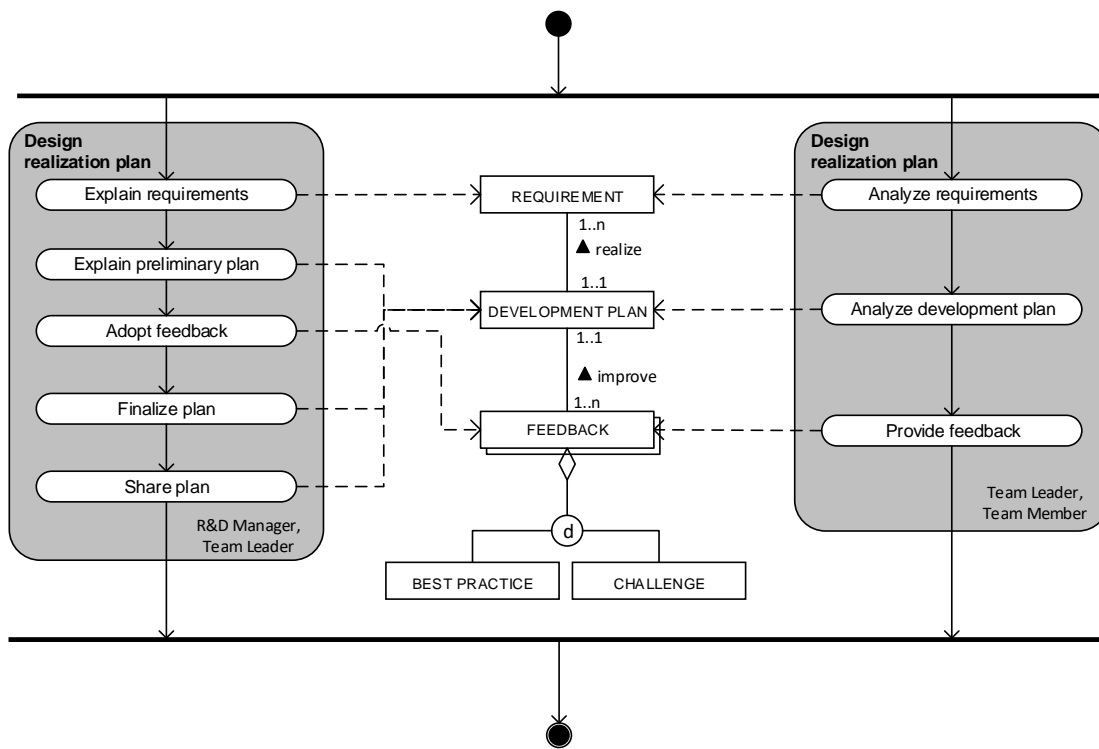


Figure 4-14 PDD of designing realization plan at DeltaSoft

**Challenges & Coordination Practices**

As seen in the above discussion, DeltaSoft applies scrum principles in its software engineering processes, such as organizing a scrum team as a matrix unit that involves business people (represented by the product owner) and engineers from various teams (iv-d-2). The Integrated development environment (IDE) prepared by the architectural team showed that DeltaSoft pays attention to technical excellence and supports the agility of the development process. DeltaSoft has established a set of engineering processes that apparently enable DeltaSoft to deliver product software gradually and frequently.

The interaction between DeltaSoft and its developer partner occurs only in the scope of software development. Not all the scrum teams in DeltaSoft work together with the developer partner team from the perspective of the organizational structure—only the Architectural 3 (A3) team that has a task to support the members from the developer partner (iv-d-2).

DeltaSoft chose to position the partner as part of the team. DeltaSoft’s partner is a company that specializes in helping other organizations in software development and implementation. They not only serve DeltaSoft but also help other companies from various industries. Six people appointed by the developer partner to work for DeltaSoft should be considered members of DeltaSoft (iv-d-1). To that end, DeltaSoft realizes that informal communication is the best way to build team cohesion. DeltaSoft also expects each team member’s openness so that the use of avatars and name aliases is avoided. Direct communication through Skype and phone calls is more preferred than indirect or asynchronous communication. The situation also becomes easier by the difference in time zones, which is only one hour (iv-d-2).

To build a bond between team members, DeltaSoft also does regular physical meetings through regular visits to Romania or vice versa. The Romanian members are also involved in meeting planning software as well as in the review meeting. DeltaSoft encourages their team

members, including the Romanians not only participating in tasks execution, but they are also supposed to provide input in product requirements analysis. As indicated by the team leader, they are free to express their thoughts, ideas, and feedback even to the R&D manager when they feel that their inputs are valuable, could bring improvements, or could challenge the existing approaches.

*“Everyone, including our friends from Romania, can talk to me or to the R&D manager if they have ideas. They can suggest something if they knew something better (iv-d-2).”*

In work collaboration, they use Slack to facilitate communication and file sharing among each other interactively. Information assets are available as tacit information kept by each employee. Transactions resulted in socialization of information through direct communication or physical meetings and partly (generally in the form of project documents) in externalization through media collaboration. Unfortunately, information externalization (e.g., best practices) as a formal knowledge structure to be accessed online is not yet optimized. However, by the help of intensive direct communication, information, or the knowledge gap caused by the absence of external knowledge can be minimized.

### ***Supporting Tools and Infrastructures***

As mentioned before, there are several tools that are employed to support task coordination. Slack is used as a collaboration tool where discussion and project artifact sharing (e.g., documents and reports) are performed over this online system (iv-d-2). They also use Skype for direct communication between remote locations. Moreover, the use of a standard development environment built internally also encourages task execution better because they work in the same setting and terminologies.

### ***Important Roles and Functions***

There is no particular role with the power to address issues that arise in global software engineering at DeltaSoft. Coordinating activities are more commonly pursued to be done informally within the virtual team. Coordination mechanisms between the two locations benefit from the short distance and the small time difference between the teams and DeltaSoft’s culture, which does not see the vertical structure as a barrier to coordination.

#### **4.2.5 ZetaSoft<sup>3</sup>**

The interview at ZetaSoft was conducted with the platform manager. Previously, he was the Product Manager of the company’s ERP product. Currently, ZetaSoft is focusing on the development of its future product, which is a revolutionary product that will replace all the current products into a single solution. The main idea of the new product is to create a flexible software generator run on the cloud that can build many ERP solutions, such as financials, sales, and logistics transactions processing for many types of industries. The CIO raised the vision of the product, and he, as the platform manager, should be able to provide a concrete ideation of the product.

---

<sup>3</sup> ZetaSoft does not do software engineering globally. The case of this company is presented to contrast the challenges experienced by global software engineering companies (AlphaSoft, BetaSoft, GammaSoft, DeltaSoft)

ZetaSoft is not a company that runs software engineering globally. As expressed by the platform manager, ZetaSoft does not believe that global software engineering could help the company improve their software engineering processes or even to reduce the engineering costs.

*“No, never, really never. Simple, because we don’t believe them. Same story with Romania. We don’t believe in outsourcing or nearshoring. I think eight out of 10 projects prove that ... you need so much more time, extra additional communication, explaining to people, traveling there and back, checking everything back (iv-z-1).”*

By having everyone under just one roof, employees can talk to one another to discuss problems, ask for someone’s help, or share information directly without any problems in time zone management.

In this organization, employees are not following a specific software engineering approach like scrum or XP for the previous product. However, for the future product, they have adopted some scrum practices, such as daily stand-up meetings. Meetings with software architects, development managers, and documentarists are also held regularly. As the platform manager, he plays a central role and performs several functions related to the engineering and project management processes.

Even though ZetaSoft does not do software engineering globally, the company is recruiting employees from foreign nationalities as well as and PhD students who are participating in research and projects (iv-z-1). As long as the international members can manage to learn the Dutch language and are eager to learn, with the help of work collaboration, they will have no difficulties performing their tasks.

Currently, there are two major teams: a team that works on the configuration and operation for the existing product and another team that works on the development of the future product (iv-z-1). In some cases, such as UI integration, there will an overlap where coordination between those teams is needed. The initiatives of the new future product, such as the use of a web interface and advanced architecture, a completely cloud-based system will need new libraries to be used by the current UI. They do not want to change the major UI functionalities because the system is closely related to the user experience of their 10,000 existing customers in the migration process.

### ***Coordination Practices***

ZetaSoft has a low hierarchical structure that speeds up coordination. The organizational structure helps ZetaSoft cut the communication chain, which mostly happens in large companies that run product software engineering projects (iv-z-1). During the development process, the platform manager, as well as the other team members, can discuss any topics with one another directly. Developers may come to the development manager first when they have technical problems, and if they have problems or something to be discussed about functional matters, they can come to the platform manager. When the platform manager needs processes to speed up, the involvement of the platform manager with the development team is strict and intense. The development team can self-organize what they think it is good to produce the requested deliverables. However, when, in the final development, the platform manager finds that the deliverables are not as designed or that the development team could not produce the deliverables within the agreed deadline, he should report it to the board of management.

The communication mechanism in ZetaSoft also can be done anywhere and at any time. When people need help, but their colleagues are not at their desks, they use Lync (Skype for



Business) to chat or make a call. For the future product project, they use Slack as their internal wiki where they can communicate, collaborate, and transfer files. Meanwhile, for the existing products, SharePoint is used as the knowledge-sharing media. The use of Slack for the future product development coordination platform can be perceived since Slack combines a collaborative workspace and communication media into a single platform that can promote intensive communication.

### ***Important Roles and Function***

Based on the discussion above, we can infer that the Product Manager (or the Platform Manager) plays an important role. This role's main function is connecting and managing expectations of the board of management and the technical team by articulating the board's vision and expressing technical concerns to the board. The Product Manager has several functions, such as

1. translating management's (the CIO's) vision into descriptive requirements;
2. proposing the requirements to the product designer/architect by
  - a. conducting brainstorming meeting about this project regularly with the designer/architect and the development manager and
  - b. finalizing product design and specification for the functionalities;
3. forwarding the validated requirements to the development manager, tester, and documentarist;
4. overseeing the development process; and
5. managing the overall project timeline.

### **4.3 The Interviews' Summary**

These investigation interviews confirmed that the key aspects of task coordination in GSE exist in the participating product software companies. They assure that communication, project control, and knowledge sharing are practices that can be applied to coordinate tasks and dependencies among tasks on teams that are globally distributed. The limited time of collaboration because of the time-zone difference and the difficulty in having co-located collaboration become the issue in the GSE, but these companies are mostly experiencing GSE challenges caused by of social background differences and expertise gaps. Companies that have been able to define the product engineering software process can perform distribution and control tasks better, also manage a more structured communication.

Furthermore, in product software companies, organizational strategies derived in the form of organizational structure design, roles determination hold important influence in the control of coordination both at the technical level and at the strategic level. Some specific roles such as product owner, product manager, scrum master, unit manager, and service coordinator are exposed as facilitators of communication, collaboration, and information transactions. The interviews also revealed some tools that can facilitate and catalyze coordination practices.

The following chapter will synthesize the results from the interviews from this chapter and the literature review presented in Chapter 3 as the foundation for the preliminary version of the Global Task Coordination method construction.

Table 4-2 Task Coordination Practices by the Participating Companies

Aspects	AlphaSoft	BetaSoft	GammaSoft	DeltaSoft	ZetaSoft
<b>Remote office / vendor location(s)</b>	Belgium, Romania, India*	Malaysia	Poland, India	Romania	Does not perform GSE
<b>Years performing GSE</b>	≥ 6 years	±17 years	± 2 years	± 2.5 years	-
<b>Processes at RO</b>	Development, testing	System design, development, testing	System design, development, testing	Development	-
<b>Team size</b>	40	≥ 100	≥ 100	6	40
<b>Software eng. process</b>	Scrum of Scrum	Similar to SAFE	Traditional	Scrum**	Traditional, Scrum**
<b>Target companies</b>	Dutch companies	Global	Global	Global, internal	Dutch companies
<b>Product</b>	ERP (Retail, Wholesale)	ERP	ERP	ERP (Supply Chain)	ERP (Finance)
<b>Challenges</b>	Communication, trust, temporal	Communication tools quality	Communication, expertise imbalance, temporal, organizational silos, culture	Lack of explicit knowledge	
<b>Communication mechanisms</b>	Both direct and indirect; Site visit	Mainly indirect	Mainly indirect	Mainly direct	Direct
<b>Control mechanisms</b>	Proactive; Mutual adjustment	Reactive; Standardization, mutual adjustment	Reactive; Direct supervision	Proactive; Standardization	Proactive; Direct supervision
<b>Knowledge sharing mechanisms</b>	Document sharing; Site visit	Formal training, document sharing	Mentoring; Document sharing	Pairing	Document sharing
<b>Roles</b>	Scrum Master, Unit Manager	Product Manager, Feature Owner, Development Manager	Service Coordinator	Team Leader	Product Manager
<b>Tools</b>	Burn Down Chart, Scrum Meetings, Skype, Ms TFS, Sharepoint	Skype, WebEx, Ms TFS	OneVision, Skype, Ms TFS	Skype, Sharepoint, Slack	Ms TFS

\*\* Not fully adopted or just similar with Scrum

# Chapter 5      SUMMARY OF STATE OF THE ART

In this chapter, we summarize the results from the literature study as presented in Chapter 4 and the interviews as reported in Chapter 5. The end goal of this summary is to identify how these key concepts are incorporated to be the foundation for the method construction in the next phase.

## 5.1 Interdependencies in GSE

Coordinated outcome is a condition in which the organization can harmonize dependencies among the distributed teams. By the time this condition is reached, a smooth task handover will be obtained, and all teams have the competencies and resources needed to perform the task. Interdependence occurs when actions taken by one referent system affects the actions or outcomes of another referent system (McCann & Ferry, 1979). Our findings in types of dependency in global software engineering projects are consistent with the dependency taxonomy proposed by Strode (2016) that defines task dependency from three points of views: Resource dependency, process dependency, and knowledge dependency.

### 5.1.1 Resource Dependency

This type of dependency occurs when a task could not be started immediately or accomplished due to the absence of an artifact. The artifacts can be concrete artifacts such as persons, location, tools, or software components. Strode (2013) mentions two forms of resource dependency: Entity dependency is a situation where a situation where a resource is not available which affects to the project progress. In AlphaSoft, the absence of Scrum Master can break the formal communication among dispersed team member. It also can be happened in GammaSoft if the product designer cannot provide the product requirements on time to the development team. Meanwhile, technical dependency occurs when a technical aspect of development (such as the absence of one software component) affects project progress. Technical dependency can happen in DeltaSoft since team members are working as a virtual team for the same application. The missing of a module that should be provided by the remote team can delay the overall project.

### 5.1.2 Process Dependency

The situation where a task cannot be executed before a previous task is accomplished is called process dependency. This type of dependency causes other team members to either switch tasks to keep their workflow or wait until the previous task is complete. Based on the categorization of interdependency provided by Malone and Crowston (1994), process dependency can be either sequential interdependence or reciprocal interdependency. Sequential dependencies can be defined as the above dependency process definition, in which some activities depend on the completion of others before beginning. While reciprocal dependency occurs during each process requires input from each other to run. For that reason, process dependency is strongly associated with the resource dependency because generally, it occurs when a process requires input from the previous process.

In our case study results, process dependency also appears along with the resource dependency. In a case of GammaSoft, development partners will not be able to begin the task of producing the software before the partner working on the product design completes the task. The designer team should deliver product specification that contains requirements and features lists that will be used as a reference for the development team in developing the product software. The same thing happened in BetaSoft, where the development team in Kuala Lumpur require product backlog which is processed by the Product Manager and the Feature Owner based on the results from the Technical Board meeting.

### 5.1.3 Knowledge Dependency.

Knowledge is a valuable asset for knowledge-based organizations such as product software companies. To manage knowledge and expertise dependencies, administrative coordination (such as assign tasks to the competence employees, allocate employees, or integrate outputs) is not enough. Companies should have expertise coordination (the management of knowledge and expertise) so that the team can recognize where knowledge and expertise are located when they need them (Faraj & Sproull, 2000). Our case studies show that having better or equal knowledge can help other team members perform better than when still having knowledge gap among the teams. That is why in DeltaSoft, the architectural team where the remote members attached to should support the remote team members with their expertise in the development tools and knowledge in the product itself. Another similar case also found in the Service Delivery department at GammaSoft. The lack of experience in providing a qualified work (e.g. efficient source code) can be rectified through a mentorship from employees from the host office.

## 5.2 Situational Factors of Task Coordination

### 5.2.1 Objectives of performing GSE.

There are different main goals that answer why companies perform GSE. Indeed, as mentioned by Ågerfalk et al. (2008), reducing development costs is the main factor that becomes the goal for companies involving development partners or building remote offices in other countries. Our respondents did not deny that the cheaper salaries for engineers in Eastern Europe, South Asia, and Southeast Asia attract companies to save money on their development costs.

Countries in those three regions also provide more young, tech-enthusiast employees that have quality skills equivalent to engineers from developed countries. As also noted by our respondent from ZetaSoft (iv-z-1), which does not perform GSE, building remote offices or having partners in other countries results in increased travel costs for host companies because

managers or engineers from host companies should travel to remote facilities frequently. This fact is also recognized and understood by DeltaSoft. The main objective DeltaSoft has when creating a partnership with development partners is to obtain additional resources. The lower salaries for engineers in remote facilities eventually makes up for the additional travel costs incurred by DeltaSoft (iv-d-1).

At an enterprise scale, being able to focus on core business processes as a holding company is the reason for companies such as GammaSoft to stretch their product development chain by engaging business partners. Thus, GammaSoft can focus more on their main business processes, which are sales and marketing and concentrate their attention on technical matters to the remote facility and the development partner.

These objectives of performing GSE by the participating companies do not directly affect the coordination practices. However, as business objective is part of organization strategy, it can affect to the how the organization and its software processes are distributed. Eventually these distributions will determine to on how companies are managing the division of tasks and resources that will be discussed in the following subsection 5.2.2 and 5.2.3.

## **5.2.2 Organizational Distribution**

In a product software company apparently, the decisions related to the product development and technology choices such as architecture, integration, product decomposition, and development allocation are derived from the business strategy and force process and tools preferences (Bosch & Bosch-Sijtsema, 2010a, 2010b; van de Weerd et al., 2010). These, in turn, would drive the distribution of the department in the organization structure.

The distribution of the departments or teams can be seen from the organizational relationship between the host office and the remote office. We organized the types of organizational relationships in GSE as an intern-distributed organization or extern-distributed organization. Intern-distributed organization is the distribution of the departments or teams where the remote office, whether it is a division or a subsidiary, is under the management of the host office. Meanwhile extern-distributed organization is the partnership relationship such as business outsourcing where the remote office becomes the business partner of companies. In our case studies, we have two examples for task distribution in a single company: AlphaSoft, BetaSoft, and GammaSoft (iv-a-1, iv-b-1, iv-c-2). The remote office of AlphaSoft is a development branch office under the management of the Belgium office. For BetaSoft, the development office in Kuala Lumpur is a subsidiary of the Netherlands' office. And GammaSoft has a remote facility in Poland to handle the product design activities. These remote offices are running under the same company flag with the head office (Figure 5-1).

Establishing partnership with external organizations such as software development vendors is the way to extend the organization capability to distribute the engineering activities. GammaSoft has been outsourcing the development activities to a software development company in India (iv-c-2). And DeltaSoft also has been establishing a partnership with a software company in Romania (iv-d-1). The vendor's primary duty is to complete the tasks given by the host office. The business agreement between these companies and their partners can be varied. The business model of GammaSoft and its vendor basically is a task-based agreement, where host company define the tasks and the requirement of the deliverables. From there, the partner can determine the number of engineers and the best approach for them to accomplish the tasks as long as the deliverables are delivered within the required time and budget. Meanwhile in DeltaSoft, the partnership is a resource-based partnership, where the

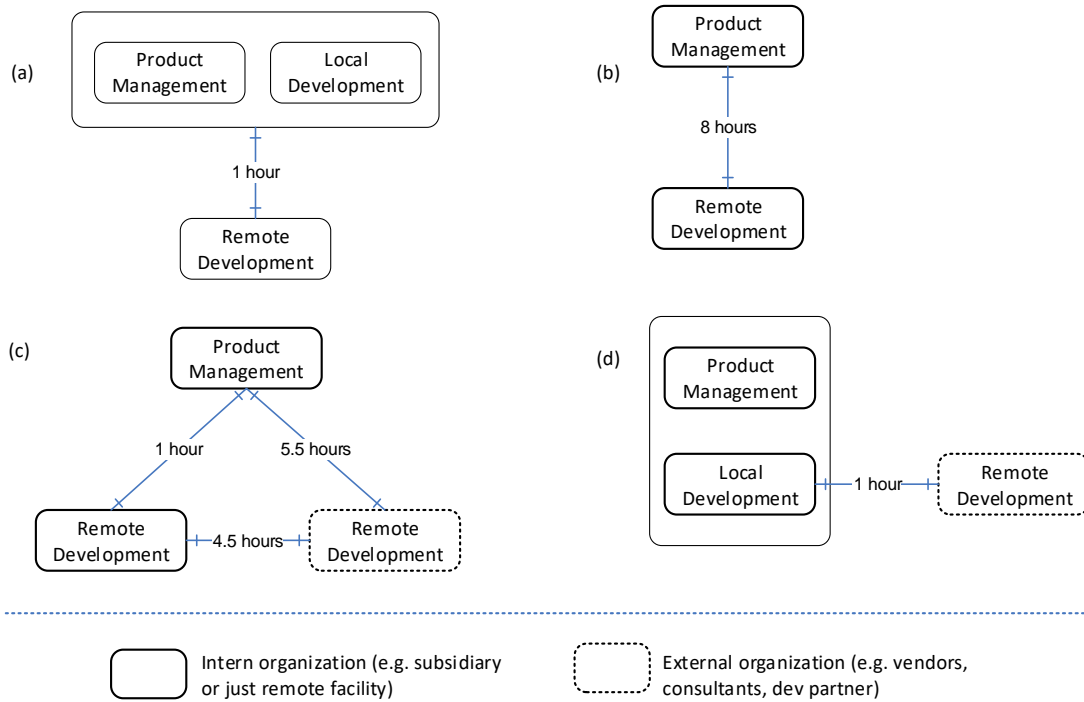


Figure 5-1 Organization distribution and their temporal dispersion distance  
 (a) AlphaSoft (b) BetaSoft (c) GammaSoft (d) DeltaSoft

partner provides the requested number of resources (employees) with specific criteria. Employees provided by the partners work directly for DeltaSoft under the supervision of team leader from DeltaSoft.

Differences in relations between the two types of business relationships above do not describe differences in how coordination is performed and do not describe the social proximity between distributed team members. For example, although teams in remote offices that work with DeltaSoft work under a different manager, their relationship seemed close. Another situation is shown by GammaSoft, which, as a holding company, has subsidiaries that also has their development teams. The social relationships proximity among the distributed teams is likely to be more influenced by the size of the team in each location and the ability of the host organization to create a more open communication environment for all team members. However, this business relationship types describes on what aspects that might have to be considered by managers at the host company in monitoring the tasks and how the tasks will be handovered among the distributed teams such as the ease to access the resources (knowledge, work products, or supplies) in the remote office, the ability to intervene or manage resources in the remote office, and the ease to build a balanced communication and trust between the distributed teams.

### 5.2.3 Software Processes Distribution

Referring to software engineering processes defined by Sommerville (2010) and key process areas in product software management competence model by Bekkers et al. (2010), as discussed in Section 3.1 and Section 4.1, there are several processes that can be shared to the remote teams. In the interviews, we found that most of the companies share their development process with the remote facilities or partners. BetaSoft and GammaSoft assign the software production

to their development facilities in other countries. Small development activities for configuration and localization are done with other regional offices. Meanwhile, AlphaSoft and DeltaSoft distribute parts of the development activities to other development teams at the remote offices. All companies in our case studies are responsible for product portfolio management. A specific case for GammaSoft, the company assigns requirements management and release planning, including feature specifications to the remote facility in Poland to accomplish these tasks.

As software product management can be perceived as procedural activities where one activity is followed by other events, obviously, there is a process dependency between those separated teams. The situation described above shows the various ways of dependencies between the head office and the remote offices. Product software companies can distribute the process to their remote facilities in two ways: the distributed team is working together to accomplish the same process, or working on an entirely different process but as a part of a larger set of processes. Companies that distribute the same process to the dispersed teams should pay attention to the control mechanism of the same resources (because the same activity uses these resources) and harmonize the collaborative work by balancing the knowledge each team has. For example, the collaborated work between the architectural team and the remote partner in DeltaSoft shows that there is both resource dependency and knowledge dependency. The remote team members are the additional support for Architectural Team 3 (AT3) because they are designed to be part of the team. As the AT3 owns the knowledge of the architectural design, they are also responsible for supporting remote team members in performing their tasks by supporting them with the information that they have. Therefore, when the required knowledge is owned by either party, such as the head office, then the manager or the team leader must facilitate the access to the knowledge for the other teams. Dependencies that arise in this situation is generally the dependency of knowledge and resources used simultaneously.

On the other hand, when each team performs a unique process, the manager must ensure that the task handover runs smoothly. The output of the work should be ascertained as having the determined quantity and quality so that the next team can use this output as an input or resource in executing the following task. In this situation, dependence arises. In this case, the dependencies are more on the process dependency and resource dependency in the form of work products.

#### 5.2.4 Software Engineering Method

The software engineering process model such as Scrum, XP, or traditional ones are not reflecting the situational factors that direct companies in choosing which coordination mechanisms within the organizations. However, some practices that characterize the process models can assist companies in reducing the risks and challenges in practice GSE. For example, intensive meetings in Scrum helps direct communication more effective and pair programming in XP assist in the transfer of knowledge between distributed team members.

Most of the participating companies use Scrum or a modified Scrum process model (iv-a-1, iv-p2-1, iv-p). In AlphaSoft, there is a Scrum of Scrum where in addition to daily Scrum meetings by each of the Scrum team, a regular cross-functional Scrum meeting is also discovered. This higher level Scrum meeting is held once a week at the project level attended by representatives from each Scrum team. In BetaSoft, the software engineering process model is comparable with Scaled Agile Framework (SAFE)<sup>4</sup> that involves broader stakeholders.

---

<sup>4</sup> <http://www.scaledagileframework.com/>

Product managers, Scrum Masters, and feature owners are involved in several meetings that similar to the Scrum meetings for the development team for product feature planning (Figure 4-6). These approaches bring evidence of where Scrum meetings such as planning, daily stand-up, review, and retrospective meetings become the coordination mechanisms. As highlighted by Paasivaara and Lassenius (2006), Agile practices like face-to-face conversation and daily interactions are important in overcoming the limited amount of communication because they do not stay in one place, and they have minimum overlapping working hours. Those intensive meetings yield a chance for team members to know others' tasks, problems, and solutions, even if they must be separated by distance.

In the meanwhile, GammaSoft does not use a specific software engineering method, even though there is evidence that the scrum is used by the development partner internally. This situation could be understandable because GammaSoft wants to focus on product portfolio management and marketing activities and to divert the processes related to product design, requirements management, and release planning to business partners. Nevertheless, the difference of methods used by two or more different organizations like GammaSoft and its partners can lead to incompatibilities of processes that ultimately result in a process bottleneck that slows down the entire project.

### **5.2.5 Experiences in GSE**

We found that the variability of the companies' experiences in GSE is reflecting the difference of organization' maturity in managing the tasks among the globally distributed teams. A company like BetaSoft that has been performing GSE for almost 17 years have a sufficiently long learning process to master many things that make it easier for them to optimize the benefits of GSE and to address the problems during the joint projects with remote teams. In the early stages when GSE was started, they found difficulties in understanding their respective cultures and bridging this cultural differences. In the end, they discovered that inculturation is better done through visiting the remote office by the team from the host office.

Other companies that experience similar cultural differences choose a quick solution by providing a communication broker or a liaison officer, such as service delivery agent or a unit manager. Companies that have less experience such the GammaSoft, still struggle to establish a better communication and work synchronization with their partners.

### **5.2.6 Challenges faced by organizations.**

Geographical distance makes the distributed teams do not have the chance to build direct contact. Direct contact as a feature that can create social bond can be done through co-located meeting or face to face communication. The lack of direct contact limits the distributed team members to know their colleagues in other locations each other more in person that makes them difficult to build teamness. They also find difficulties in monitoring and controlling the work of their remote partners in other locations. In addition, a high geographical distance enables the opportunity of temporal gaps that restrict them to have enough collaboration time.

Companies who have nearshore remote offices still able to have direct interaction for communication to and supervision of remote offices. They can optimize synchronous communication tools, such as communicating by telephone or WebEx and video conference. But the differences caused by geographical and temporal distances bring the situation where product software companies find difficulties in having more chance to have more direct communication. Thus, indirect communication through communication broker or managers can be used where these brokers can help in analysing, compiling questions or requirements and



communicating to the appropriate teams. The use of asynchronous communication such as by using email also can be used to complement the direct communication.

The difference of tools or methods that are used by each location makes the process handover cannot be done smoothly because the tools or methods are not compatible each other. The organization maturity in handling the tasks, for example, the lack of experiences of the global partner in adopting new technology also can harm the project performance. The organization may work harder to manage these issues among the distributed teams that often make the communication becomes exhaustive. In addition, companies that share part of the processes with offshoring offices, such as those in Kuala Lumpur or India, should manage how they could improve limited communication to be better. Because the head offices cannot get involved in every task in detail, they expect that the remote teams can provide the output according to the head offices' expectations. Therefore, companies should determine the standardization of process, output, and knowledge to ensure that the teams can have enough knowledge to perform the tasks, perform the tasks effectively, provide the work product as the required quantity and quality.,

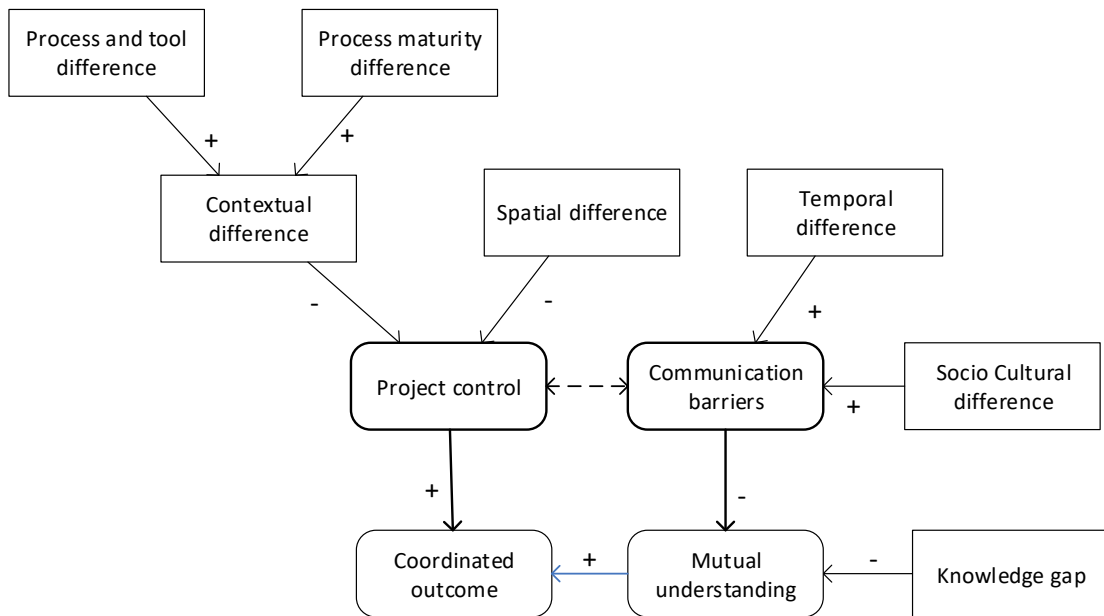


Figure 5-2 GSE challenges causal model

As the conclusion, it can be perceived that the faced challenges affect product software companies in different ways which eventually influence the way of these companies in determining the appropriate coordination mechanism (Figure 5-2). These challenges force whence companies are attempting to achieve the coordinated outcomes by controlling the tasks and reducing the communication barriers that can hurt the team members to build mutual understanding. Therefore, the GSE challenges can be understood as the situational background for the coordination mechanism for product software companies.

### 5.3 Task Coordination Approaches: Communication, Control, and Knowledge Sharing

From the literature study, we identified three main coordination means that are also affirmed by the participating companies during the interviews as their mechanisms to manage the task

dependency among the distributed teams, which are communication, control, and knowledge sharing.

### 5.3.1 Communication

Communication encompasses the process of transfer and exchange of information that takes place between communication partners (Altmann, 1999, p.2). Communication is defined as an organic coordination mechanism to manage dependencies through providing feedback and mutual adjustment (Van De Ven et al., 1976). According to the study performed by Lamersdorf, Munch, and Rombach (2009), there are two types of communication mechanisms: Direct communication and indirect communication mechanisms. What is meant by direct communication, in this case, is a communication made directly between two parties without any intermediaries. Lamersdorf, Münch, and Rombach (2009) found that direct communication was not possible to be done in distributed software engineering situation. Even so, indirect communication also was not easy because of the difficulties in finding the responsible person on the other side.

Based on the practices performed by the participating companies, they strive to optimize direct communication as much as possible such as through phone and video conferences. Regular and scheduled meetings in Scrum are events where team members from remote offices also attend through video conferencing or WebEx. Additionally, site visits to have face-to-face meetings, some of which are regularly programmed, and some of which are not. Ad-hoc direct communication between team members commonly occurs in a smaller distributed environment, such as in AlphaSoft and DeltaSoft. The communication, mostly done through Skype, happens when a team member needs to arrange a meeting, ask questions or for feedback, or ask for some help from other team members. Meanwhile, on a larger scale, communication is mostly done through intermediaries, such as the Product Owner. For companies that have five to seven-hour differences with their remote facilities such in BetaSoft and GammaSoft, the chance to have synchronous communication is limited. Asynchronous communications, such as the use of email, are done for less significant coordination purposes as a complement to the lack of synchronous communication. However, the synchronous communication remains more preferred. Therefore, they try to optimize their small overlapping working hours as much as they can.

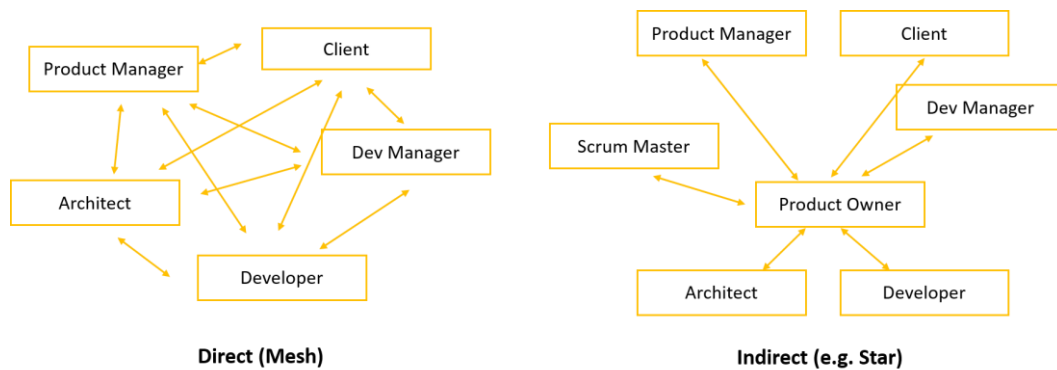


Figure 5-3 Organization design in communication

Indirect communication can be in a is also used when there is a mediation role that integrates the information among distributed stakeholders (Chiu, 2002). In Figure 5-3, a star communication model describes that the Product Owner in Scrum process model becomes the central role. The product owner translates business requirements from clients and product

manager into product's features in a technical language that can be understood by the engineers and articulates the engineers' consideration that can change the requirements to the business users. Moreover, indirect communication in a large company can be in a hierarchical form following the information flow mechanism within the organization top to bottom and vice versa. It is necessary to break the network into smaller groups to facilitating communication (Chiu, 2002).

We found that communication among team members in global software engineering projects are done concurrently and spontaneously. The situational factor that affected the way of team members communicating each other is the level of temporal dependency. When the chance for dispersed team members to work together is high, then it is better to have more direct and synchronous communication. On the other side, direct and synchronous mechanisms still must be performed and fulfilled using indirect communication mechanisms.

To have better communication, we propose a set of activities a reference method that should be carried out as below:

1. Identify factors causes lack of communication, such as
  - a. horizontal causalities such as cultural gap and lack of trust
  - b. vertical causalities such as the problem of knowing to whom team member should consult
  - c. the level of temporal dependency
2. Improve communication mechanisms
  - a. Perform cultural internalization
  - b. Assign medio-broker
  - c. Share organization structure (who knows who & what)
  - d. Improve communication protocol (especially for indirect & asynchronous)

### **5.3.2 Controlling Dependency and Synchronizing Tasks**

Dependencies arise when multiple individuals or teams, their tasks, resources are interacting and creating a chain of processes that need to be synchronized as a joint task. Each individual or team might be able to manage their tasks and their resources. However, they could not be able to perform well, should take another longer way around, or even could not continue their tasks if the tasks are related to respective dependencies from other tasks. By recalling coordination definition as an effort to manage dependencies, coordinated outcome can be seen as a state where all the respective dependencies are well-managed.

A team as an individual entity might be performing its tasks individually. However, when it takes into a project where each process is chained one to another such as in software engineering, the team should consider the needs of other teams. Other teams may require not only the deliverables of the previous tasks should be done within the required time, but also in the right quantity and quality. That is why in a software engineering projects, leaders are needed to do some management functions, depend on what kind of functions required in the project. Leaders in software engineering projects at product software companies could be varied, there are project manager, product owner, service broker, or team leader. However, they have a common function: to manage the dependencies by synchronizing the activities. Synchronization activities will bring all the team members together at the same time and place for some pre-arranged purpose.

In product software companies, dependencies occur not only in the development area such as the need of the data structure for UI/UX developers from the business logic development

team. Our case studies show that at the enterprise level, there are dependencies between product portfolio management team to the technical team. Each team also has some dependencies between team members in it, for example at no dependency technical team process between product designers and product developers. Companies need to adopt software engineering method that supports multilevel coordination or coordination in a wider scale such as Scrum of Scrums and Scaled Agile Framework (SAFe) to integrate task dependencies that occur in multi-scale level (Paasivaara & Lassenius, 2016).

For companies that have more experience in managing GSE projects, such as BetaSoft, as time goes by, they have more stable coordination process at the operational level. The coordination process can provide work direction and reporting procedures to guide the teams. This approach emphasizes the teams as self-organizing teams who able to decide what the best ways to perform their tasks. As the example, AlphaSoft and BetaSoft use mutual adjustment mechanisms to manage interdependencies. They have scrum masters to facilitate the coordination among team members by organizing regular meetings and updating information to the knowledge base. Managers, supervisors or facilitators help the team in doing their horizontal coordination to achieve their best performance. However, for companies that are just starting GSE such as GammaSoft, they should keep doing direct supervision to their remote teams or partners until they are able to deliver the results with appropriate quality.

Thus, to have a better-synchronized outcome, there are three coordination mechanisms derived from Mintzberg (1979) related to the managing interdependencies and aligning the tasks and work products. Therefore, we propose a set of guidelines in task coordination as below:

1. Identify team's ability in managing interdependencies
  - a. Facilitate mutual adjustment
  - b. Perform direct supervision
2. Provide standardization
  - a. Work processes standardization, for example standard programming style
  - b. Outputs standardization, such as documentation and work product's quality and quantity

### **5.3.3 Distributing Knowledge**

Almost all the practices at our participant companies present similarities of knowledge-sharing mechanisms. The knowledge is shared using a document repository to store explicit knowledge and direct individual interactions for tacit knowledge. A collaboration platform (Microsoft TFS) is commonly used as well as SharePoint, which is the document repository. In this particular case, because of the closeness of the social interaction among team members at DeltaSoft, they prefer to have informal and direct communication or transaction information by using Slack.

Based on the aspects provided by Kotlarsky et al. (2008) on how organizations optimize knowledge as an asset to support coordination, there are some differences in the way companies manage knowledge. DeltaSoft uses tacit knowledge through social interaction. Knowledge is perceived as social capital that can be accessed by anyone with a direct interaction. This situation makes explicit knowledge less common in their coordination activities. Meanwhile, companies like AlphaSoft, BetaSoft, and GammaSoft have distributed teams with broader scales (e.g., locations and number of employees). They optimize organizational functions by providing several roles and job functions to manage knowledge and make knowledge stored explicitly in online repositories that can be accessed by team members. Those companies are

similar in terms of knowledge management in how they use of tools to facilitate communication and collaboration. Comparison of knowledge management mechanisms is delivered more detail in (Table 5-1).

*Table 5-1 Knowledge coordination mechanisms by Kotlarsky et al. (2008)*

<b>Mechanisms</b>	<b>AlphaSoft</b>	<b>BetaSoft</b>	<b>GammaSoft</b>	<b>DeltaSoft</b>
Facilitating knowledge flows	By design, the Unit Manager connects both remote offices.	The remote office is designed as an independent organization under the control of the head office. The feature owners and product managers have the responsibility of managing knowledge flows.	Not specifically mentioned, but there is a liaison officer from the distance partner.	One of the teams is assigned to collaborate with the nearshore partner directly.
Making knowledge explicit	The scrum meetings become spaces for direct coordination activities. Product owners are responsible for making the knowledge explicit.	Engineering processes (roadmap definition, product design, and product development) are distributed to partners or remote office. Product owners, feature owners and product managers are responsible for sharing the information related to the work as explicit knowledge in the collaboration workspace.		They use task-based work management. They realize the importance of explicit knowledge, but they feel more comfortable with intensive direct interaction.
Amplifying knowledge	All the companies commonly use collaboration platforms that support both synchronous and asynchronous coordination.			
Building social capital	Direct communication between team members is based on professional relationships. Unit managers become the spokespeople of communication and the gatekeepers of social interaction from remote teams to the head office.	Informal interaction during the site visit helps employees from the head office to feel the hospitality of people from remote offices and encourage both sides to adapt to the culture.	The interactions are based on professional relationships.	Intensive social interaction over the professional relationships melts the ice between those two sides.

Software engineering is a knowledge-intensive activity know (Bjørnson & Dingsøyr, 2008). As a part of knowledge-based process chain, knowledge sharing is believed to contribute to the collaboration in software engineering projects, including global software engineering (Kotlarsky & Oshri, 2005). Knowledge sharing also enables team members to help others in developing knowledge about the tasks and the team which helps them coordinate implicitly. A shared cognition enables team members to explain and anticipate task states and member actions (Espinosa, Lerch, & Kraut, 2002). Without an effective knowledge sharing, the project can suffer due to the failure of coordination problems that encourage collaboration (Herbsleb &

Moitra, 2001). Adopting coordination expertise delivered by Faraj and Sproull (2000), coordination of knowledge is done in stages as follows:

1. Identify the existence of knowledge, such as what kind of knowledge, where the knowledge is located, and who has the knowledge.
2. Determine the needs of knowledge
3. Make the knowledge available and accessible

**Identification of the existence of knowledge.** Product software companies as organizations that carry out the software engineering to produce software as a product should be able to identify the existence of knowledge as their assets. Knowledge can be in the form of tacit knowledge. Tacit knowledge cannot be expressed explicitly but lead or enable people to behave and carry out their duties. Knowledge can also be expressed in an explicit form as textual documents or other symbolic forms such as diagrams and drawings (Nonaka & Takeuchi, 1995). Our findings on practices such as assigning nearshore outsourcing by GammaSoft and building an offshore development facility as DeltaSoft's show that tasks collaboration with distributed teams indicates segregation of expertise. The expertise separation means that each location has its unique capabilities. Our findings also indicate a different situation which on a broad scale, the explicit knowledge, as well as they who are responsible for knowledge storing, can be easily recognized. While on a smaller scale distributed team such as in DeltaSoft, knowledge mostly presents as tacit knowledge.

**Identifying the need for knowledge.** Companies need to recognize the cognitive level of team members to know what kind of knowledge needed by the team members. A study performed by Kristjánsson et al. (2014) reveals that process novels (such as new tools, technologies, or methods) bring a knowledge gap that needs appropriate adjustment in knowledge level. Knowledge gap also can occur when two teams from different locations with different tools and approaches should collaborate and decide to use only single approach or tool in the project (Kotlarsky et al., 2008). The knowledge gap can occur across all phases of the development process within a company. Therefore, the company can create a mapping between the available knowledge and the knowledge required to provide a knowledge gap analysis. The analysis can be used to determine the proper knowledge sharing mechanisms that suit for the organization.

**Making the knowledge available and accessible.** When companies already know where the knowledge is located and what kind of knowledge is needed, then the following step is creating access to the necessary knowledge. Our study identified several mechanisms in disseminating or distributing knowledge based on the knowledge transformation categorization by Nonaka and Takeuchi (1995), namely: Socialization, externalization, combination, and internationalization (Figure 5-4).

A knowledge broker is needed to let the knowledge flows to the knowledge owner when the required knowledge or the knowledge owner cannot be accessed directly. A knowledge broker may connect two distributed team with different expertise. For example, when a developer at the remote office needs to clarify an unclear requirement, the developer needs to contact the Product Manager from the head office. The Product Manager can provide boundary spanning function that connects the developer to the business user who has the requirement. In other situation, a broker also can be someone who maintains the knowledge boundaries that makes each team focuses on their specific expertise. In our example, a knowledge broker will not allow a team member has a

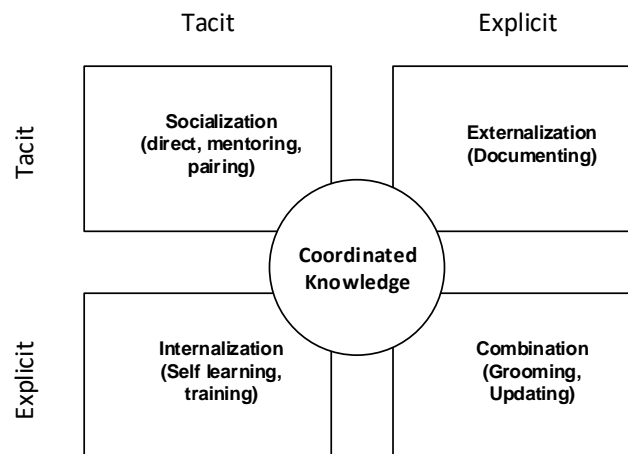


Figure 5-4 Knowledge coordination mechanisms

meeting with the respective business user to clarify the requirement and provide the answer to the developer.

By adopting expertise coordination approach by Faraj and Sproull (2000) and considering the practices performed by the companies to coordinate knowledge in global distributed environment projects, we propose a set of activities as a reference method that should be made as below:

1. Identify source of knowledge: where the knowledge is stored, is it a tacit or explicit knowledge, how others need the knowledge
2. Identify types of knowledge needed by the team (such as technical or functional information)
3. Identify gap of knowledge
4. Identify situational factors (such as organizational distribution, size of virtual team)
5. Open access to knowledge by choosing proper mechanisms

#### 5.4 Involved Tools in Task Coordination

We identified several tools that are used in global software engineering setting. The purposes of the tools can vary, such as providing collaboration space, enabling direct communication, amplifying the distribution of knowledge, and enhancing project control (Table 5-2).

Tools for collaboration tool are commonly found, such as Microsoft TFS. Collaboration tool is used to support two or more individuals or teams to accomplish a common task or to achieve a common objective (Signell et al., 2008). This type of tool can support not only collaboration, but also project management, integration with the development environment and document repository, and organizational information (Lanubile, Ebert, Prikladnicki, & Vizcaino, 2010). Other tools such as Skype and WebEx that support organic coordination mechanisms to maintain dependencies and share feedback through direct communication such as phone calls, video calling, teleconference, and videoconference are the basic tools for communication in software engineering in a global environment. They also can be used to support knowledge sharing to encourage shared cognition. Non IT-based instruments such as Scrum board and burn down chart that is used also can be assumed as tools because these instruments also support managers and team members in project and tasks monitoring and identifying dependencies.

Table 5-2 Tools adopted to support coordination in GSE

Tools	Functions	Examples
Collaboration tools	Provide collaboration space and versioning control for the product development	Standard IDE, Microsoft TFS, Slack, OneVision
Communication tools	Enabling dispersed sites to have a direct communication	Skype, WebEx, video conference (Polycom, Cisco)
Knowledge sharing tools	Making knowledge explicit or accessible Facilitating knowledge flows	SharePoint, Slack
Project management tools	Supporting managers in monitoring progress to establish coordination mechanisms for better project performance Showing team members profile to show other member's profile and his/her tasks	Microsoft TFS, OneVision, organizational chart, Scrum Board, Burn Down Chart

There are situational factors that can make the chosen tools are varied among these companies. When the distributed team size is small, tacit knowledge is more used that explicit one, or in a situation where an extensive direct conversation is mostly optimized, a tool with chat and file transfer features such as Slack and Skype are mostly used. However, for a situation where the knowledge and information are centralized, or coordination dispersed teams should be mediated, online repository and tools to conduct virtual meetings (e.g. WebEx, video conference) are frequently applied.

## 5.5 Organizational Support for Task Coordination

This support defines the instruments provided by the organization to facilitate task coordination. The organization support comprises the following aspects:

- Organization strategy and governance. The organization defines the initiatives that utilize the organization resources to perform GSE as well as the mechanism that oversees the strategy implementation in achieving its goals.
- Organization structure. The organization optimize the line arrangement of authority, communication, roles, and functions of the stakeholders.
- Stakeholders. Stakeholders in this context are they who have concern in task coordination in the organization. They can be someone who facilitate, participate, or have the authority in coordinating tasks and manage interdependencies in a distributed product development environment.

From the literature study and our case studies, we identify several roles and functions needed to maintain task coordination in global software engineering: Supervisor, facilitator and knowledge broker or boundary spanner. The situation that can affect the kind of coordination mechanisms needed by the companies is software engineering methods, the size of the virtual team, and organizational distribution.

The roles or job positions provided to perform coordination functions are different from one company to another, which can be seen from their coordination mechanisms. AlphaSoft uses Scrum meetings as the main communication means in coordinating tasks. Therefore, a scrum master becomes the important role here. Because the Scrum Master stays in the head office,



AlphaSoft also has a unit manager to become their coordination bridge between teams at the remote office and the head office. Meanwhile, in BetaSoft and GammaSoft, each location has a particular task specialization related to software process engineering. For that reason, they need several roles, such as product managers and feature owners (product owners) who can bridge these processes. In this case, the product manager and the feature owners are assisted by a development manager assigned specifically to manage internal coordination in the development team. Both of them maintain the tasks in integrating engineering processes and managing dependencies between the processes performed by their partners or colleagues at remote locations. In contrast to DeltaSoft, which virtually merges remote employees as team members in the head office, the team leader becomes an important function to facilitate communication and conduct supervision of the implementation of tasks.

The roles and job positions identified from the literature and interviews that perform coordination functions are summarized as presented in Table 5-3.

*Table 5-3 Roles and their job functions related to task coordination in GSE*

<b>Roles</b>	<b>Job Functions</b>	<b>Example job position</b>	<b>Situational Factor</b>
Supervisor	Managing dependencies, organizing resources	Product Manager, Development Manager	Organizational silos, large-scale virtual team size, high dependencies between dispersed sites.
Facilitator	Facilitate team members to arrange coordination by themselves	Scrum Master, Product Owner, Product Manager, Team Leader	Companies have already established their engineering processes, small-scale virtual team size.
Knowledge broker/boundary spanner	Connecting team members to the source of knowledge Mediating distributed location to the others	Unit Manager, On-site Coordinator	Knowledge and expertise are distributed in different locations



# **PART THREE SOLUTION DESIGN AND VALIDATION**

**Chapter 6. Method Design: Towards Methodological Support for Task  
Coordination in GSE**

**Chapter 7. Method Validation: Evaluation and Evolution**



# Chapter 6      METHOD DESIGN:

## TOWARDS METHODOLOGICAL SUPPORT

### FOR TASK COORDINATION

In this chapter, we will summary our findings from the literature review and case studies to find the concepts that reflect coordination mechanisms and formulate our instrumentation of methodological support for task coordination.

#### 6.1 Method Construction Preparation

##### 6.1.1 Situational Factors

After identifying the feature groups, a set of situational factors from the literature study and our case studies are summarized and organized to recognize the factors' variabilities as can be seen in Table 6-1. These situational factors can be organized into inter-organizational (strategic) factors and practical factors caused by the GSE challenges:

1. Internal factors

The organization itself has numerous situational aspects affecting to how the organization prepares and manage task interdependencies. The objectives of performing GSE brings consequences to the chosen mechanisms, such as development cost management by distributing processes to more competence vendors encourages companies to increase their travel budgets for the site visits, but compensated by the lower salary cost for the remote engineers. Other companies might focus only on building a social bond among the developers because they choose only to expand their number of engineers for specific tasks.

- Organization distribution. How large are dispersed teams, legal relationships between scattered organizations, and how organizations divide the engineering works.
- Process distribution. The relationship among the tasks, the proportion of overlapped tasks, and the process chain between the distributed teams.
- Dependency. How the artifacts are shared or transferred among the distributed teams.

2. GSE Challenges.

We consider that GSE challenges in Table 3-1 provide variability in determining appropriate coordination practices. Problems emerge from the incompatibility of processes, tools, and issues related to collaboration bottlenecks because the teams do not stay in one place are expected to impact on the way job settings and dependencies. Temporal challenge and socio-cultural problems frequently become the communication barriers that inhibit the achievement of mutual understanding. These issues ultimately lead to the threaten of achieving a coordinated outcome.

Table 6-1 Organization of the Situational Factors

Factors	Variability
<b>Organization profile</b>	
Objectives	{cost saving, expertise fulfillment, resource fulfillment}
Organizational Distribution	{holding, partnership}
<b>Software Strategy</b>	
SE Method	{Agile (Scrum, Scrum of Scrum), traditional}
Distribution in SW Processes	{expertise distribution, process, distribution, resource distribution}
<b>Challenges</b>	
Geographical distance	{low, high}
Temporal distance	{low, high}
Cultural gap	{low, high}
Knowledge gap	{low, high}

Since task coordination is a creative approach, we identify companies' preferences in selecting coordination mechanisms into the following task coordination profiles as elaborated in

Table 6-2:

1. Methodical – The approach where tasks coordination method is used to support organization to manage tasks segregation. Each division (a team, a vendor, or remote facility) has its responsibility of in a different task which is not handled by the other division.
2. Practical – The organization prefers to be more pragmatic in coordinating tasks. It describes how coordination among team member is horizontally performed.
3. Combination – Organization manages coordination mechanisms methodically by supervising the task dependencies management as well as consider to maintain the peer-to-peer coordination.

Some companies are still learning to manage the best approaches in coordinating tasks, some of them already find the best approach and even can optimize their approach to satisfy the dynamic situation of global software engineering. By referring to CMMI level definitions (Software Engineering Institute, 2010) and the coordination pyramid by Sarma, Van der Hoek, and Redmiles (2010), we defined a set of experience levels for the organization (Table 6-3). The experience level definitions are used as a quick reference for companies to perform continuous improvement in managing task coordination.

Table 6-2 Coordination Mechanisms Profiles

		Coordination Profiles	
		Practical	Methodological
<b>Organization characteristics</b>		Small team-sized and working on the same software processes	Large, each team or individual works on different software processes
<b>Mechanisms</b>	<b>Maintaining communication</b>	Direct	Indirect, aligned with the SE processes
	<b>Controlling project</b>	Direct supervision, mutual adjustment, standardization	Direct supervision, mutual adjustment, standardization
	<b>Sharing knowledge and expertise</b>	Socialization, Internalization	Internalization, Externalization, Combination
<b>Tools</b>		Tools that support direct social communication such as Slack	Tools that integrates project management and support collaboration in software engineering processes such as TFS
<b>Organization support</b>		Almost none since the collaboration is done directly and in a small-size team	Organization structure that defines clear role and functions distinction. Supported by communication and knowledge broker

Table 6-3 Task Coordination Experience Level

Experience Level	Description
Initial	The company has not been specifically defined functions in business processes and organizational structure regarding GSE and tend to be reactive in dealing with problems in the coordination of tasks.
Managed	The organization has managed task coordination in GSE projects by using current organizational processes and structure.
Defined	The organization has been specifically defined functions in business processes and organizational structure regarding task coordination in GSE projects.
Quantitatively Managed	The organization has defined the process control and able to contextualize the information. The distributed team also have considered the workspace awareness.
Optimizing	The organization also continuously improve the approach in managing task coordination in GSE projects

### 6.1.2 Identify Activity Groups

By referring the use of feature group terminology in the study conducted by Luinenburg et al. (2008) We define an activity group as a set of relevant activities that possess a similar characteristic. We elicited the activity groups from the existing approaches from the preliminary study phase (Appendix D). We will use the activity groups to serve as the

association criteria for in the designed method construction phase. The elicited activity groups are presented in the following Table 6-4.

*Table 6-4 Activity Group*

<b>Activity Group</b>	<b>Description</b>	<b>Main Sources</b>
Identify organizational planning	Companies reflect the current enterprise strategy in GSE and product management strategy.	[L5], [L6], [CB]
Diagnose situational factors and challenges	Companies reflect their practices in performing global software engineering projects and identify situational factors and challenges that they are facing.	[L5], [CB]
Identify task coordination support	Companies identify organizational support such as structure, roles, and job functions in the product software engineering roadmap that relates to global software engineering projects. Companies analyze the infrastructure or tools that they have.	[L5], [L6],[CB]
Determine appropriate task coordination mechanism	Based on the situational factors and challenges, companies select which coordination mechanisms that match with their profiles.	All sources
Perform process improvement	Companies evaluate the improvement of the choose of the coordination mechanisms and improve the practices by again reflecting the current situational factors and new challenges that they have.	[L6,CB]

## 6.2 Constructing Task Coordination Methodological Support

### 6.2.1 Method Association

In performing the method association phase, we use an association table to create a map from the method fragments to the activity groups (Luinenburg et al., 2008; van de Weerd, Brinkkemper, Souer, et al., 2006). We defined new terms for the key activity names standardization. Several activities from the preliminary study phase may have different names but have a similar meaning. For example, “Assign a liaison officer” [L7] and “Assign a service coordinator” [CC] have two different concepts namely “liaison officer” and “service coordinator”, but these concepts can be understood as a single concept: “On-site Coordinator”. Another activity might consist of two activities, such as “Collaboratively develop, communicate, and distribute work plan” should be split into “Develop work plan” and “Distribute work plan”.

From the association table, a framework and a method for task coordination in GSE projects at product software companies are built. The framework depicts how the concepts are incorporated. Meanwhile, the method that describes how companies can choose the appropriate coordination mechanisms that suit with their organization. The following table (Table 6-5) shows how activities acquired from our preliminary study. The complete association matrix is provided in Appendix E. Additional activities also added based on our subjectivity to maintain the logical order and flow of the activities within the method.



Table 6-5 Method Association (Example)

Activity	[L1]	[L2]	[L3]	[L4]	[L5]	[L6]	[L7]	[CA]	[CB]	[CC]	[CD]
Identify challenges	x										
Identify types of dependencies	x										
Assign onsite coordinator							x	x		x	
Determine organization structure		x							x		
Identify software engineering processes		x						x	x		

### 6.2.2 The GSE Task Coordination Framework

A preliminary version of the framework is presented in Figure 6-1. This preliminary framework is built by considering the concepts that we have elaborated in Chapter 5. Briefly, the framework shows two concepts that are affecting product software companies in choosing the appropriate mechanisms in coordinating tasks in global software engineering (Organization situational factors and Challenges) and two concepts that are supporting the operation of the mechanism (Organizational support and Tools) to achieve coordinated output.

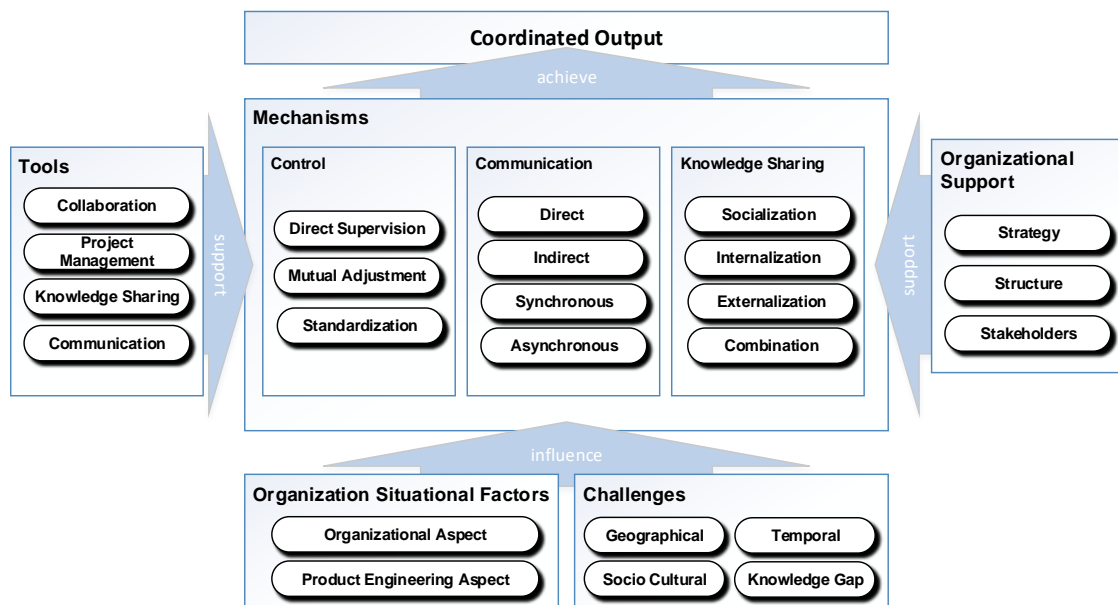


Figure 6-1 Framework for coordination mechanisms in GSE

### 6.2.3 The GSE Task Coordination Method

A primary version of Task Coordination Method referring our framework in Figure 6-2 is presented in this section in step by step processes based on the activity groups identified in Section 6.2.2.

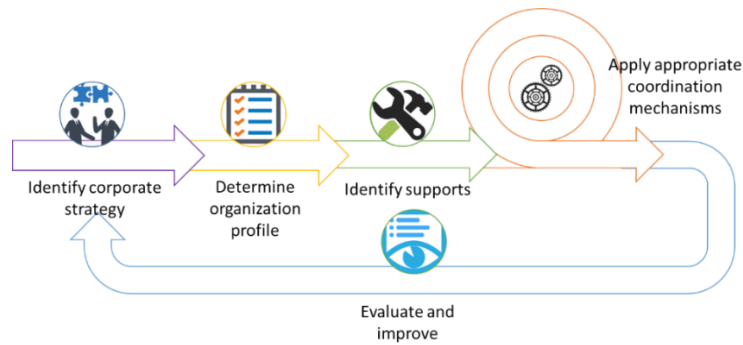


Figure 6-2 Picture Diagram of Task Coordination Method

The task coordination method’s PDD is presented as depicted in Figure 6-3 and the more detail PDDs will be introduced in the following discussion.

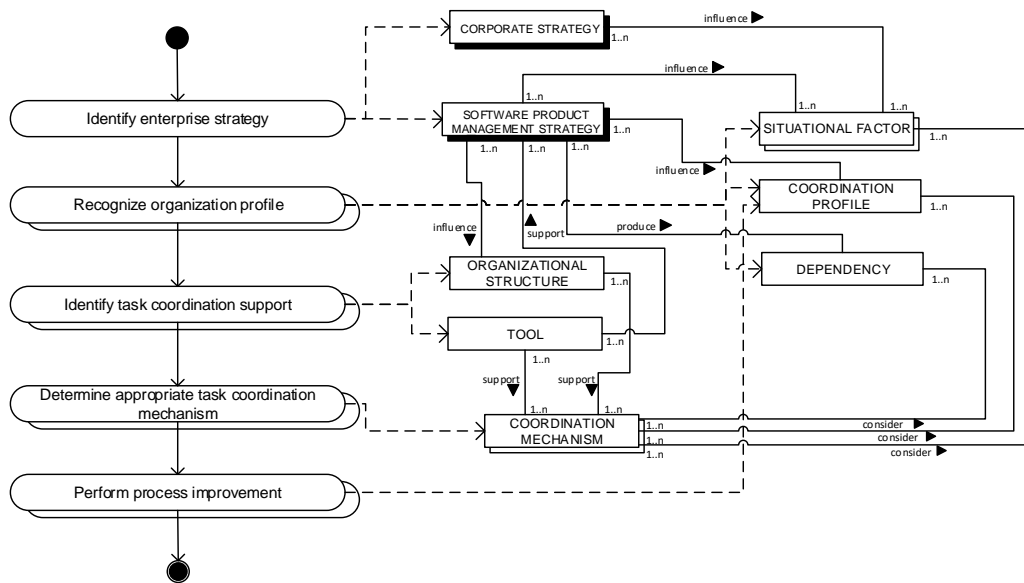


Figure 6-3 High-level PDD of GSE Task Coordination Method

The following discussion elaborates each of the main steps of the GSE task coordination reference method.

**Identify enterprise strategy.**

This first activity group simply reminds organizations to reflect what are their corporate strategy and their product software management strategy. Those strategies might not directly affect to the coordination mechanisms, but they will guide the employees throughout all processes.

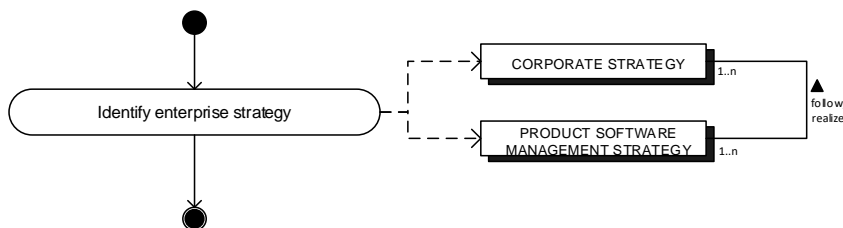


Figure 6-4 Activity Group 1: Identify enterprise strategy

**Recognize organization profile.**

The activity group is concerned with identifying any profiles related to the situational factors in the organization, such as organization strategy in GSE, organization strategy in its product engineering processes, and challenges faced by the organization. The organization then identify its current coordination profile to measure its current level and to improve its coordination approaches in the future. The organization also should recognize what kind of dependencies that occur in their product engineering processes.

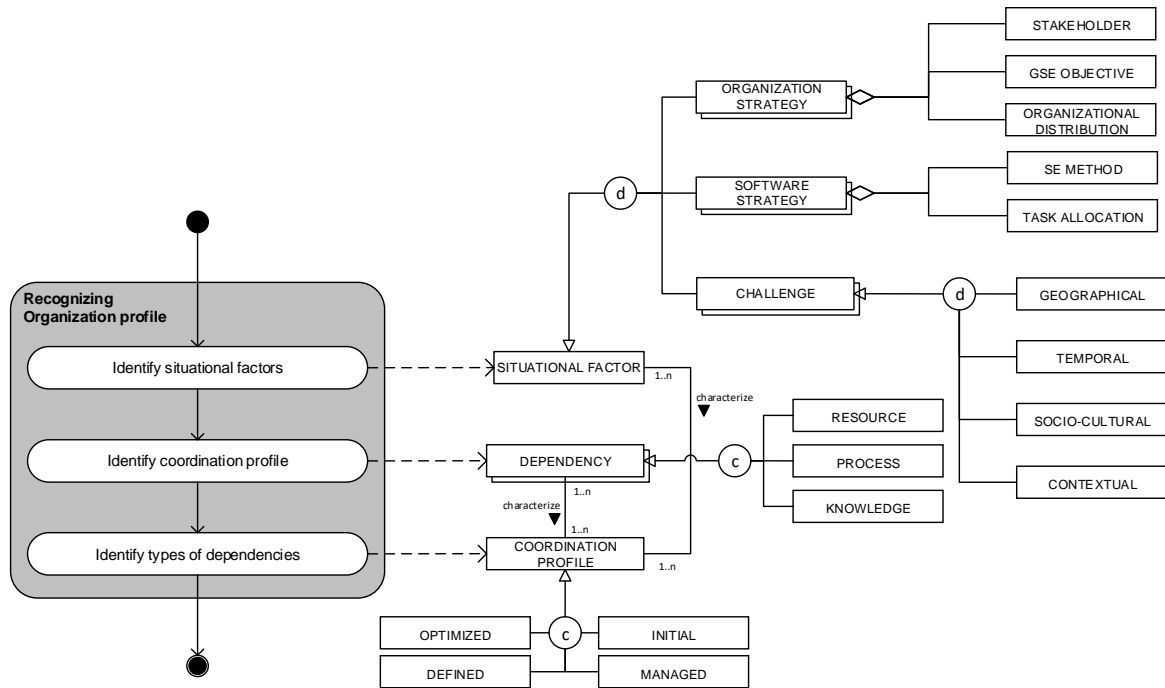


Figure 6-5 Activity Group 2: Recognizing organization profile

As product software engineering is a continuous experimentation and innovation process that produces continuous improvement in the business strategy and development operations strategy (Fitzgerald & Stol, 2017; Rodríguez et al., 2017), product software companies should also consider scaling their coordination practices. Agile software development approaches can be used to scale the coordination activities horizontally (among distributed team members) and vertically (decomposition for alignment between different functional teams). Regarding the vertical decomposition, companies should define appropriate approach such as such as Scrum of Scrum to ensure the parallel tasks are organized and to minimize the technical and social dependencies (Nord, Ozkaya, & Kruchten, 2014). By performing Scrum of Scrum, the approach to align the interdependencies is brought and replicated to a larger level to solve vertical coordination issues such as synchronization problem between different functional teams or even development team with the product design team.

**Identify task coordination support.**

The company identifies roles and job functions in company’s product software engineering processes that participate in global software engineering projects to see how organizational structure in supporting task coordination mechanisms. Company then identifies the tools and types of functionalities of the tools that are employed in global software engineering projects.

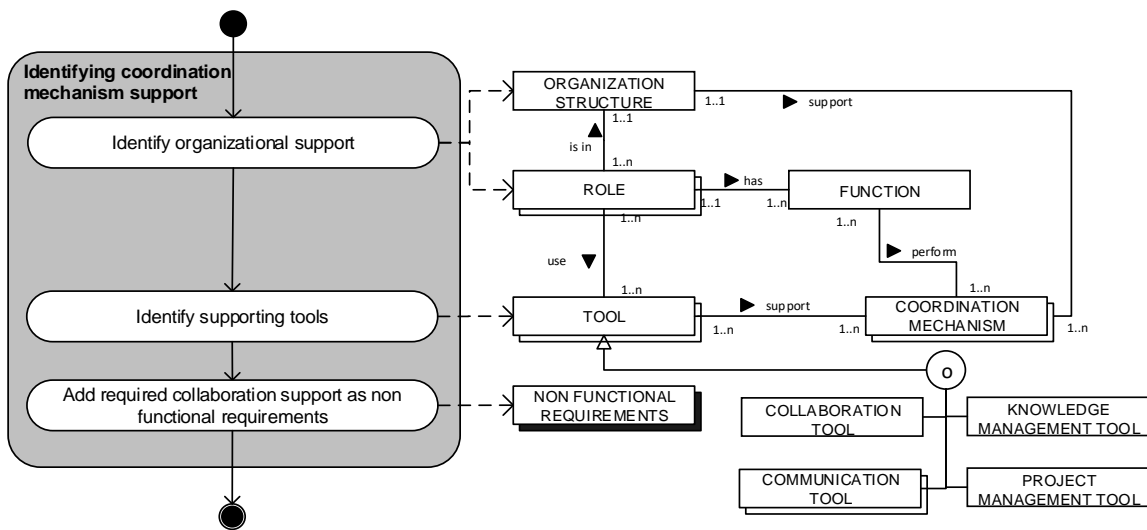


Figure 6-6 Activity group 3: Identifying task coordination support

**Determine appropriate task coordination mechanisms.**

This activity group is the main part of the task coordination method. The company can focus on one of the types of coordination mechanisms or combine several mechanisms because the situational factors and challenges are varied and interacting each other. That is why we describe the coordination mechanisms selection as parallel processes that are not chained each other the concepts (Communication, Control, and Knowledge Sharing) are using the overlapping notation for the generalization.

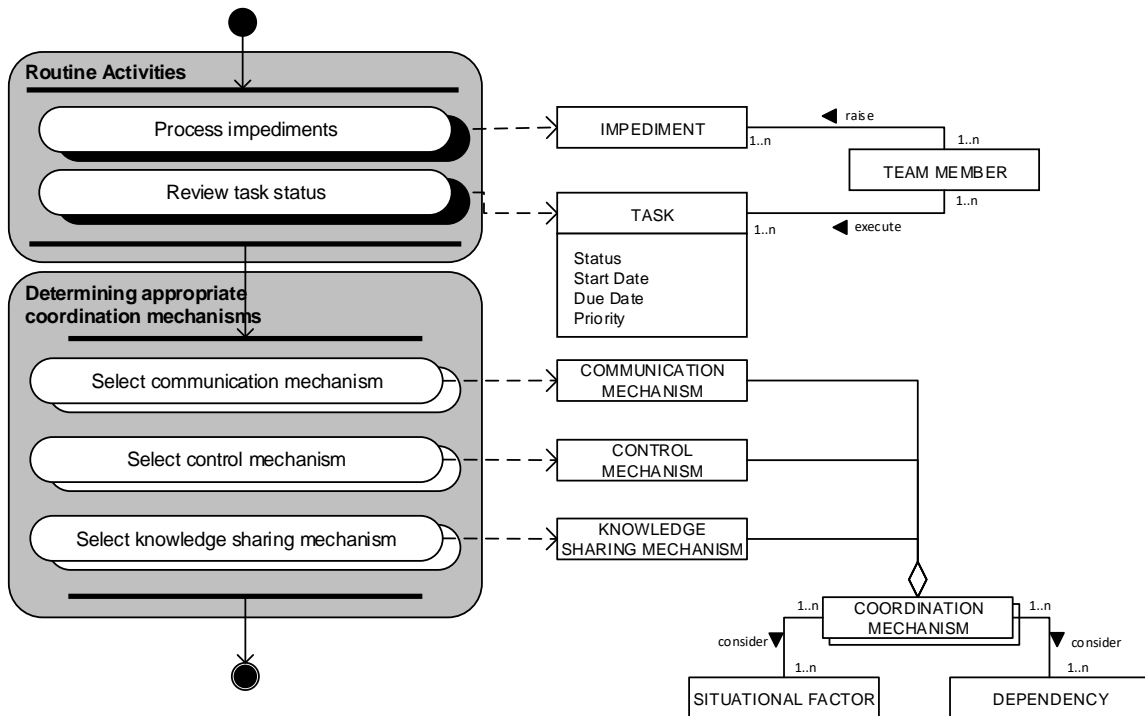


Figure 6-7 Activity group 4: Determining appropriate coordination mechanisms

**Select communication mechanism.** As described in Section 5.3.2, to define appropriate communication mechanism, companies should consider situational factors especially challenges in performing GSE.

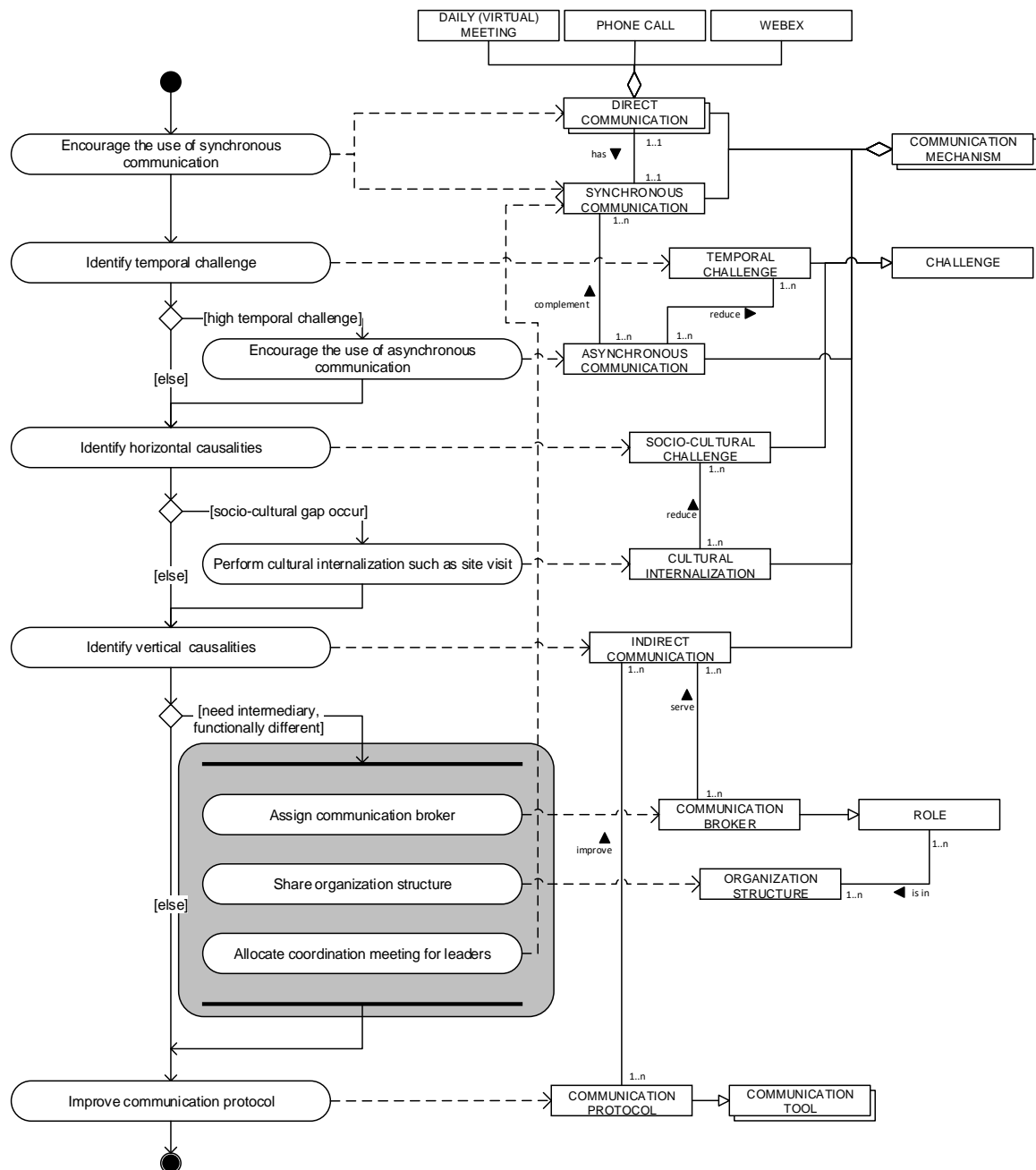


Figure 6-8 Select communication mechanism

**Select control mechanism.** By referring the guideline provided in Section **Error! Reference source not found.**, companies determine the appropriate mechanisms for controlling dependencies in software processes in GSE projects.

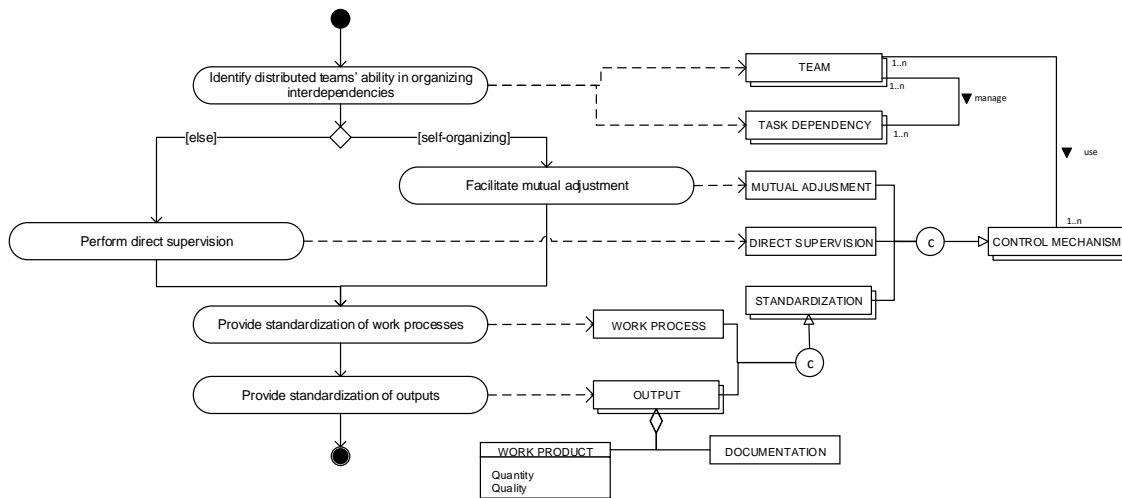


Figure 6-9 Select control mechanism

**Select knowledge sharing mechanism.** The following diagram based on the guideline provided in Section 5.3.3 depicts the approach to having an equal level of knowledge that boosts companies in performing tasks.

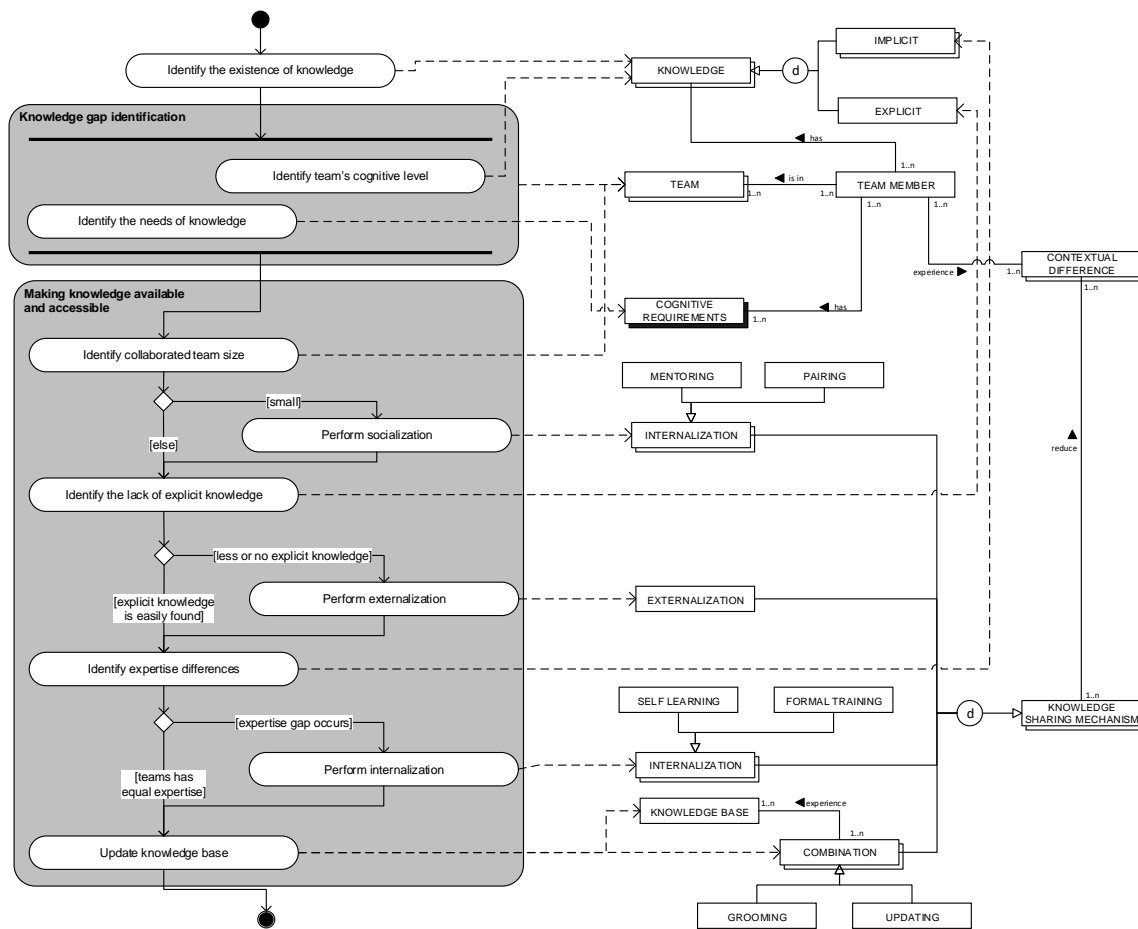


Figure 6-10 Select knowledge sharing mechanism

***Perform process improvement.***

After a company employs coordination mechanisms based on its situational factors, the company should reflect its current coordination profile. The reflection is used to see the changes before and after it improve its coordination mechanisms and continuously advance coordination practices.

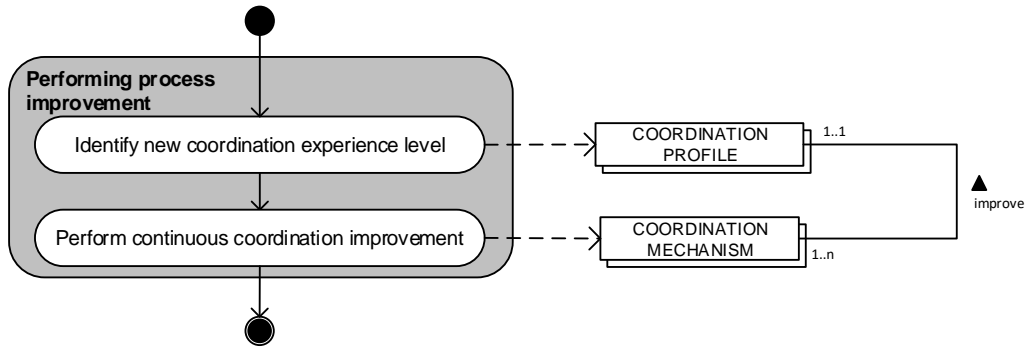


Figure 6-11 Activity Group 5: Continuous improvement

**6.3 Primary Conclusion**

As explained at the beginning, this chapter discusses the construction process of the preliminary version of our research artifacts which are the Global Task Coordination Framework and the Global Task Coordination Method. As abstracted by the framework, we identified that to achieve coordinated outcomes product software companies should consider situational backgrounds consisting of inter-organizational and faced challenges. Software product roadmap is a part of the company’s strategy as a part of the inter-organizational aspects specifically indicates that the artifacts are designed for product software companies. In addition to these two situational factors, product software companies should also prepare coordination supports in the form of tools support and organizational support. Here again, through organizational support, the role of strategy, organizational structure, and stakeholders in a product software company shows the character that the research artifacts are intended for product software company.

However, the constructed framework and the method are the early versions resulted from the synthesis of the literature study and the preliminary studies conducted through interviews in several product software companies. Therefore, these artifacts still need to be validated. In the next chapter, we will present the strategy and the performed evaluation phase of where the two artifacts are evaluated gradually and iteratively to produce the final artifacts that are expected to help product software companies in coordinating tasks in a globally distributed environment.





# Chapter 7      METHOD VALIDATION:

## EVALUATION AND EVOLUTION

The designed task coordination framework and method have been evaluated through expert interviews referring to a set of acceptance criteria. We expected feedback that criticize for the improvement as well as the perceived intention to use of our method. In the end, the feedback will be discussed and considered to improve our designed artifacts.

### 7.1 Global Task Coordination Method Evaluation Scenario

#### 7.1.1 Method Evaluation Participants

The participating experts consist of scientific experts and business practitioners. The scientific experts are a researcher in global software engineering domain from a technical university and a Method Engineering course' student assistant from Utrecht University. Meanwhile, the practitioners are those who are participated in our preliminary study (Table 7-1).

*Table 7-1 Participating experts*

#Evaluation	Expert's profiles	Background experiences
1.	Researcher	Seven years in GSE projects and a professor who focuses his research in GSE and teaches GSE course in a technical university.
2	Scrum Master	Involved in several GSE projects. Her company has been performing GSE for almost seven years
3.	Technical Director	More than one year as the Development Manager and Principal in the remote facility, and three years in the current position
	Product Manager	Principal Product Management for more than two years. Their company has been performing GSE for almost 17 years
4.	Method Engineering Course's Student Assistant	Expert in method engineering
5.	Service Delivery Manager	Almost two years in the current position that are working with global IT team for internal service development and operation.

The rationale for inviting the researcher is to obtain his feedback and critics from a person who has a broader perspective in global software engineering domain from the scientific standpoint. Other experts would be expected to provide their feedback and critics from their daily practices to assess the usability of the artifacts.

We provide a method base document that contains the background of this research, the diagrams and the description of the activities and concepts (PDD Documentation). In every cycle, the method base document is updated based on feedback from the previous session. The feedback and the evolution of the artifacts are presented in Section 7.2.

### 7.1.2 Method Evaluation Cycles and Criteria

The design science is an iterative approach in building solution artifacts. For evaluating the method, we applied the FEDS, a Framework for Evaluation of Design Science by (Venable, Pries-Heje, & Baskerville, 2016). As we propose artifacts are user oriented that should be evaluated with real users in their real context to fulfill the need of improving task coordination problems, we selected the “Human Risk and Effectiveness” strategy. Formative assessment starts the evaluation strategy and progressively the evaluation engages more summative assessment focusing on the applicability of the artifacts. The approach of our evaluation cycles is presented in Table 7-2.

*Table 7-2 Evaluation Cycles*

#Cycle	Method	Focal Points	Expert(s)
1	Criteria-based	Completeness, consistency, efficiency, reliability, applicability	Researcher from a technical university
2	Case Study	Perceived usefulness and perceived ease of use	Practitioners from AlphaSoft
3	Case Study	Perceived usefulness and perceived ease of use	Practitioners from BetaSoft
4	Criteria-based	Completeness, consistency, reliability	Method Engineering Expert
5	Case Study	Perceived usefulness and perceived ease of use	Practitioners from GammaSoft

The first evaluation adopts the criteria-based approach. We consider evaluating the model based on the criteria in assessing a method designed by method assembly approach (Brinkkemper et al., 1999), which are: Completeness, consistency, efficiency, reliability, and applicability.

1. Completeness: the situational method contains all the method fragments that are referred to by other fragments in the situational method.
2. Consistency: all activities, products, tools and people plus their relationships do not contain any contradiction and are thus mutually consistent.
3. Efficiency: the method can be performed at minimal cost and effort
4. Reliability: the method is semantically correct and meaningful
5. Applicability: the developers can apply the situational method.

For the rest evaluations, we involve real users to assess our design artifacts with a naturalistic setting that offers more critical face validity and also assures more rigorous assessment of the acceptance of the artifact (Venable et al., 2016). We adopt two constructs from Technology Acceptance Model (TAM) which are Perceived Usefulness and Perceived Ease of Use (Davis, 1989). TAM usually is used to test the behavioural acceptance or intention of

using information technology such as application framework (Polančič, Heričko, & Rozman, 2010), software process engineering tools (Wagenaar, Overbeek, & Helms, 2017), and a new designed method in software engineering (Koc, Timm, Espana, Gonzalez, & Sandkuhl, 2016). Perceived usefulness is defined by as "the degree to which a person believes that using a particular system would enhance his or her job performance." Meanwhile, perceived ease of use refers to "the degree to which a person believes that using a particular system would be free of effort" (Davis, 1989, p. 320). Since method engineering is used in the engineering of methods and tools in information system and technology domain (Brinkkemper, 1996), we assume that the adoption of TAM will be useful to evaluate the designed artifacts.

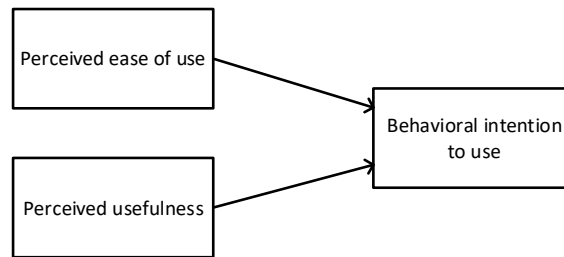


Figure 7-1 Method acceptance variables

## 7.2 Evaluation Results

### 7.2.1 1<sup>st</sup> Evaluation Session

The first assessment has been executed with Expert 1 as a CTO of a software development partner of product software companies and a researcher of global software engineering at a technical university in the Netherlands. As a partner, his company provides and manages the Scrum teams in Bangalore, India. The description of the context and development of GTC method are briefly presented, and the participant was positive towards to the presented artifacts as he summarizes that the artifacts would be a useful guideline for companies for managing tasks in GSE projects. However, there are some aspects need to be improved as described as follow (va-p101):

1. *Completeness.* The artifacts (framework and method) covers the practices of task coordination in GSE projects. However, the participant gave several notes, which are:
  - a. Regarding the socio-cultural aspect, the participant suggested focusing on individual levels. Someone's behaviors can be affected by his/her job characteristics, organization culture, family culture, team culture, and nationality culture. The previous studies in GSE that provide the discussion about socio-cultural aspects do not provide clear context about culture compared to the studies from social science domain. Then he suggested combining the socio-cultural identification with the study of Myers-Briggs Type Indicator (MBTI) in software engineering (Yilmaz, O'Connor, & Clarke, 2014).
  - b. There are missing stakeholders or unclearly described in the method base such as customers, management board members, and product owner. He suggested using abstract concepts such as "stakeholder" that can cover broad types of involved positions or roles.
  - c. The participants also added the importance of virtual teamness to reduce the perceived distance by increasing overlapping hours in the concurrent working

time window. This can be done by shifting the working hours to have bigger, making the remote team members visible through an online video camera and big screen in the working room. In addition, the business phone of the remote facility can be aliased number with the local country code that makes the customers or teammates feel that they are talking with their colleagues in the same country.

2. *Consistency.* The participant argued that communication and knowledge sharing cannot be separated. Communication always contains an information transferred between two or more peoples. Therefore, communication (especially in software engineering) is understandable as the mechanism to provide or share information.
3. *Efficiency.* The participants felt the artifacts are clear and can be followed easily. However, the participant felt that if there is enough time, he suggested performing a real case study to measure the efficiency quantitatively.
4. *Reliability.* There are several typing errors and unclear definition of the activities and concepts such as “horizontal causalities” and “vertical causalities”. The participants suggested to rephrase the terminologies or elaborate the concepts in the documentation. The participant also recommended to eliminate “synchronous communication” and “asynchronous communication” because these concepts are already defined by the communication mechanisms implicitly.
5. *Applicability.* Essentially, the participant thinks that the artifacts can be applied by companies. However, due to the limited time of discussion, the participant felt that he cannot contribute enough feedback for this criterion. He assumed that the following assessments with practitioners would provide better feedback.

By considering the expert's feedback, we made several adjustments to our artifacts:

**Combining “Communication” and “Knowledge Sharing”.** The participant suggested modifying knowledge sharing as part of the communication mechanism. His suggestion is also augmented by Rus, Lindvall, and Sinha (2001) which states that communication in the context of software engineering is often associated with the transfer of knowledge and collaboration is a form of mutual transaction knowledge. For example, when communication is done systematically, and there is a storage process as a document, the exchanged knowledge will be externalized and organized into organizational memory (Rus et al., 2002, p. 13).

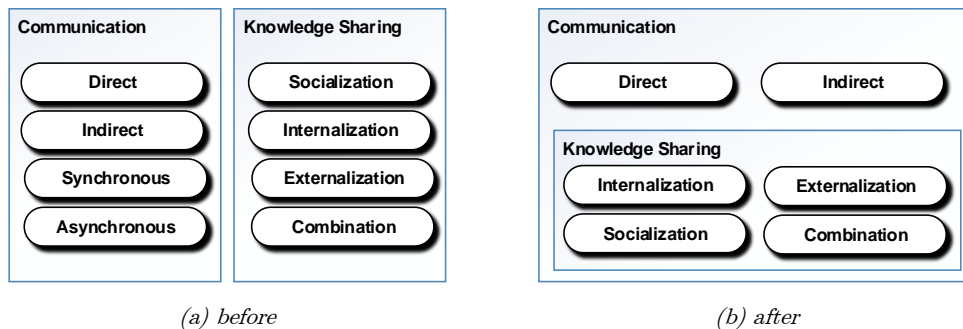


Figure 7-2 Merging “Knowledge Sharing” concept to “Communication”

As the concepts of knowledge sharing and communication are merged, the guidelines also should be adjusted as depicted in Figure 7-3. Communication itself is the central collaboration process where team members communicate their ideas, sharing their expertise, resources, and responsibilities (Chiu, 2002).

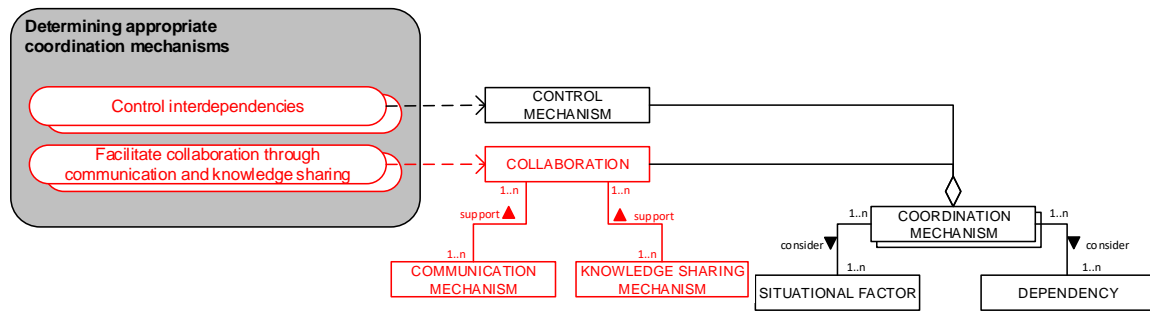


Figure 7-3 Adjusting communication mechanisms

**Elaborating stakeholders.** Impediments can come not only from the team members. Other stakeholders such as customers or business users, product owners, and management board members can raise impediments that might slow the engineering process. Thus, we elaborate these stakeholders in the routine activities as depicted in Figure 7-4.

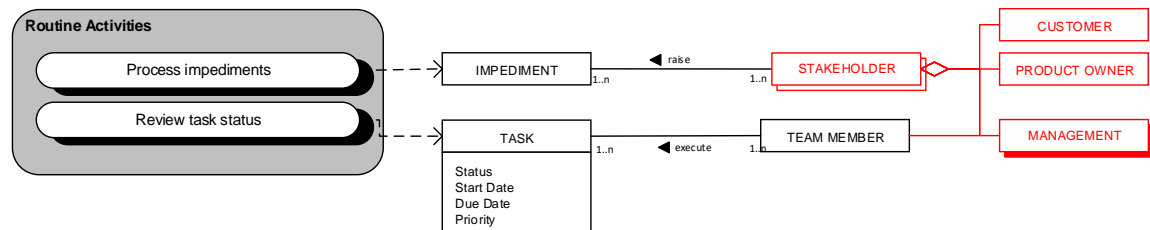


Figure 7-4 Elaborate other stakeholders

**Reducing “perceived distance”.** The participant suggested that adjusting working hours can help organizations to increase their opportunities in having more collaboration time (va-p1-1). Moreover, having a virtual office that makes the distributed teams can see each other at the real time can increase the opportunity of having direct communication. The virtual office also can be equipped with phone numbers or extension numbers that are network-aliased with local numbers that can bring the feel of having distance among distributed teams. The expert believes that these approaches can help the organization in building the teamness (Figure 7-5).

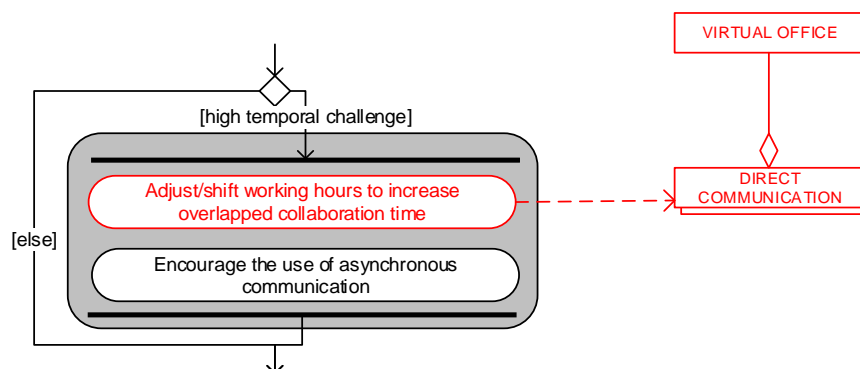


Figure 7-5 Reducing “perceived distance”

**Improving activities related to socio-cultural challenges identification.** As collaboration is involving two or more individuals, managers must consider each team member’s

behaviors. The participants suggested that managers can explore the culture starting from the culture of the origin country where the team members are coming from and based (va-p1-1). A site visit is also useful to identify the organizational culture and little bit deeper to the team culture. Job characteristics also can be a good start for managers in understanding each individual characteristic. Capretz and Ahmed (2010) noted that individual characteristics would provide the information of the ability in communication, interpersonal, cognition, and work attitude that eventually will form the team culture. Therefore, the adjustment has done by detailing the CULTURE concept and renaming the activities with more explicit description (Figure 7-6).

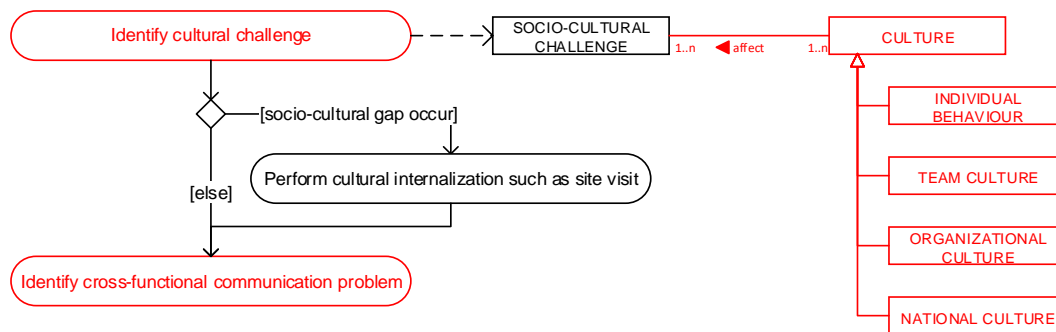


Figure 7-6 Adjustment for the vertical and horizontal cultural issues

### 7.2.2 2<sup>nd</sup> Evaluation Session

Our second assessment was performed by the Scrum Master from AlphaSoft. In general, she noticed that everything that is presented in the framework and method already includes what in the day-to-day coordination practices.

Nevertheless, the participant concerns that the guidelines should be elaborated more for the coordination control mechanisms. As a Scrum Master, she underpinned that facilitating distributed team members are not only serving them by providing room, distributing the meeting invitation and the minutes. A Scrum Master should have the “servant leadership” capability. A Scrum Master must be able to help team members to build their critical thinking to analyze the impediments, moderate the discussions, and encourage the team members to develop the best solutions by themselves without intervening the tasks allocation. As a Scrum team is a small team, she also conveyed that socio-cultural should be identified to the individual level. A national culture such as having more respect to the hierarchy might occur with people from the same country, but in daily practices, personal behaviors are more often seen in his/her interaction with his/her environment (iv-p2-1).

As a practitioner, the participant is required to provide her feedbacks about the perceived usefulness and ease of use as below:

**Perceived usefulness.** The participant sees that the method would be beneficial especially for companies which want to start to perform GSE. They can learn the aspects that should be prepared before deciding to start their GSE projects. Moreover, for those who have been performing GSE in their projects, the method would be useful as a reference where line managers or team leaders can go back when they find problems in coordinating interdependencies to see what activities that should be improved. The framework might be more useful for high-level managers. Meanwhile, the detail guidelines would be useful for development and operation teams (iv-p2-1).

**Perceived ease of use.** The participant sees that the method is presented in a technical manner. For her as part of the engineering team, she could understand the notation easily by reading through the diagrams and the process delivery diagram (method base) documentation. The framework also depicts a clear description on how the concepts are correlated each other. However, for those who do not have a technical background in system engineering, it would be better to provide them the framework and the guidelines by using picture diagram (iv-p2-1).

Based on her feedbacks, we improved our artifacts by applying her input to adjust our coordination control guideline as depicted in Figure 7-7.

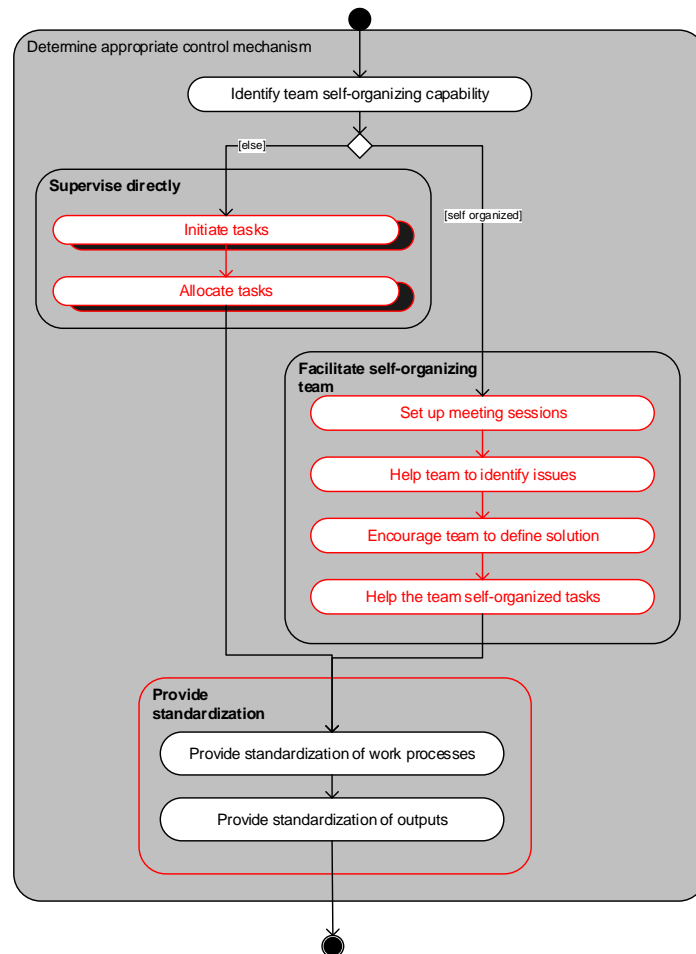


Figure 7-7 Adjusting control mechanisms

### 7.2.3 3<sup>rd</sup> Evaluation Session

The third evaluation was conducted in BetaSoft. The participants were the Technology Director and Product Manager. The Technology Director responded our method nicely by expressing that the task coordination model and the method recall and expose the practices they have done for more than ten years, as can be inferred from their statement:

*“I think it’s a useful method”... “Of course each should work out in the practical guidelines, in order to make it ease to use, but I think the change management should be taken into account of the method then it would be easier to use the method (va-p3-1).”*

Meanwhile, the Product Manager also conveyed to augment the Technology Director comments by reflecting their past experiences as follow:

*“The model is useful and we recognize a lot of things... There’s part of the method that can help us in different ways (of coordination). We also think that it’s easy to use because we already used to it. We still can use the guidelines (va-p3-1).”*

They also add that, although they have made many improvements, still at this time, they are still striving to improve and to streamline the current processes. Referring to the coordination level categorization as shown in Table 6-3, both participants felt they had routinely monitored their coordination activities and quantitatively measured the overall engineering process. Communication still becomes the most crucial issue, especially related to communication with the non-technical team, that is how to align product engineering with the management team, the marketing team, and the sales team (va-p3-1).

To that end, both participants suggested to add and to elaborate several aspects as below:

1. An organization’s business strategy covers a long term strategy and short term strategy. The long-term strategy usually defines what the organization wants the business to be like in the next five or 10 years. Meanwhile, the short-term strategy, also called as the business objective, defines organization’s initiatives as the gradual process to achieve the long-term goals (va-p3-1). In product software engineering, the initiatives are recognized as a progressive and continuum of approaches that involve business, development, and operations aspects (Fitzgerald & Stol, 2017).
2. Regarding above feedback, change management is always needed when organization needs to implement a new strategy or methodology. The product manager underpinned that change management identifies the current situations and the intended (future) situations and how an organization can handle the gap between these two situations. Software engineering has been considered as a socio-technical system (Fuggetta & Di Nitto, 2014). Thus, change management should be a focus on the people development that will add value to the business (Cristal, Wildt, & Prikladnicki, 2008). For example, when the organization moved to Scrum methodology to improve software engineering practices, including the process that involved the offshore development office, they started to practice Scrum (such as the daily stand-up, the review, and the retrospective meetings) with the lowest technical level. They have been continuously and gradually expanding the practices to involve broader business functions and higher organization levels (va-p3-1, (va-p3-2).
3. An organization should regulate the formation of subject matter experts by segmenting the expertise to provide a clear work separation but at the same time managing the relationships between different domain experts to streamline the coordination process. This practice promotes the interdisciplinary coordination and converges the cross functions collaboration (va-p3-1). Based on this feedback, we elaborated our guideline in managing knowledge dependencies by redefining the “Identify knowledge location” step as “Organize knowledge and experts as organization’s assets” to accentuate the importance of managing domain experts and their expertise to catalyze knowledge flows in a distributed organization.
4. Related to the communication model in BetaSoft described in Figure 4-7, the Technology Director criticized the absence of Scrum Master role in the model. He noticed that the function of organizing and facilitating communication at the technical



level are the responsibilities of a Scrum Master. The Development Manager is not directly involved in product development viewed from the communication management and process control standpoints. The Development Manager is responsible for assisting the team in the execution of tasks and protects the team from impediments, but is not involved in the team’s internal decision-making. More precisely, the determination for the internal task management is delegated to the team itself in determining the best task distribution and execution. Thus, the team will sense to take the ownership of the product even though they are not in the same location with product management team at the head office (va-p3-1).

As a practitioner, the participants are required to provide their feedbacks about the perceived usefulness and ease of use as below:

**Perceived usefulness.** With almost 17 years of experience in GSE, the participants judged that they were mature enough in coordination management with their team in Kuala Lumpur. They already feel the challenges faced due to differences in time, distance, and culture. Judging from their maturity, they reflect on what they have been through, so the participants infer that the methods introduced already include the practices they undertake (va-p3-2). However, they can also learn from the practices of other companies, for example, the use of virtual meetings to build more strong communication and social closeness between better-distributed teams.

**Perceived ease of use.** The participants see that the method would be easy to use for them because they have been performing GSE for years, feeling the challenges and gradually improving their coordination practices to address the emerging issues. Again, change management is also essential if organizations want to apply this method so that the method can be adopted and implemented more easily (va-p3-2). Change management should gradually be done when organizations want to adopt this method this method suggests some practices that might change their existing practices. This consideration was suggested by the fact when BetaSoft wanted to change traditional software engineering methods to Scrum six years ago, BetaSoft has been managing the implementation of Scrum starting slowly from the lowest technical level and progressively increasing Scrum scalability at a higher organizational level (va-p3-2).

Based on the feedback, we considered the importance of change management. Thus, we decided to elaborate the first step of our method as can be seen in Figure 7-8.

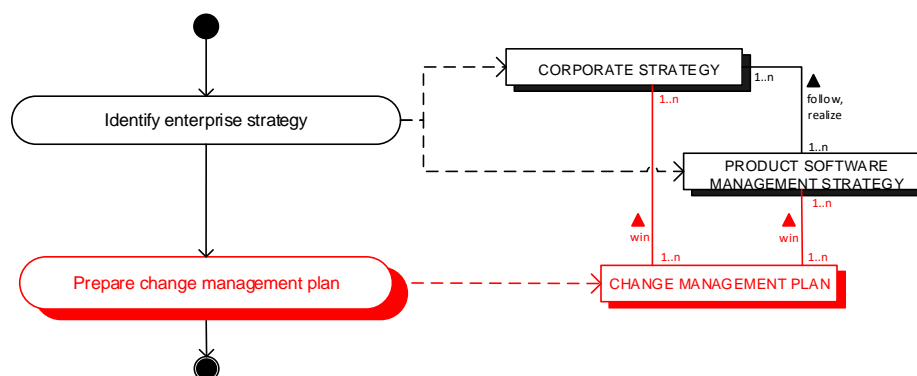


Figure 7-8 Improving task coordination preparation step

#### 7.2.4 4<sup>th</sup> Evaluation Session

This time, the evaluation aimed to assess the completeness, correctness, and consistency of the method from the method engineering technique perspective. The participant was a student assistant of the Method Engineering course at Utrecht University. Regarding the completeness, she indicated that the method has already elaborated the primary processes and concepts related to task coordination in global software engineering. On the other hand, reliability became her main critic where several concepts with different names have the same meaning such “SOCIO-CULTURAL” and “SOCIO-CULTURAL CHALLENGE”. She also noticed that the use more abstract naming (such as using “ACTOR” instead of “TEAM MEMBER”) and simplifying the activity label would help to maintain the consistency (va-p4-1).

However, besides the feedback related to the correctness in the use of PDD notations and the consistency of the activity and concept names, she noticed that the method should be adjusted to improve the readability. Previously, the method consisted nine PDDs that makes difficult to maintain the traceability and the internal consistency. Finally, the top two PDD levels are merged, and the detail level PDDs are altered to conform the high-level PDD and the task coordination framework. The revised version of the PDDs are presented in Appendix F.1.

#### 7.2.5 5<sup>th</sup> Evaluation Session

The last assessment was performed with the Service Delivery Manager from GammaSoft. During the walkthrough of the research motivation and the method, several comments were expressed that led to series of discussion. In general, the expert has a positive attitude towards both the method and the framework. He noticed that the method provides a broad overview of task coordination in global software engineering. Furthermore, he provided a positive comment by commenting:

*“I like the overview that you have that really helps me. It’s more than just theoretical. I’ve learned a lot (va-p5-1).”*

Moreover, the participant commented to the perceived usefulness and the perceived ease of use as below:

**Perceived usefulness.** He indicated that GammaSoft is still struggling with the socio-cultural problems caused by intern-organizational silos and cross-cultural differences with a vendor in India such as the way of communicating as well as the difference in their first language. The lack of expertise of the young employees from the vendor increases the negative impact on the overall work performance. These situations motivate the expert to look forward the usefulness of the method (va-p5-1). As the Service Delivery Manager in GammaSoft, he must collaborate with the Service Coordinator from the development vendor and the development and support teams in India. Eventually, he certified that the method provides not only theoretical overview but also useful practical guideline.

**Perceived ease of use.** *During the presentation, we also presented several visualizations of the concepts such as depicted in Figure 5-2, Figure 5-3, Figure 5-4, Table 6-1, and*

Table 6-2. These visualizations as well as the GSE Task Coordination Framework complement the method and provide a practical guideline (va-p5-1). As the participant’s technological background, the method presented by using PDD as the modeling

language also gives a clear explanation of the activities, practices, and concepts involved within the guideline.

A missing subject that he really wanted to hear throughout the presentation is how the method would help the organization in managing the governance of task coordination among distributed team member. The participant argued that a strong leadership would be needed for companies such his company, GammaSoft, still must struggle with global dispersion issues especially cultural, knowledge, and expertise. Governance is defined as the action, manner, or system of governing<sup>5</sup>. This definition is closely related to the authority and top-to-bottom coordination control mechanism. In global software engineering, governance can be perceived as the ability or exercise within the organization to control the distribution of work by assigning the roles who have the authority to supervise the dispersed team member. When the distributed teams can manage dependencies by themselves, the authority in governing the software product engineering is shared as bottom-up 'empower and reflect' situations to make the team members also take part the responsibility of the decision making (Talby & Dubinsky, 2009).

The participant, as well as Bannerman (2009), see governance as different from management, and governance is a multidimensional concept which for example can be considered as a method, strategy or process. There is no an absolute effective governance approach, in which depends on the characteristics of the organizational circumstance. By considering from a meta-management perspective, governance is a cell that compromises several elements: purpose, structure, process, and relational mechanism. A governance cell can be applied to particular domains within the scope of product software management, e.g. board of directors, product development steering committee, and technical / development board. This cell governance can be used to define a distributed software development governance to meet its engineering and business needs (Bannerman, 2009). By considering our participant's feedback and the foundation of depicting software development governance presented by Bannerman (2009), we decided to extend our method by adapting the software development governance concepts.

As this evaluation session is the final session, the final global task coordination method is achieved. In the next chapter, we summarize the findings from all the five evaluation sessions, present the ultimate version of the global task coordination method as shown in Appendix F.2, and discuss the limitations of this research.

---

<sup>5</sup> <http://www.dictionary.com/browse/governance>



## **PART FOUR    CLOSING**

**Chapter 8. Discussion**

**Chapter 9. Conclusions**



# Chapter 8      DISCUSSION

In this chapter, we will reflect on the findings and discuss the most relevant results. Nonetheless, while this study earned valuable insights, this research and its artifacts are subject to some limitations. Therefore, the discussion in this chapter will be guided by the synthesized findings and the borders of this design science project.

## 8.1 Evaluation Summary: The Synthesized Findings

To start, it is important to re-emphasize the context in which this discussion takes place: a methodological support for task coordination in global software engineering projects at product software companies. We argue that product managers and those who are closely related to the product software engineering should be able to manage task interdependencies among globally distributed team members. There are studies performed by scholars that propose solutions to manage tasks where coordination becomes the crucial part for a successful global software engineering projects. Each study is focusing on a certain topic such as improving internal development team coordination through Agile practices adoption, improving project management and control with the use of PMBOK® guidelines, and facilitating knowledge to enhance collaboration among scattered engineering teams. At the same time, we also considered that every product software company with its internal organization and product engineering process complexity raise its situational backgrounds that could make the way organization manages the task dependencies is unique one to another. These considerations became our motivation to perform this research.

Subsequently, we performed five semi-structured interviews to obtain more insights from the practices carried out by product software companies in the Netherlands. Four of the companies are performing global software engineering projects by offshoring or nearshoring parts of the development activities or product engineering processes. The interview results confirmed the coordination mechanisms explained by the literature and provided valuable insights how companies analyze their capabilities through managing processes, improving organization infrastructure, and optimizing tools to develop their coordination mechanisms.

Based on the results, we developed a task coordination framework for organizations that perform their product software engineering globally. The first version of the GSE task coordination framework depicts three main mechanisms in a globally distributed collaboration works to achieve a coordinated output, which is: Control mechanism, communication mechanism, and knowledge sharing mechanism (Figure 8-1). These mechanisms are supported by the organization itself that develops a structure to the roles and functions that perform the coordination activities. A strategy also should be provided to underlie and guide the actors in managing the tasks. The coordination actors also must be aided by the tools that help the

actors in performing better communication, managing the project, facilitating the collaboration, and promoting the knowledge flows. These coordination practices are affected by how the organization is distributed, what strategies do they have, and challenges that they should address.

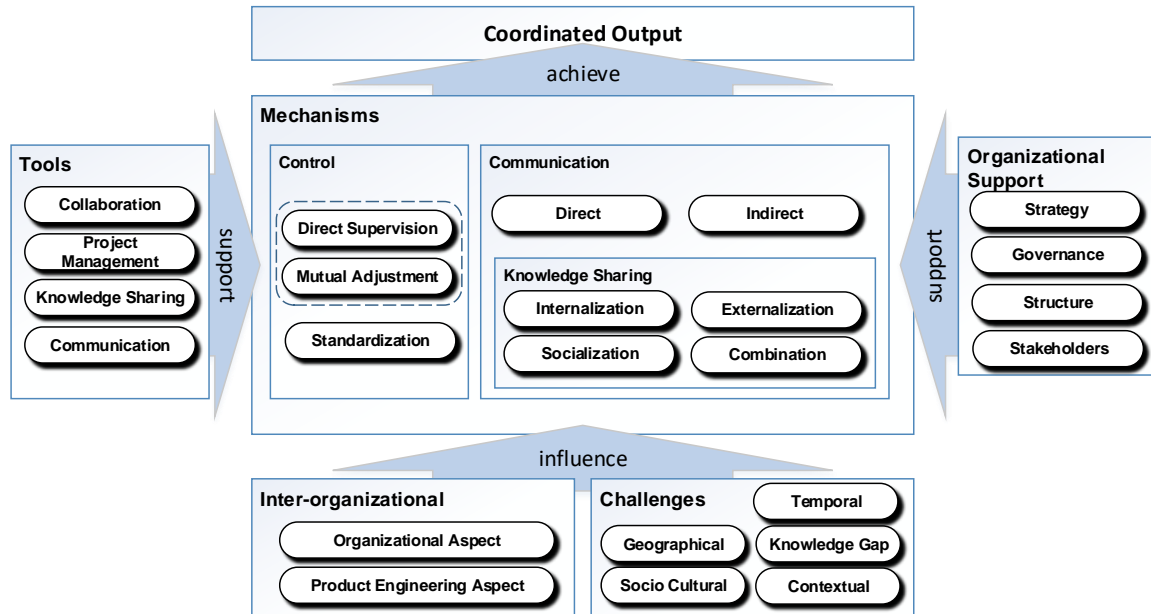


Figure 8-1 The Final Global Task Coordination Framework

We also suggests organizations take a look the guideline that we provide refer to the coordination framework: (1) Identify organization strategy; (2) Assess the situational backgrounds which are organization internal aspects and challenges caused by the distribution of the teams; (3) Assess the supports, which are organizational supports and tool supports; (4) Determine organization coordination practices from the project control perspective and communication perspective by considering its situational factors; and (5) Reflect and plan to continuously improve the practices (Appendix F.2, Figure 9-5).

To evaluate our research artifacts, we conducted five interview sessions with different roles from three product software companies, a global software engineering researcher, and a method engineering student assistant. From the interviews, we conclude that practitioners think that the task coordination framework and method have covered their daily practices in managing product engineering projects that involve team members or vendors in other countries. They positively respond to the applicability of the artifacts by stating that the framework provides a contextual presentation of coordination perspectives and the method elaborates the framework in a more detail guideline. However, we also noticed that the researcher and the student assistant are inclined to negatively respond to the readability of the first version of the method due to its complexity and low-level granularity. By considering our participants' feedback, we improved iteratively and presented a final version of the artifacts.

### 8.1.1 Meta-Modelling Criteria Viewpoint

The GSE task coordination framework and method were assessed by using process modeling criteria by



**Completeness.** The participants were satisfied with the framework and the method. The practitioners indicated that the method not only covers the practices that have performed but also provides the new practices captured from other companies. The method also describes the roles who are responsible for the specific activities in managing the distributed works and team members. At the same time, the framework that gives the holistic overview of task coordination approach for product software companies to assess their situational background and the required support.

**Consistency.** The attempt to provide a guideline at more detailed levels threaten coherence of the developed method. The first-round evaluation directly criticized the consistency issue related to the relationship between communication and knowledge sharing in domain software engineering. In the subsequent rounds of assessment, participants found that the concepts and the activities are autonomous and mutually consistent.

**Efficiency.** Partly satisfied. The scientific experts argued that the method will not be easy to be followed by non-technical users due to the complexity and granularity. Indeed, as noticed by the practitioners, the artifacts cover all task coordination aspects in global software engineering because the artifacts attempt to cover broad topics. It is a challenge to provide a solution that comprises broad issues, which on the other hand, the solution should also present a clear explanation and applicative guideline.

**Reliability.** Satisfied. During the evaluation sessions, some disagreements and suggestions of the terminologies were conveyed by the participants. In the first session, the expert suggests using more specific and general terminologies to avoid misperception and uncertainty, while in the second session the expert suggested that the control mechanism should be elaborated. Then, we find it difficult to keep the method compact. After the fourth session, based on the suggestion from the expert we modified the model and optimized the documentation to make the method more concise. It is easier to maintain the reliability and consistency of concepts and activities presented in the method.

**Applicability.** The practitioners indicated that both the method and the framework could be applied as a reference guideline where they can come back to see when they need it as well as in their daily practices. The Section 8.1.2 discuss the applicability from the perspective of behavioral intention to use by discussing the perceived usefulness and the perceived ease of use of the artifacts.

### 8.1.2 Behavioral Intention to Use Viewpoint

The practitioners as the participants of the expert validation sessions indicated to have an intention to use the GSE task coordination method. The participants from BetaSoft was enthusiastic and considered the usefulness and ease of use of the method even though they have been doing global software engineering for more than ten years. Meanwhile, the participant from AlphaSoft indicated that the framework could be useful for those who have higher management roles and the detail guidelines will be helpful for line managers and team leaders. We noticed that the experts preferred to see the method as a set of best practices guideline where they can come back anytime, assess their current situation to detect the coordination deficiencies while enhancing their coordination practices. The practitioners could see the benefits of the method. They notified that they are very pleased with the method and desire to use the method in their daily practices.

## 8.2 The Final Global Task Coordination Method

The iterative evaluations obtained useful feedback and critics to be analyzed that led to several areas of improvement (Section 7.2) as summarized in Table 8-1. This section presents changes to the evaluated version of the GSE task coordination framework and method, which reach to the final version of the artifacts (depicted in Figure 9-5 in Appendix E). The following table depicts the summary of the changes have been done throughout the evaluation and validation session.

*Table 8-1 Method Evolution Summary*

Improvement Aspects	Targeted Points
Elaborating missing activities and concepts	<ul style="list-style-type: none"> <li>• Elaborate the variability in cultural issues,</li> <li>• Specify the missing stakeholders</li> <li>• Elaborate the control mechanisms</li> <li>• Specify Change management to facilitate the ability of the stakeholders to face the changes of organizational strategies and methods</li> <li>• Specify governance in product software development</li> </ul>
Beautification	Simplify the method to improve the readability of the PDDs
Improving consistency and integrity	<ul style="list-style-type: none"> <li>• Merge communication and knowledge sharing</li> <li>• The simplification also used to maintain the consistency and integrity among the concepts and activities in the PDDs</li> </ul>

## 8.3 Limitations

**Construct Validity.** We used multiple data sources to construct our method. First, we conducted literature studies (Section 2.2.1) and conducted investigation interviews (Chapter 4) with five companies with different stakeholders. There were product managers, technology director, team leader, Scrum master, and development manager participated for in the interviews. Each company is also distributed differently from the perspective of engineering processes, distribution and organizational structure. We also performed the validation phase by involving both scientific experts and business experts. We consider these approaches to ensure that the method is built comprehensively examined and gained objective judgments not only from a single point of view.

However, due to time constraints, we decided to conduct the validation phase by using expert opinions that focus more on the desire to use the method. To perform a measurement test such a desired tangible output through the application of this method, for example, a measurement of tasks hand over effectiveness in a global software engineering project, a longitudinal case study which takes longer observation is required. Although the experts claimed that this method embraces a holistic overview and positively accept the method, this limitation is obviously a threat to the construct validity of the method.

**External Validity.** External validity refers to the extent to which the outcomes of this research can be generalized to other contexts. Three of the four participating product software companies adopt Agile methodology in their software engineering processes. In addition, these companies are based in the Netherlands. The evaluation also involved participants from the Netherlands. We tried to maintain the external validity by selecting companies with a different

characteristic of global distribution. Companies can be distributed as a holding organization, through a partnership with other companies, or a combination of both. The distributed organizations may work on different tasks but also on the same tasks. Also, the dispersion factors as mentioned in Table 3-1 characterize each company differently. Nonetheless, it may be possible that another investigation phase and validation phase at another organization outside the Netherlands yields different results.

**Reliability.** The objective of the reliability test is to be sure that if a later researcher followed exactly the same procedures and conducted the same case study, the subsequent researcher should arrive at the same findings and conclusions. We aimed to perform a highly reliable research by documenting all the research activities, the protocols, and providing the linkage between the discussion comprehensively. However, a limitation regarding the reliability is that the results of the investigation phase and the validation phase are heavily dependent on the experience of the experts, which possibly will raise a threat to the reliability of this research.



# Chapter 9 CONCLUSIONS

Product software companies involve complex factors in their software engineering processes. The product becomes an integral part of the organization itself that asks for the involvement of many parties in the process of ideation, engineering, and management that ensures the success and continuity of the products produced. An intuitive thought arises, as the complexity increases in situations where engineering processes are carried out in a globally distributed environment: the need to coordinate tasks and teams that will be influenced by the differentiating factors that make coordination practices unique for each organization. This thesis analyzes the challenges facing these organizations, how to overcome these problems, and those involved by observing situational stipulations that may reshape the implementation of coordination practices.

## 9.1 Results

Guided by the following main research question, we follow design science framework to present an answer and validate the deliverable: *“How can we provide methodological support for the improvement of task coordination in global software engineering projects in a product software company?”* We developed various sub-research questions that guide us to answering the main question that will be briefly discussed in the following sections.

**SQ1:** What are the current task coordination challenges in global software engineering?

The term challenge in our research context refers to a set of issues that can limit or result in risks to achieve a successful global software engineering projects. In Chapter 3, we distinguish many types of diversities caused by the distribution. There are several challenges in coordinating tasks among globally distributed teams. The geographical distance shows how teams are distributed in different locations spatially that restrict the organization to have direct communication. The time-zone difference (temporal) difference for companies that have distributed teams at other continents limits the opportunity of having overlaid collaboration time. The socio-cultural challenge can occur in many levels. A team can be characterized by its organizational or team culture. However, for small-sized distributed team, the organization should manage this challenge to the individual level. The difference of knowledge and expertise also can increase the dependency that harms the information flows. Last but not least, the difference of process, method, experience and maturity among distributed locations cause incompatibility that can misuse the process flows.

**SQ2:** What are the current practices performed by product software companies in executing global software engineering projects?

There are many best practices and methods found from the literature and during the interviews of which we organize into two main mechanism categories: Control and Communication mechanisms. Control mechanism provides the overview of the practices in

managing vertical coordination that involves the role who has the authority to manage the interdependencies among distributed teams. This vertical coordination can be seen as top-to-bottom approach which is undertaken through authorized entities such as line managers, project managers, or functional managers, and bottom-up approach that involves mutual adjustments through distributing responsibility in managing tasks among peers by themselves. Both approaches need to be complemented by the standardization of work process, methodology, and work output that all distributed teams must follow for fluid task switching.

Direct communication cannot be replaced in any situation, even for companies that performing global software engineering projects. Companies should try to increase the intensity and optimize their opportunity for direct communication although the chance to collaborate only a little and difficult to have face-to-face communication enough. Some best practices such as regular site visit, virtual office, and the daily stand-up meeting through video conference can be done to increase the intensity of direct communication. However, indirect communication comes with an agreed communication protocol between distributed teams are suggested to fill the communication deficiency caused by the shortage of direct communication.

In software engineering, communication is the mechanism that allows information flows. Due to the possibility of knowledge gap and problems in accessing the knowledge, companies are suggested to find the best approach to facilitate the knowledge flows. The team size and the organization culture defines the knowledge sharing mechanism. The organization should be able to identify the location of the knowledge and provide access to the knowledge.

**SQ3:** What method can be designed to facilitate companies for coordinating tasks in global software engineering projects?

The answer to this sub research question yields the GSE task coordination method resulting from the literature review and interviews. To start, a framework provides the general overview to help companies understand what are the related concepts in task coordination in global software projects. The framework depicts that coordination mechanisms should be supported by organizational support and tool support. Organizational support refers to organization structure, roles, and governance. Tool support refers to a collaboration tool, communication tool, project management tool, and knowledge sharing tool. The appropriate coordination mechanism can differ from one company to others, which can be influenced by the inter-organizational factors (such as configurational factor, distribution strategy, and software engineering methodology) and challenges caused by the dispersion (geographical, temporal, knowledge, socio-cultural, and contextual diversities)

The method consists of five main activities:

1. Business analysis: determine the organization strategy, prepare for the change management plan
2. Situational analysis: analyze situational analysis, identify dependencies, and identify coordination profile
3. Support analysis: identify required supports, assess coordination support, identify gap or requirements, determine the collaboration governance, add required collaboration support as non-functional requirements.
4. Task coordination: perform routine activities, determine appropriate control mechanism, and determine appropriate communication mechanism
5. Finalization and improvement: review current practices and determine improvement plan

In these five activities, a coordination mechanism is not an individual concept that stands alone in daily software engineering activities. Considering which task coordination mechanisms that are appropriate for a product software company should consider the business strategy and product engineering strategy. The method is also seen as a continuous improvement activities to adapt the changes of the influential aspects and enhancements that have been done before. We provide two coordination matrixes and coordination mechanism profile (pragmatic and methodological) that depicts the mechanisms cultivate from the interviews and literature review that can be followed in the method.

**SQ4:** How to improve the developed method in task coordination after validation by considering its benefits and drawbacks?

To finalize the method, an iterative validation approach inspired by method evolution approach and the Framework for Evaluation of Design Science (FEDS) was performed with five different institutions (Chapter 7 ). Two sessions were done with a participant with a scientific background to assess the method from the method engineering perspective, and three sessions were conducted with the practitioners to evaluate the applicability and intention of use from the usefulness and ease of use perspectives. Overall, as has been summarized in Section 8.1, the participants are satisfied with the method. The method successfully covers both theoretical and practical aspects. They have seen that the method reflects their daily activities and provides some suggestion or best practices that might be useful for the improvement of task coordination in their companies. However, some feedbacks and critics were raised, but we saw them as useful input that triggered us to evaluate and at the end improve the method.

**RQ:** “How can we provide methodological support for the improvement of task coordination in global software engineering projects in a product software company?”

The answer comes in the form of the Global Coordination Method tailored using the references from the literature and best practices from participant companies. Overall, the method was perceived positively. The participants acknowledged the benefits of the method for its completeness and flexibility in combination with its applicability. The method promotes the explicit relationships between task coordination practices, the supporting bases, and the situational backgrounds thereby embracing the overall aspects to the application of the method such as preparation, execution, and evaluation. Yet, by considering the benefits of the method, the participant indicated the intention to adopt the method. However, the method is certainly not faultless. As the method developed and evaluated based on best practices by companies that have been operating global software engineering and most of them are practicing Agile methodology. The method should offer the guideline for the change management and software development governance aspects in more detail especially for product software companies that want to commence global software engineering projects. Finally, the findings suggest that GSE task coordination method enables managers to handle challenges and interdependencies in global software engineering projects. For a better result, the method should be performed continuously and involve all key areas of a product software company.

## 9.2 Future Research

This research results and its limitations give opportunities for other extended researches in several directions. First, the evaluation was performed by using expert opinion approach that limits the depth of the evaluation. Therefore, when the time is on the side of both researcher and participants, we suggest a longitudinal case study or an action research that integrate a real situation in PSOs daily global software engineering processes can be performed to evaluate

the usefulness and applicability of the method reliably. Action research requires the participation of the research participants in the implementation of the proposed solution in which the researcher becomes part of the participants. Action research offer an in-depth and first hand understanding the researcher obtains. Meanwhile, a longitudinal case study is the researcher becomes an investigator rather than participant and performs integrated observation over long periods of time (Benbasat, Goldstein, & Mead, 1987).

Secondly, related to the above suggestion, this method also has not presented quantitative measurement that indicates its contribution to the business practices e.g. accelerating the process of hand over work between the distributed teams for a more efficient GSE project, or increases the satisfaction of the distributed team members in daily coordination practices. An action research or longitudinal case study also allows researchers to make some measurements that show more tangible benefits of this method.

Thirdly, the GSE task coordination method itself was solely validated by participating experts from companies in the Netherlands. Hence, the method still can be generalized to a larger extend by conducting another research with experts and companies from other countries. Last but not least, several activities and concepts were added based on the feedback emerged during the validation phase may require further validation.



# REFERENCES

- Ågerfalk, P. J., Fitzgerald, B., Olsson, H. H., & Conchúir, E. Ó. (2008). Benefits of Global Software Development: The Known and Unknown. In *Making Globally Distributed Software Development a Success Story* (Vol. 5007, pp. 1–9). Berlin, Heidelberg: Springer.
- Altmann, J. (1999). Cooperative software development: concepts, model and tools. In *TOOLS '99 Proceedings of the Technology of Object-Oriented Languages and Systems* (p. 194).
- Ancona, D. G., & Caldwell, D. F. (1992). Bridging the Boundary: External Activity and Performance in Organizational Teams. *Administrative Science Quarterly*, 37(4), 634–665.
- Arora, A., & Gambardella, A. (2004). *The Globalization of the Software Industry: Perspectives and Opportunities for Developed and Developing Countries. Policy* (Vol. 5). Cambridge, MA.
- Artz, P., van de Weerd, I., Brinkkemper, S., & Fieggen, J. (2010). Productization: Transforming from Developing Customer-Specific Software to Product Software. In *First International Conference, ICSSOB 2010* (pp. 90–102). Utrecht, the Netherlands.
- Bannerman, P. L. (2009). Software Development Governance : A Meta-management Perspective. In *2009 ICSE Workshop on Software Development Governance* (pp. 3–8).
- Bekkers, W., Spruit, M., van de Weerd, I., Van Vliet, R., & Mahieu, A. (2010). A Situational Assessment Method for Software Product Management. In *Proceedings of the 18th European Conference on Information Systems* (pp. 1–12). Pretoria, South Africa: AISel.
- Bekkers, W., van de Weerd, I., Spruit, M., & Brinkkemper, S. (2010). A Framework for Process Improvement in Software Product Management. In A. Riel, R. O'Connor, S. Tichkiewitch, & R. Messnarz (Eds.), *Software and Services Process Improvement* (Vol. 99, pp. 1–12). Berlin, Heidelberg: Springer.
- Benbasat, I., Goldstein, D. K., & Mead, M. (1987). The Case Research Strategy in Studies of Information Systems. *MIS Quarterly*, 11(3), 369.
- Bertram, M., Schaarschmidt, M., & Von Kortzfleisch, H. F. O. (2012). Customization of product software: Insight from an extensive is literature review. *IFIP Advances in Information and Communication Technology*, 389, 222–236.
- Bjørnson, F. O., & Dingsøyr, T. (2008). Knowledge management in software engineering: A systematic review of studied concepts, findings and research methods used. *Information and Software Technology*, 50(11), 1055–1068.
- Bosch, J., & Bosch-Sijtsema, P. M. (2010a). Coordination Between Global Agile Teams: From Process to Architecture. In *Agility Across Time and Space* (Vol. 23, pp. 217–233). Berlin, Heidelberg: Springer Berlin Heidelberg.
- Bosch, J., & Bosch-Sijtsema, P. M. (2010b). From integration to composition: On the impact of software product lines, global development and ecosystems. *Journal of Systems and Software*, 83(1), 67–76.
- Brinkkemper, S. (1996). Method engineering: engineering of information systems development methods and tools. *Information and Software Technology*, 38(4), 275–280.
- Brinkkemper, S., Saeki, M., & Harmsen, F. (1999). Meta-modelling based assembly techniques for situational method engineering. *Information Systems*, 24(3), 209–228.
- Brisaboa, N. R., Cortiñas, A., Luaces, M. R., & Pol'la, M. (2015). A Reusable Software Architecture for Geographic Information Systems Based on Software Product Line Engineering. In *Lecture Notes in Computer Science* (Vol. 9344, pp. 320–331).
- Budgen, D., & Brereton, P. (2006). Performing systematic literature reviews in software engineering. In *Proceeding of the 28th international conference on Software engineering - ICSE '06* (Vol. 45, p. 1051).

- New York, New York, USA: ACM Press.
- Capretz, L. F., & Ahmed, F. (2010). Making Sense of Software Development and Personality Types. *IT Professional*, 12(1), 6–13.
- Carmel, E., & Agarwal, R. (2001). Tactical approaches for alleviating distance in global software development. *IEEE Software*, 18(2), 22–29.
- Carmel, E., Espinosa, J. A., & Dubinsky, Y. (2010). “Follow the Sun” Workflow in Global Software Development. *Journal of Management Information Systems*, 27(1), 17–38.
- Chiu, M.-L. (2002). An organizational view of design communication in design collaboration. *Design Studies*, 23(2), 187–210.
- Cisco. (2017). Cisco WebEx. Retrieved June 21, 2017, from <https://www.webex.com/why-webex/overview.html>
- Cohen, D., & Crabtree, B. (2006). Semi-structured Interviews Recording Semi-Structured interviews. Retrieved December 16, 2016, from <http://www.qualres.org/HomeSemi-3629.html>
- Cohn, M. (2007). Differences Between Scrum and Extreme Programming. Retrieved February 22, 2017, from <https://www.mountaingoatsoftware.com/blog/differences-between-scrum-and-extreme-programming>
- Conchúir, E. Ó., Ågerfalk, P. J., Olsson, H. H., & Fitzgerald, B. (2009). Global Software Development: Where are the Benefits. *Communications of the ACM*, 52(8), 127–131.
- Cossentino, M., Gaglio, S., Henderson-Sellers, B., & Seidita, V. (2006). A metamodeling-based approach for method fragment comparison. *CEUR Workshop Proceedings*, 364, 57–70.
- Cristal, M., Wildt, D., & Prikladnicki, R. (2008). Usage of SCRUM Practices within a Global Company. In *2008 IEEE International Conference on Global Software Engineering* (pp. 222–226). IEEE.
- Crowston, K. (1994). *A Taxonomy Of Organizational Dependencies and Coordination Mechanisms. Organizing Business Knowledge: The MIT Process Handbook*. Michigan, USA.
- Culture. (2017). Retrieved June 21, 2017, from <https://www.merriam-webster.com/dictionary/culture>
- D’souza, A., Kabbedijk, J., Seo, D., Jansen, S., & Brinkkemper, S. (2012). Software-as-a-Service: Implications for Business and Technology in Product Software Companies. In *Proceedings of the Pacific Asia Conference on Information Systems (PACIS)* (p. Paper 140).
- Davis, F. D. (1989). Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. *MIS Quarterly*, 13(3), 319.
- Deneckère, R., Hug, C., Onderstal, J., & Brinkkemper, S. (2015). Method Association Approach: Situational construction and evaluation of an implementation method for software products. In *Proceedings - International Conference on Research Challenges in Information Science* (Vol. 2015–June, pp. 274–285). Athens, Greece: IEEE.
- Deshpande, S., Beecham, S., & Richardson, I. (2011). Global software development coordination strategies - A vendor perspective. *Lecture Notes in Business Information Processing*, 91 LNBIP, 153–174.
- Espinosa, J. A., & Carmel, E. (2004). The effect of time separation on coordination costs in global software teams: a dyad model. *37th Annual Hawaii International Conference on System Sciences, 2004. Proceedings of the*, 0(C), 1–10.
- Espinosa, J. A., Lerch, J., & Kraut, R. (2002). *Explicit vs. Implicit Coordination Mechanisms and Task Dependencies: One Size Does Not Fit All*. Pennsylvania, USA.
- Espinosa, J. A., Nan, N., & Carmel, E. (2007). Do gradations of time zone separation make a difference in performance? A first laboratory study. In *Proceedings - International Conference on Global Software Engineering, ICGSE 2007* (pp. 12–22). Munich, Germany: IEEE.
- Espinosa, J. A., Slaughter, S. A., Kraut, R., & Herbsleb, J. (2007). Team Knowledge and Coordination in Geographically Distributed Software Development. *Journal of Management Information Systems*, 24(1), 135–169.
- Faraj, S., & Sproull, L. (2000). Coordinating Expertise in Software Development Teams. *Management Science*, 46(12), 1554–1568.
- Fitzgerald, B., & Stol, K.-J. (2017). Continuous software engineering: A roadmap and agenda. *Journal of Systems and Software*, 123(0), 176–189.
- Fricker, S. (2012). Software Product Management. In A. Maedche, A. Botzenhardt, & L. Neer (Eds.), *Software for People* (Vol. 31, pp. 53–81). Berlin, Heidelberg: Springer Berlin Heidelberg.
- Fuggetta, A., & Di Nitto, E. (2014). Software process. In *Proceedings of the on Future of Software Engineering - FOSE 2014* (pp. 1–12). New York, New York, USA: ACM Press.

- George, S. Z., Coronado, R. A., Beneciuk, J. M., Valencia, C., Werneke, M. W., & Hart, D. L. (2011). Depressive Symptoms, Anatomical Region, and Clinical Outcomes for Patients Seeking Outpatient Physical Therapy for Musculoskeletal Pain. *Physical Therapy, 91*(3), 358–372.
- Glaser, B. G. (1965). The Constant Comparative Method of Qualitative Analysis. *Source: Social Problems Hospitals American Journal of Nursing American Sociological Review, 12*(4), 436–445.
- Glueck, W. F. (1980). *Strategic Management and Business Policy*. McGraw-Hill.
- Herbsleb, J. D. (2007). Global Software Engineering: The Future of Socio-technical Coordination. In *Proceedings of the Future of Software Engineering (FOSE '07)* (pp. 188–198). Washington DC, USA: IEEE.
- Herbsleb, J. D., & Mockus, A. (2003). An empirical study of speed and communication in globally distributed software development. *IEEE Transactions on Software Engineering, 29*(6), 481–494.
- Herbsleb, J. D., & Moitra, D. (2001). Global software development. *IEEE Software, 18*(2), 16–20.
- Jain, R., & Suman, U. (2015). A Systematic Literature Review on Global Software Development Life Cycle. *ACM SIGSOFT Software Engineering Notes, 40*(2), 1–14.
- Kang, S., Myung, J., Yeon, J., Ha, S., Cho, T., Chung, J., & Lee, S. (2010). A General Maturity Model and Reference Architecture for SaaS Service. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (Vol. 5982 LNCS, pp. 337–346).
- Kircher, M., Jain, P., Levine, D., & Corsaro, A. (2001). Distributed extreme programming. In *Proceedings of the International Conference on eXtreme Programming and Flexible Processes in Software Engineering* (pp. 66–71).
- Klopper, R., Lubbe, S., & Rugbeer, H. (2007). The Matrix Method of literature review. *Alternation, 14*(1), 262–276.
- Klubnikin, A. (2016). How to Choose Software Outsourcing Company? Retrieved November 30, 2016, from <http://r-stylelab.com/company/blog/it-outsourcing/how-to-choose-software-outsourcing-company>
- Knowledge. (2017). Retrieved June 21, 2017, from <https://www.merriam-webster.com/dictionary/knowledge>
- Koc, H., Timm, F., Espana, S., Gonzalez, T., & Sandkuhl, K. (2016). A Method for Context Modelling in Capability Management. *Research Papers, 43*.
- Kotlarsky, J., & Oshri, I. (2005). Social ties, knowledge sharing and successful collaboration in globally distributed system development projects. *European Journal of Information Systems, 14*(December 2004), 37–48.
- Kotlarsky, J., van Fenema, P. C., & Willcocks, L. P. (2008). Developing a knowledge-based perspective on coordination: The case of global software projects. *Information and Management, 45*(2), 96–108.
- Kraut, R. E., & Streeter, L. A. (1995). Coordination in software development. *Communications of the ACM, 38*(3), 69–81.
- Kristjánsson, B., Helms, R., & Brinkkemper, S. (2014). Integration by communication: Knowledge exchange in global outsourcing of product software development. *Expert Systems, 31*(3), 267–281.
- Lamersdorf, A., Munch, J., & Rombach, D. (2009). A Survey on the State of the Practice in Distributed Software Development: Criteria for Task Allocation. In *2009 Fourth IEEE International Conference on Global Software Engineering* (pp. 41–50). IEEE.
- Lamersdorf, A., Münch, J., & Rombach, D. (2009). A Decision Model for Supporting Task Allocation Processes in Global Software Development. In A. Jedlitschka & O. Salo (Eds.), *Software Quality Journal* (Vol. 9, pp. 332–346). Berlin, Heidelberg: Springer Berlin Heidelberg.
- Lanubile, F., Ebert, C., Prikładnicki, R., & Vizcaino, A. (2010). Collaboration Tools for Global Software Engineering. *IEEE Software, 27*(2), 52–55.
- Lee, A. S., Baskerville, R., Lee, A. S., & Baskerville, R. L. (2017). Generalizing Generalizability in Information Systems Research. *Information Systems Research, 14*(3), 221–243.
- Li, Y., & Maedche, A. (2012). Formulating Effective Coordination Strategies in Agile Global Software Development Teams. *Icis-Rp, 1*–12.
- Luinenburg, L., Jansen, S., Souer, J., van de Weerd, I., & Brinkkemper, S. (2008). Designing web content management systems using the method association approach. In *Proceedings of the 4th International Workshop on Model-Driven Web Engineering (MDWE 2008)* (pp. 106–120).
- Mak, D. K. M., & Kruchten, P. B. (2006). Task coordination in an agile distributed software development environment. *Engineering, (May)*, 606–611.

- Malone, T. W., & Crowston, K. (1994). The Interdisciplinary Study of Coordination. *ACM Computing Surveys*, 26(1), 87–119.
- Mantyla, M. V., & Vanhanen, J. (2011). Software Deployment Activities and Challenges - A Case Study of Four Software Product Companies. In *2011 15th European Conference on Software Maintenance and Reengineering* (pp. 131–140). IEEE.
- McCann, J. E., & Ferry, D. L. (1979). An Approach for Assessing and Managing Inter-Unit Interdependence. *The Academy of Management Review*, 4(1), 113.
- McChesney, I. R., & Gallagher, S. (2004). Communication and co-ordination practices in software engineering projects. *Information and Software Technology*, 46(7), 473–489.
- Mintzberg, H. (1979). *The Structuring of Organizations*. Pearson.
- Mintzberg, H. (1980). Structure in 5'S: A synthesis of the research on organization design. *Management Science*, 26(3), 322–341.
- Mishra, D., Mishra, A., Colomo-Palacios, R., & Casado-Lumbreras, C. (2013). Global Software Development and Quality Management: A Systematic Review. In *Lecture Notes in Computer Science* (Vol. 8186 LNCS, pp. 302–311).
- Myers, M. D., & Newman, M. (2007). The qualitative interview in IS research: Examining the craft. *Information and Organization*, 17(1), 2–26.
- Na, K.-S., Simpson, J. T., Li, X., Singh, T., & Kim, K.-Y. (2007). Software development risk and project performance measurement: Evidence in Korea. *Journal of Systems and Software*, 80(4), 596–605.
- Nguyen-Duc, A., & Cruzes, D. S. (2013). Coordination of software development teams across organizational boundary-An exploratory study. In *Proceedings - IEEE 8th International Conference on Global Software Engineering, ICGSE 2013* (pp. 216–225). IEEE.
- Nguyen-Duc, A., Cruzes, D. S., & Conradi, R. (2012). Dispersion, coordination and performance in global software teams. *Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement - ESEM '12*, (Idi), 129.
- Nguyen-Duc, A., Cruzes, D. S., & Conradi, R. (2015). The impact of global dispersion on coordination, team performance and software quality-A systematic literature review. *Information and Software Technology*, 57(1), 277–294.
- Niazi, M., Mahmood, S., Alshayeb, M., Riaz, M. R., Faisal, K., Cerpa, N., ... Richardson, I. (2016). Challenges of project management in global software development: A client-vendor analysis. *Information and Software Technology*, 80, 1–19.
- Nidumolu, S. (1996). A comparison of the structural contingency and risk-based perspectives on coordination in software-development projects. *Journal of Management Information Systems*, 13(2), 77–113.
- Noll, J., Beecham, S., & Richardson, I. (2010). Global software development and collaboration. *ACM Inroads*, 1(3), 66.
- Nonaka, I., & Takeuchi, H. (1995). Knowledge-Creating Company. *Knowledge-Creating Company*, (August), 3–19.
- Nord, R. L., Ozkaya, I., & Kruchten, P. B. (2014). Agile in Distress: Architecture to the Rescue. In *Agile Methods. Large-Scale Development, Refactoring, Testing, and Estimation* (Vol. 199, pp. 43–57).
- Olsson, H. H., Conchúir, E. Ó., Ågerfalk, P. J., & Fitzgerald, B. (2006). Global Software Development Challenges: A Case Study on Temporal, Geographical and Socio-Cultural Distance. In *Proceedings of the IEEE International Conference on Global Software Engineering (ICGSE'06)* (pp. 3–11). Florianopolis, Brazil: IEEE.
- Olsson, H. H., Fitzgerald, B., Ågerfalk, P. J., & Conchúir, E. Ó. (2006). Agile Practices Reduce Distance in Global Software Development. *Information Systems Management*, 23(3), 7–18.
- Paasivaara, M., & Lassenius, C. (2006). Could Global Software Development Benefit from Agile Methods? In *Proceedings of the 2006 IEEE International Conference on Global Software Engineering (ICGSE'06)* (pp. 109–113). Florianopolis, Brazil: IEEE.
- Paasivaara, M., & Lassenius, C. (2016). Scaling scrum in a large globally distributed organization: A case study. In *Proceedings - 11th IEEE International Conference on Global Software Engineering, ICGSE 2016* (pp. 74–83).
- Pardo, C., Pino, F. J., García, F., Piattini, M., & Baldassarre, M. T. (2012). An ontology for the harmonization of multiple standards and models. *Computer Standards & Interfaces*, 34(1), 48–59.
- Piri, A., Niinimäki, T., & Lassenius, C. (2012). Fear and distrust in global software engineering projects.

- Journal of Software: Evolution and Process*, 24(2), 185–205.
- PMI. (2000). *A guide to the project management body of knowledge (PMBOK® guide)* (4th Editio). Pennsylvania, USA: Project Management Institute.
- Polančič, G., Heričko, M., & Rozman, I. (2010). An empirical examination of application frameworks success based on technology acceptance model. *Journal of Systems and Software*, 83(4), 574–584.
- Portillo-Rodríguez, J., Vizcaíno, A., Piattini, M., & Beecham, S. (2012). Tools used in Global Software Engineering: A systematic mapping review. *Information and Software Technology*, 54(7), 663–685.
- Pressman, R. s. (2010). *Software Engineering: A Practitioner's Approach* (7th Editio). New York, USA: McGraw-Hill.
- Pries-Heje, L., & Pries-Heje, J. (2011). Why Scrum Works: A Case Study from an Agile Distributed Project in Denmark and India. In *2011 AGILE Conference* (pp. 20–28). IEEE.
- Process. (2017). Retrieved June 21, 2017, from <https://www.merriam-webster.com/dictionary/process>
- Purna Sudhakar, G., Farooq, A., & Patnaik, S. (2011). Soft factors affecting the performance of software development teams. *Team Performance Management: An International Journal*, 17(3/4), 187–205.
- Ramasubbu, N., Cataldo, M., Balan, R. K., & Herbsleb, J. D. (2011). Configuring global software teams: a multi-company analysis of project productivity, quality, and profits. In *Proceedings of the 33rd International Conference on Software Engineering (ICSE)* (pp. 261–270). Waikiki, Honolulu, Hawaii: ACM Press.
- Reuwer, T., Jansen, S., & Brinkkemper, S. (2013). Key factors in the internationalisation process of SMEs exporting business software as a service. *International Journal of Business Information Systems*, 12(2), 140.
- Richardson, I., Casey, V., Burton, J., & McCaffery, F. (2010). Global Software Engineering: A Software Process Approach. In *Collaborative Software Engineering* (pp. 35–56). Berlin, Heidelberg: Springer Berlin Heidelberg.
- Richardson, I., Casey, V., McCaffery, F., Burton, J., & Beecham, S. (2012). A process framework for global software engineering teams. *Information and Software Technology*, 54(11), 1175–1191.
- Rodríguez, P., Haghightakhah, A., Lwakatare, L. E., Teppola, S., Suomalainen, T., Eskeli, J., ... Oivo, M. (2017). Continuous deployment of software intensive products and services: A systematic mapping study. *Journal of Systems and Software*, 123, 263–291.
- Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14(2), 131–164.
- Rus, I., Lindvall, M., & Sinha, S. S. (2002). Knowledge management in software engineering. *IEEE Software*, 19(3), 26–38.
- Sääksjärvi, M., Lassila, A., & Nordström, H. (2005). Evaluating the software as a service business model: From CPU time-sharing to online innovation sharing. In *IADIS International Conference e-Society* (pp. 177–186).
- Sangwan, R., Bass, M., Mullick, N., Paulish, D., & Kazmeier, J. (2007). Critical Success Factors for Global Software. In *Global Software Development Handbook* (pp. 9–20).
- Sarker, S., & Sahay, S. (2004). Implications of space and time for distributed work: an interpretive study of US–Norwegian systems development teams. *European Journal of Information Systems*, 13(1), 3–20.
- Sarma, A., Van der Hoek, A., & Redmiles, D. (2010). *The Coordination Pyramid: A Perspective on the State of the Art in Coordination Technology*. CSE Technical Reports (Vol. 160).
- Schneider, S., Torkar, R., & Gorschek, T. (2013). Solutions in global software engineering: A systematic literature review. *International Journal of Information Management*, 33(1), 119–132.
- Schwaber, K. (1997). SCRUM Development Process. In *Business Object Design and Implementation* (pp. 117–134). London: Springer London.
- Schwaber, K. (2004). *Agile Project Management with Scrum*. Washington, USA: Microsoft Press.
- Setamanit, S.-O., Wakeland, W., & Raffo, D. (2006). Planning and improving global software development process using simulation. In *Proceedings of the 2006 international workshop on Global software development for the practitioner - GSD '06* (pp. 8–14). New York, New York, USA: ACM Press.
- Signell, R. P., Carniel, S., Chiggiato, J., Janekovic, I., Pullen, J., & Sherwood, C. R. (2008). Collaboration tools and techniques for large model datasets. *Journal of Marine Systems*, 69(1–2), 154–161.
- Smirnova, I., Münch, J., & Stupperich, M. (2014). A Canvas for Establishing Global Software Development Collaborations. In *Communications in Computer and Information Science* (Vol. 465, pp. 73–93).
- Šmite, D. (2007). *Global Software Development Improvement. Doctoral Thesis for Ph.D. Academic Degree*.

- University of Latvia.
- Smith, E. A. (2001). The role of tacit and explicit knowledge in the workplace. *Journal of Knowledge Management*, 5(4), 311–321.
- Software Engineering Institute. (2010). *CMMI<sup>®</sup> for Development, Version 1.3 Improving processes for developing better products and services*. Pittsburgh.
- Sommerville, I. (2010). *Software Engineering*. (M. Horton, Ed.), *Software Engineering* (9th ed.). Boston: Addison-Wesley.
- Strode, D. E. (2013). Extending the dependency taxonomy of agile software development. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (Vol. 8224, pp. 274–289).
- Strode, D. E. (2016). A dependency taxonomy for agile software development projects. *Information Systems Frontiers*, 18(1), 23–46.
- Strode, D. E., Hope, B., Huff, S. L., & Link, S. (2011). Coordination Effectiveness In An Agile Software Development Context. In *Proceedings of the PACIS 2011* (pp. 1–16). Brisbane, Australia: Queensland University of Technology.
- Strode, D. E., Huff, S. L., Hope, B., & Link, S. (2012). Coordination in co-located agile software development projects. *Journal of Systems and Software*, 85(6), 1222–1238.
- Sudhakar, G. P. (2013). A Review of Critical Success Factors for Offshore Software Development Projects. *Organizacija*, 46(6), 282–296.
- Talby, D., & Dubinsky, Y. (2009). Governance of an agile software project. In *2009 ICSE Workshop on Software Development Governance* (pp. 40–45). IEEE.
- Vähäniitty, J. (2006). *Do small software companies need portfolio management?* Helsinki University of Technology.
- Van De Ven, A. H., Delbecq, A. L., & Koenig Jr., R. (1976). Determinants of Coordination Modes within Organizations. *American Sociological Review*, 41(2), 322–338.
- van de Weerd, I., & Brinkkemper, S. (2009). Meta-Modeling for Situational Analysis and Design Methods. *Handbook of Research on Modern Systems Analysis and Design Technologies and Applications*, 38–58.
- van de Weerd, I., Brinkkemper, S., Nieuwenhuis, R., Versendaal, J., & Bijlsma, L. (2006). On the Creation of a Reference Framework for Software Product Management: Validation and Tool Support. In *Proceedings of the 2006 International Workshop on Software Product Management (IWSPM'06 - RE'06 Workshop)* (pp. 3–12). Minnesota, USA: IEEE.
- van de Weerd, I., Brinkkemper, S., Souer, J., & Versendaal, J. (2006). A situational implementation method for web-based content management system-applications: method engineering and validation in practice. *Software Process Improvement and Practice*, 11(5), 521–538.
- van de Weerd, I., Brinkkemper, S., & Versendaal, J. (2010). Incremental method evolution in global software product management: A retrospective case study. *Information and Software Technology*, 52(7), 720–732.
- Van Gameren, B., Van Solingen, R., & Dullemond, K. (2013). Auto-erecting virtual office walls a controlled experiment. In *Proceedings - IEEE 8th International Conference on Global Software Engineering, ICGSE 2013* (pp. 206–215).
- Venable, J., Pries-Heje, J., & Baskerville, R. (2016). FEDS: a Framework for Evaluation in Design Science Research. *European Journal of Information Systems*, 25(1), 77–89.
- Verner, J. M., Brereton, O. P., Kitchenham, B. A., Turner, M., & Niazi, M. (2014). Risks and risk mitigation in global software development: A tertiary study. *Information and Software Technology*, 56(1), 54–78.
- Victor, B., & Blackburn, R. S. (1987). Interdependence: An Alternative Conceptualization. *Academy of Management Review*, 12(3), 486–498.
- Wagenaar, G., Overbeek, S., & Helms, R. (2017). Describing Criteria for Selecting a Scrum Tool Using the Technology Acceptance Model. In *Intelligent Information and Database Systems* (pp. 811–821).
- Webster, J., & Watson, R. T. (2002). Analyzing the past to prepare for the future writing a literature review. *MIS Quarterly*, 26(2), xiii–xxiii.
- Wen, L. (2016). Crossing the Communication Barrier in Global Software Development Projects via Global Software Development Brokers. In *Proceedings of the 24th Australasian Conference on Information Systems* (pp. 1–11).
- Wieringa, R. J. (2014). *Design science methodology: For information systems and software engineering. Design Science Methodology: For Information Systems and Software Engineering*. Berlin, Heidelberg:

Springer Berlin Heidelberg.

- Xu, L., & Brinkkemper, S. (2007). Concepts of product software. *European Journal of Information Systems*, 16(5), 531–541.
- Yilmaz, M., O'Connor, R. V., & Clarke, P. (2014). An Exploration of Individual Personality Types in Software Development. In *Communications in Computer and Information Science* (Vol. 425, pp. 111–122).
- Yin, R. K. (2013). Case Study Research: Design and Methods. *Applied Social Research Methods Series*, 5, 1–53.

# APPENDICES

## Appendix A. Interview Protocol

### **Methodological Support for Task Coordination on Global Software Engineering Project in a Product Software Organization**

**Interview Protocol  
Department of Information and Computing Science**



**Universiteit Utrecht**

**Utrecht, The Netherlands**

Interviewee : \_\_\_\_\_  
Date & Time : \_\_\_\_\_  
Interviewers : Carolus Borromeus Widiyatmoko  
Research Supervisor : dr. Sietse J. Overbeek  
Prof. dr. Sjaak Brinkempper

First of all I want to thank you for your cooperation and taking the time to conduct this interview. The purpose of this interview is to gather information on the current practice of task coordination in distributed software engineering project in your organization.

In the following 45 to 60 minutes we will run through this list in the form of an interview. If during the interview you ever feel uncomfortable or if you for any reason may wish not to answer, you are ever free to do so. This interview will be recorded, will only be used for this research, and will never be disclosed to third parties.

We would like to start by understanding your role in brief, and will thereafter be focusing on two topics:

- Your organization, which is including your position, organization mission, and product software produced by your organization
- Coordination approaches, which are including the methods, tools that are used as well as benefits, problems, criteria become the concerns of your organization

The detailed questionnaire is on the next page.

Thank you.



## A. Product Software Organization

### ***Organization Structure, Stakeholders, and Policy***

Duration: 5-10 minutes

We would like to explore the history and the policy of global software development projects in this company.

*Note: it might be that our respondent cannot answer these questions below. In that case, ask him who would be the right person to talk about these matters.*

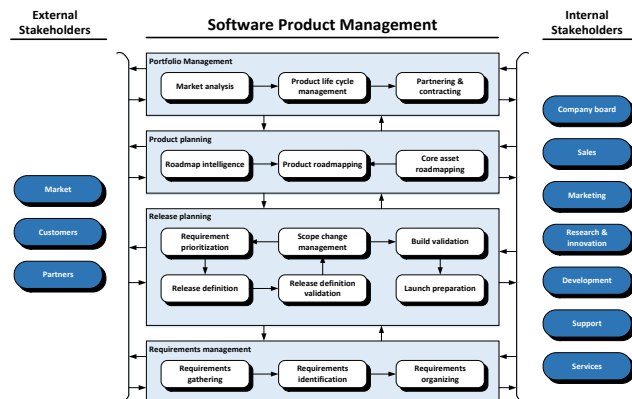
1. What is your role in this company in general?
2. How long has this company performed distributed approach in software development projects?
3. Which role in this company who is having the most responsibility for the success of global software development projects? Is there a policy associated to the global business of this company?
4. Remote sites.
  - a. What are the types of your remote sites? Did you acquire startups or other companies? Alternatively, did you develop remote sites and recruit or move your employees to the new sites? What are the considerations of (acquiring other companies / developing remote sites) instead of the other options? (Nguyen-Duc et al., 2012, 2015)
  - b. What are the functions of the remote sites?
5. What are the metrics that you use to measure your project performance? (Na, Simpson, Li, Singh, & Kim, 2007)
6. What are the goals that your company wants to achieve by distributing the projects to dispersed resources? Or in other words, what are the benefits for this company? And what are the benefits for your resources?

### ***Product Software***

Duration: 10-15 minutes

Xu and Brinkkemper (2007) define product software as “a packaged configurations of software components or a software-based service, with auxiliary materials, which is released for and traded in a specific market”. Hence, a product software organization can be described as companies which develop and sell mainly software as their products for a target market without customer specific modifications (Vähäniitty, 2006).

7. What are the product software that your company has built?
8. Do you have a roadmap or product line management for each of your software?
9. Check this framework.



Which parts of this framework which are affected by global software engineering?

## B. Task Coordination

### ***Task Distribution***

Duration: 5-10 minutes

The following questions are exploring the decision in allocating tasks to dispersed resources. This section is used to get the big picture of task management in global software development projects.

10. Who does decide the task management (breaking down requirement into tasks, allocating tasks, and monitoring the progress of each task)? Moreover, to whom he/she reports the project progress?
11. Could you please elaborate the stages in breaking and allocating the tasks in a brief?
12. What are the (main) factors considered by this company when a project manager (or another role mentioned in B.1) in choosing what tasks to which resources? Hint: specific capability (e.g. specialized in UI, API/middleware, etc.), volume/number of resources, level of knowledge (e.g. ability to understand the architectural design)

### ***Task Coordination***

Duration: 20-25 minutes

Based on the study conducted by (Sangwan, Bass, Mullick, Paulish, & Kazmeier, 2007), there are three main issues in global software engineering which are communication, coordination, and control. Another study by (Jain & Suman, 2015) added knowledge aspect in understanding system design as another issue. Those are the examples of issues in global software engineering based on the literatures. The following questions are used to help us in developing a taxonomy that can describe the challenges in global software projects and a model that describe the processes from current practices performed by companies.

13. Could you please define “coordination” based on your experiences? And, what are the important things that should be considered in coordinating tasks in regards to performing software engineering globally?
14. What is the goal of coordinating tasks?
15. Do you prefer to manage the tasks with direct supervision or mutual adjustment?
16. How do you know the progress of a task or if there is a problem occurs in a task?
17. Do you have an established approach in coordinating tasks between (dispersed) teams?
  - a. If the answer is:
    - i. NO: Can you describe it in your own words what are the steps? Do you have any problems in formulizing your processes into an established method?
    - ii. YES: Can you elaborate the steps? (e.g. Do you start with project progress checking (budget & costs, deliverables), continued by problems checking or performance optimization)
  - b. What are the tools that you use? And who are or what are the roles involved in the processes?
18. There are numerous aspects related to task coordination.
  - a. Do you usually use direct supervision to manage the coordination? Or do you prefer to let them arrange and decide the communication by themselves?
  - b. Control (McChesney & Gallagher, 2004; Portillo-Rodríguez, Vizcaíno, Piattini, & Beecham, 2012)
    - i. Did you ever have a problem in project controlling?
    - ii. Related to the processes that you have mentioned before, who is responsible for the project control (Is it the same roles with task distribution?) and how does he/she control the development chain and monitor the progress?
    - iii. How does he/she know when there is a problem occurs in a task?
    - iv. What are the problems in controlling the dependencies?
  - c. Communication
    - i. Do you have a problem in internal communication between sites?
    - ii. How do you manage the communication between teams (HO to remote sites, between remote sites)? What are the problems of the communication? (Portillo-Rodríguez et al., 2012)

- iii. How frequent do you manage the communication between the resources? (Mak & Kruchten, 2006)
- iv. Do you use informal or formal communication with your remote sites? And why? (Mak & Kruchten, 2006)
- v. Have you ever found any fear and distrust among resources? How do you manage that situation? (Piri, Niinimäki, & Lassenius, 2012)
- d. Stakeholders  
Who are the stakeholders involved in coordinating processes? Are there specific persons or roles who are very important in coordinating tasks (e.g. knowledge brokers, communication, on-site coordinator, cross-site delegators)? (Deshpande et al., 2011)
- e. Dependency  
What types of cross-sites dependency in your company (e.g. process dependencies, knowledge and expertise dependencies)? (Deshpande et al., 2011)  
How do you manage the inter dependency?
- f. Knowledge  
Does knowledge (differences) become important to your company in relation with your business in coordinating tasks?  
Is it a barrier or an advantage for your company?  
How do you manage the imbalance in knowledge and expertise? (Kotlarsky et al., 2008; Purna Sudhakar, Farooq, & Patnaik, 2011)
- g. Tool  
Do you use tools to help you to coordinate tasks among sites?  
What are the functions of the tools and for what reasons? (Portillo-Rodríguez et al., 2012)
- h. Performance - Project Monitoring and Controlling (continuing C.2.b)  
After all, how do you measure the progress and keep the all teams work to achieve the best performance?
- i. Do you have any other things that should be considered?

## Appendix B. Systematic Literature Review

Table 9-1. Selected papers

Authors	Type of Research	Number of Citations	Summary
Bekkers, Spruit, et al., (2010)	Design science	46	Their research presents a competence model and a maturity matrix for software product management. The model is aimed to be used by product software companies as a solid basis for product software process improvement. This paper is selected to bring an understanding of the characteristics that should be performed by a product software.
Bosch and Bosch-Sijtsema (2010a)	Case study in 3 projects in a company	13	In their research, they studied the relation between large-scale and agile approaches to global software development projects. They present “ architecture-centric software engineering” as an integration of best practices at the case study companies. Their approach attempts to remove inter-team dependencies to bring more efficiency and productivity in global software development projects. This paper introduces dependency as an important concept in task coordination. In this case, dependency is related to the processes and costs of communication, integration, and interaction.
Noll, Beecham, and Richardson (2010)	Systematic literature reviews of 26 papers	140	They identified eight categories of barriers: geographic distance, temporal distance, linguistic and cultural distance, fear and trust, problems stemming from organizational structure, process issues, barriers deriving from infrastructure, and barriers due to product architecture. Moreover, to addressing these obstacles, seven categories of solutions emerged: approaches to address language and cultural differences; techniques for promoting trust and overcoming fear; communication infrastructure; management interventions; organizational structures; and distributed development processes. Since distance issue becomes the top barrier, solutions attempt to overcome this by providing more in person communication experiences (e.g. online face-to-face meetings); by adapting processes and organizational structure to address delays; and by providing infrastructure and processes to promote knowledge sharing in a co-located setting. This paper becomes a gate to what have been studied that brings a broad knowledge about the challenges and solutions to cope the barriers in global software projects.

<b>Authors</b>	<b>Type of Research</b>	<b>Number of Citations</b>	<b>Summary</b>
Anh, Cruzes, and Conradi (2012)	Empirical study of 28 papers	16	This article presents more comprehensive knowledge on how resources dispersion effects to the coordination mechanism and its impact on the performance in global software projects. This study identified five common dispersion dimensions: geographical, temporal, cultural, work process and organizational dispersion. They found that these dimensions could bring impacts to the team performance indirectly by affecting the team communications. Unfortunately, the article does not detail on the communication levels and aspects affected by the dispersion dimensions. It brings an opportunity for further exploration in our research.
Richardson, Casey, McCaffery, Burton, and Beecham (2012)	Case studies in a global software engineering projects in three companies	65	They found that several companies are struggling with the successful implementation of global software engineering. They propose Global Teaming as a software process which includes specific practices and sub-practices. The goal of their approach is to improve the quality of software product by implementing efficient software processes. This paper brings the example of how global software engineering is applied in product software development processes.
Mishra, Mishra, Colomo-Palacios, and Casado-Lumbreras (2013)	Systematic literature reviews of 144 articles	2	This paper brings the bridge to the previous researches on how to manage quality in global software engineering projects. The paper arranges the studies into three main concepts: Quality Assurance, Process, and Verification and Validation.

Table 9-2 Task Coordination Concept Matrix

Articles	Summary	Control	Communicatio	Stakeholders	Dependency	Knowledge	Tool	Performance
Š mite (2007)	In her thesis, Š mite mentions that building team cohesion by focusing on project tailoring does not only become managers' concern. It should become a consideration of all roles or stakeholders of the projects.		+	++				
Deshpande et al. (2011)	Their study addresses coordination strategy from the vendor' s perspective. They revealed that process interdependency is a critical factor in coordination tasks, especially in large-scale development projects. Several roles such as onsite coordinator, cross-site delegates and liaisons are needed to improve cross-site communication.		+	++	+++			
Ramasubbu, Cataldo, Balan, and Herbsleb (2011)	The paper explains that variations in characteristics of dispersed teams lead to different project performance outcomes. A project performance is measured by considering its development productivity, process quality, and profits. The study revealed that by distributing their development across longer distance, companies improve their productivity but also decrease their quality significantly. They also notice productivity and quality contribute to higher profit positively.		++					+++
Portillo-Rodríguez, Vizcaí no, Piattini, and Beecham (2012)	This paper a systematic mapping review which aimed to discover the available tool involved in highly distributed teams that can support communication, coordination, and control. Most of the tools are used for communication, project management, and knowledge sharing.	++	++			+	+++	+
Sudhakar (2013)	In his research, Sudhakar revealed six CFSs in offshore software development projects: trust, efficient communication, cultural understanding, relationship between client and vendor, contract type and efficient knowledge transfer.		+++		+++	+++		
Smirnova, Mü nch, and Stupperich (2014)	Global Canvas describe the activity roadmaps that should be set up by companies when establishing global software development collaborations. The goal of model is proposed to help companies to manage and control projects to complete software projects successfully.	+++			++			+
Legends:								
+++ : The concept is the focus of the paper								
++ : The concept is introduced and well elaborated.								
+ : The concept is introduced but not explained in more detail								
Blank : The concept is not introduced								

## **Appendix C. Company Profiles**

### **C.1 AlphaSoft**

AlphaSoft is a product software company which offers Software Solutions, IT Outsourcing, BPO and Staffing Services. AlphaSoft was started in 1992. Now, the company network is spread in the Netherlands, Belgium, Germany, Switzerland, Norway, Sweden and Romania, where Dutch companies in the Netherlands are their target market.

AlphaSoft has an ambition to grow internationally. They start to recruit professionals from other countries and developing a remote office to support the development process in Romania. Their flagship product is ERP software, which focuses on specific functions such as building management system, electronic banking and point of sales, HR software, and CRM system. The product is prepared for industry-specific market such as construction, financial sector, government, supply chain, education, retail, and healthcare.

### **C.2 BetaSoft**

Our interviews at BetaSoft were performed with the Technology Team in Business Solution organization. BetaSoft itself is a product software company that consists of three main organizations: Cloud Solution, Business Solution, and Specialized Solution. At Business Solution, there are five major disciplines: Marketing, Sales, Product Marketing, Customer Service, and Technology where each discipline is led by a director. The Technology Team consists of some roles: architect, UX designer, product management, and development. The development where spread in several countries. The development of global product is mainly done in Kuala Lumpur (KL) and for the localization solution are performed in some countries such as Spain, Belgium, and the Netherlands.

The development office in Kuala Lumpur was started in 1999. There were several reasons behind the development of the remote office:

1. Difficulties in finding enough resources in the Netherlands
2. Lower salary for human resources
3. As a big city, Kuala Lumpur provides Location was selected where KL is a big city, and a lot of people with technical education, combination of the facility, and English is good compared to some other countries
4. They (government) facilitate the college with multimedia (Multimedia University)
5. KL bring people from other countries

### **C.3 GammaSoft**

GammaSoft is a holding company that acquires many software companies from different countries to spread its network. Each subsidiary is acting as GammaSoft representative office as well as local development products. As a holding company, GammaSoft also develop ERP solution software which is distributed and operated globally. The ERP solution has several core capabilities in financial management, HR management, procurement, and asset management. The software is aimed to be distributed all over the world, not only for the Netherlands market, and not limited to specific types of industries.

The ERP solution is built by the help of two main partners. The first partner in Poland is a consultant that help GammaSoft for the product planning, requirements engineering, and parts of product release management. The other partner in India is focusing only on the

software production. Meanwhile, GammaSoft itself is working on the product portfolio management and sales activities.

Two interviews were performed in different scope of organizations. First interview was done with the Service Delivery Manager from Business Application department, a unit that support internal system for Unit4. This business unit has a partnership with the development partner for the system development, implementation, and operation of IT system in GammaSoft office in the Netherlands. The second interview was conducted with the Global Lead Data Architect who represents GammaSoft as a holding organization.

### **C.4 DeltaSoft**

DeltaSoft has been running as a product software companies for 26 years. Currently, it has two sales offices in Belgium and Germany. DeltaSoft is targeting mainly customers in the Netherlands and those two countries, but are not limited to local companies but also international companies. Currently, they have more than 1,500 customers. They built an ERP product that can be configured specific businesses such as fleet management, facility management, and asset management.

To improve its product because of the needs of new technology adoption and greater scale of development effort, DeltaSoft builds a partnership with a software company in Romania. This partner provides additional human resources to work as part of DeltaSoft's engineers remotely.

### **C.5 ZetaSoft**

In contrast with previous companies, ZetaSoft does not perform software product engineering globally. ZetaSoft provides single ERP software package for various business purposes such as financial, logistics, and HR management. At this time, ZetaSoft is preparing a new product that serves as a platform functioning as a software generator. It is expected to make developers can generate a software through configuration without having to build the software from scratch. The new platform is also designed to be accessible through cloud that enables ZetaSoft to create new business models. ZetaSoft's target market is companies in the Netherlands.



**Appendix D. Appendix I Coding scheme**

The following table presents the codes that refers to the interviews performed with experts for the problem investigation and method validation. To keep the confidentiality, we replace the interviewee's names with their job position.

<b>Code</b>	<b>Purpose</b>	<b>APA Reference</b>
iv-a-1	Problem investigation interview at AlphaSoft	(Scrum Master, personal communication, January 10, 2017)
iv-a-2	Problem investigation interview at AlphaSoft	(Unit Manager, personal communication, January 10, 2017)
iv-z-1	Problem investigation interview at ZetaSoft	(Platform Manager, personal communication, January 27, 2017)
iv-b-1	Problem investigation interview at BetaSoft	(Technology Director, personal communication, February 3, 2017)
iv-b-2	Problem investigation interview at BetaSoft	(Product Manager, personal communication, February 3, 2017)
iv-d-1	Problem investigation interview at DeltaSoft	(Service Delivery Manager, personal communication, February 7, 2017)
iv-d-2	Problem investigation interview at DeltaSoft	(Global Data Architect, personal communication, February 3, 2017)
iv-c-1	Problem investigation interview at GammaSoft	(R&D Manager, personal communication, February 22, 2017)
iv-c-2	Problem investigation interview at GammaSoft	(Team Leader, personal communication, February 3, 2017)
va-p1-1	Validation interview	(Researcher, personal communication, May 11, 2017)
va-p2-1	Validation interview at AlphaSoft	(Scrum Master, personal communication, May 16, 2017)
va-p3-1	Validation interview at BetaSoft with the Technology Director	(Technology Director, group discussion, June 8, 2017)
va-p3-2	Validation interview at BetaSoft with the Product Manager	(Product Manager, group discussion, June 8, 2017),
va-p4-1	Validation interview	(Student Assistant, personal communication, June 13, 2017)
va-p5-1	Validation interview at DeltaSoft	(Service Delivery Manager, personal communication, June 16, 2017)

**Appendix E. Method Association**

S1..S5 : Activity group (Section 5.2.2)

L1..L7 : Literature (Section 3.3)

CA..CD : Companies (Section 4.2)

*Table 9-3 Association Matrix for the Activities*

Group	ACTIVITIES	L1	L2	L3	L4	L5	L6	L7	CA	CB	CC	CD
S1	Product management planning		x							x		
	Reward achievement			x								
	Determine task allocation					x						
	Set goals						x		x	x	x	x
	Determine task schedule						x					
	Define collaboration strategy						x					
	Establish process collaboration						x					
	Identify organization V&M									x		
	Establish SE method						x					
S2	Identify socio-cultural profiles					x	x		x	x		x
	Identify required knowledge					x						
	Identify challenges					x			x	x	x	x
	Identify situational factors									x	x	
S3	Determine organization structure			x		x						
	Role assignment			x		x				x	x	
	Assign onsite coordinator						x					
	Identify supporting tools					x	x					
S4	Relocating team	x										
	Optimize asynchronous tool	x										
	Establish communication norm	x					x		x	x		
	Conduct site visit	x							x	x		x
	Facilitate meeting		x				x		x	x		
	Optimize direct communication		x						x	x	x	x
	Manage impediments		x	x		x			x	x	x	x
	Perform team pairing		x	x							x	
	Encourage collaboration		x									x
	Determine task schedule			x		x						
	Manage knowledge internalization			x								
	Conduct team building			x						x		
	Conduct cognitive assessment			x								
	Assign onsite coordinator			x				x	x	x	x	
	Determine task allocation			x		x			x	x	x	x
Manage knowledge externalization				x	x	x		x	x	x		

References and Appendices

Group	ACTIVITIES	L1	L2	L3	L4	L5	L6	L7	CA	CB	CC	CD
S4	Build social capital				x				x	x		x
	Form virtual teaming					x			x			x
	Identify required knowledge					x						
	Determine coordination mechanism					x	x					
	Establish standardization in reporting					x						
	Apply communication norm					x						
	Establish work product collaboration						x					
	Establish process collaboration						x		x	x	x	x
	Distribute task schedule						x					
	Determine communication norm									x	x	
	Define work product quality standardization									x	x	
	Standardize process									x	x	x
S5	Organize process improvement						x			x	x	
	Continuous improvement									x		
<b>CONCEPTS</b>												
		L1	L2	L3	L4	L5	L6	L7	CA	CB	CC	CD
ROLE				x						x		
CHALLENGE	x											
FORMAL TRAINING										x		
KNOWLEDGE BASE					x					x	x	
INDIRECT COMMUNICATION								x				
COMMUNICATION PROTOCOL	x											
SE STRATEGY										x		
WORK PRODUCT									x	x	x	x
CORPORATE STRATEGY										x		
TEAM				x		x	x					x
INTERNALIZATION					x							
SITE VISIT									x	x		
EXTERNALIZATION					x							
DIRECT COMMUNICATION	x											
COGNITIVE ASPECT					x							
TEAM PAIRING											x	x
VIRTUAL TEAMING						x						
TOOL												x
COORDINATION MECHANISM									x	x	x	x
COMMUNICATION BROKER	x							x			x	
PRODUCT MANAGER			x						x	x		
ASYNCHRONOUS COMMUNICATION	x								x	x	x	x
PRODUCT OWNER			x						x			
IMPEDIMENT			x						x			
FEATURE OWNER			x							x		
KNOWLEDGE					x					x	x	

## References and Appendices

---

CONCEPTS	L1	L2	L3	L4	L5	L6	L7	CA	CB	CC	CD
DEVELOPMENT MANAGER, LINE MANAGER								x	x	x	x
(VIRTUAL) MEETING								x	x	x	x
SCRUM MASTER, FACILITATOR		x						x			
SOCIO-CULTURAL	x							x	x		
MANAGEMENT BOARD									x		
ORGANIZATIONAL SUPPORT			x								
TECHNOLOGY BOARD									x		

Appendix F. Method Base

F.1 PDD of Task Coordination Method v.4

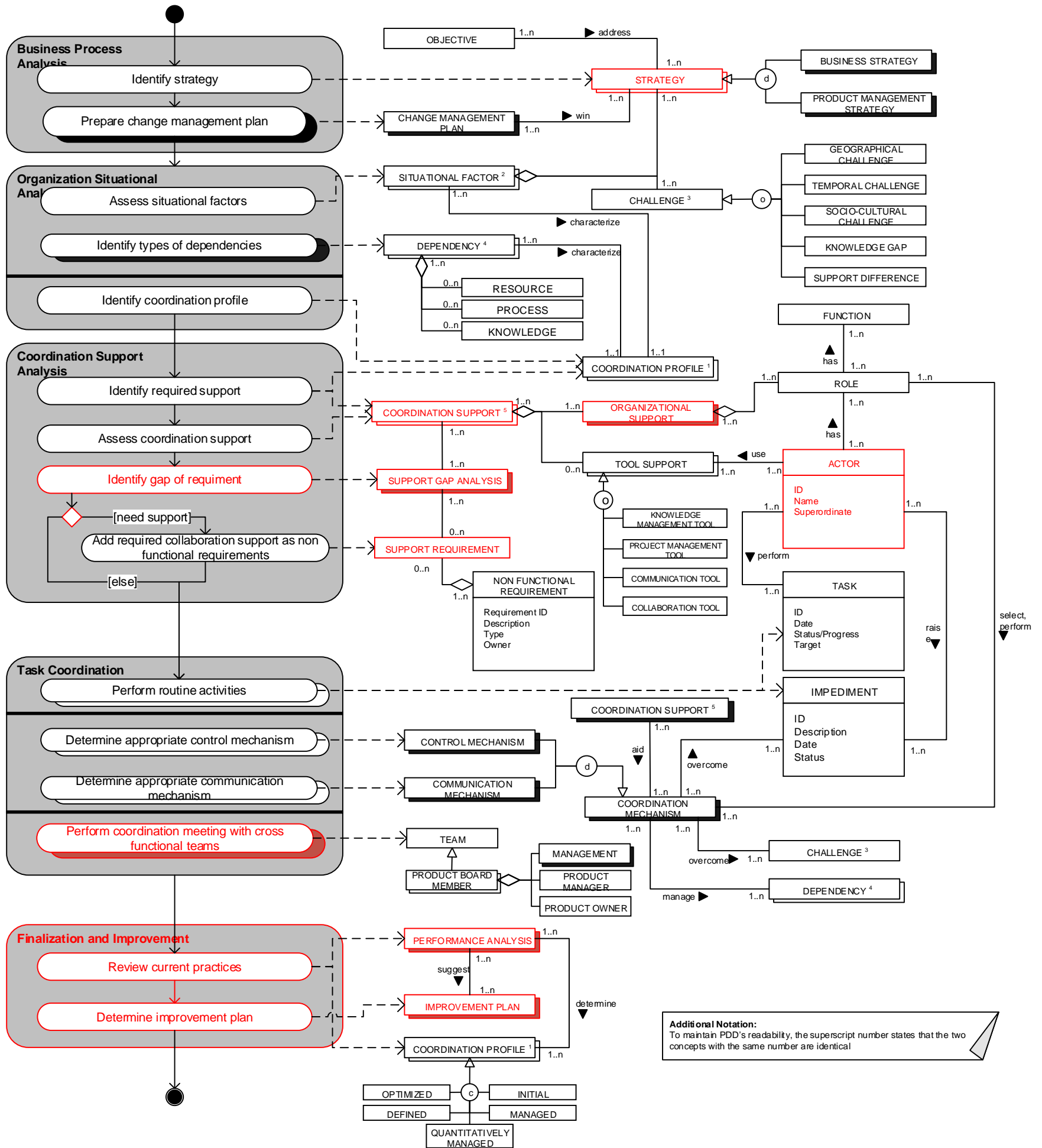


Figure 9-1 PDD of the high level GTC Task Coordination Method

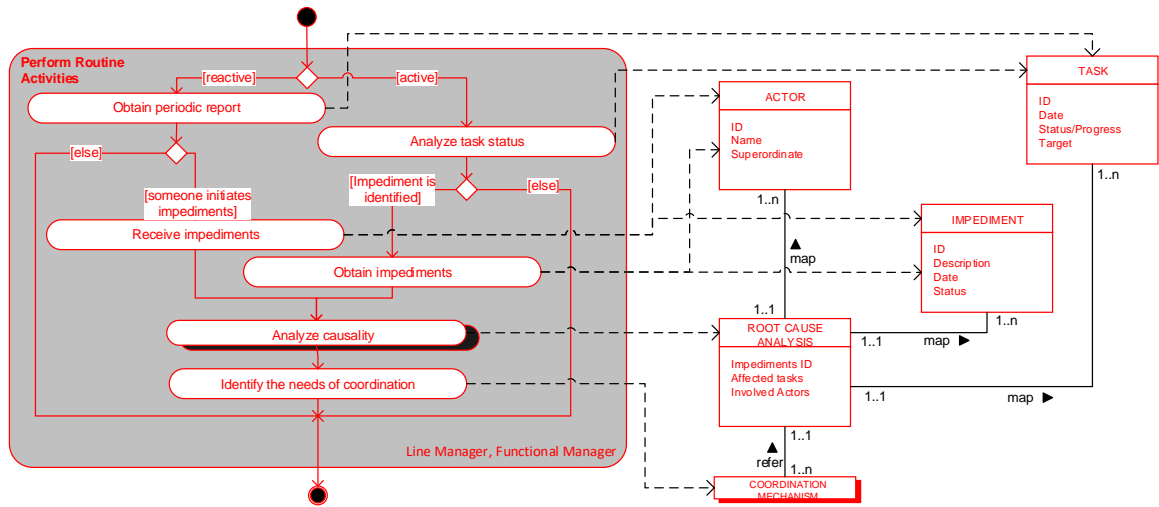


Figure 9-2 PDD of "Perform Routine Activities"

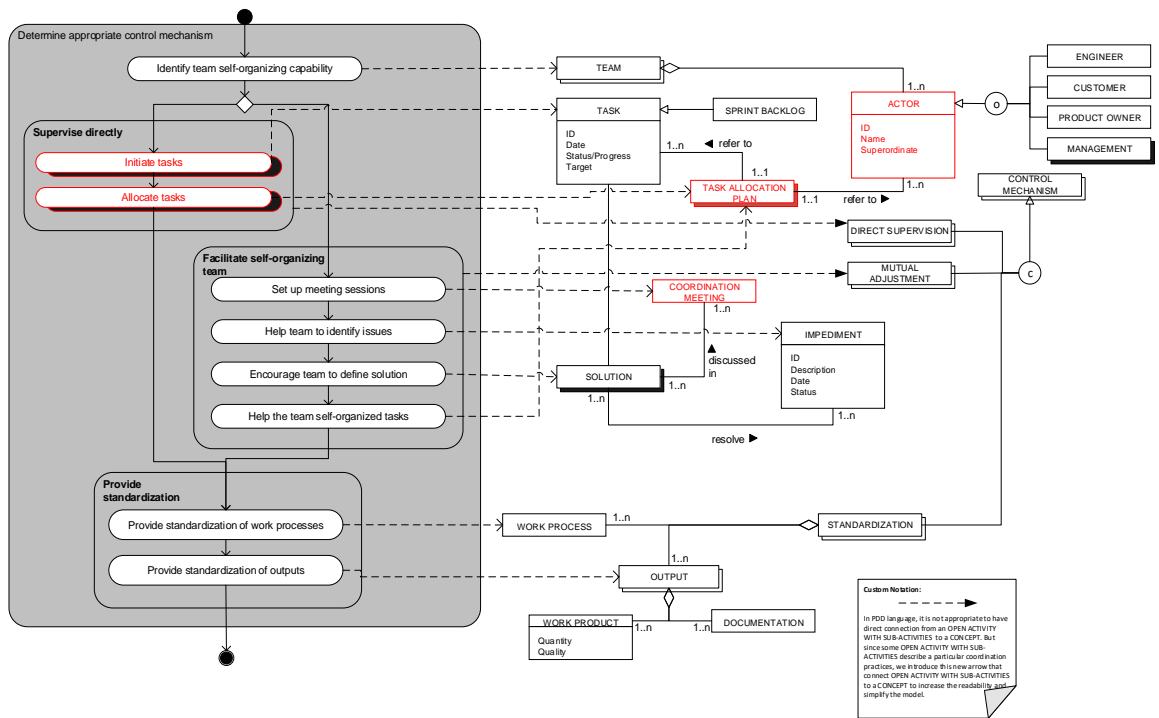


Figure 9-3 PDD of "Determine Control Mechanism"

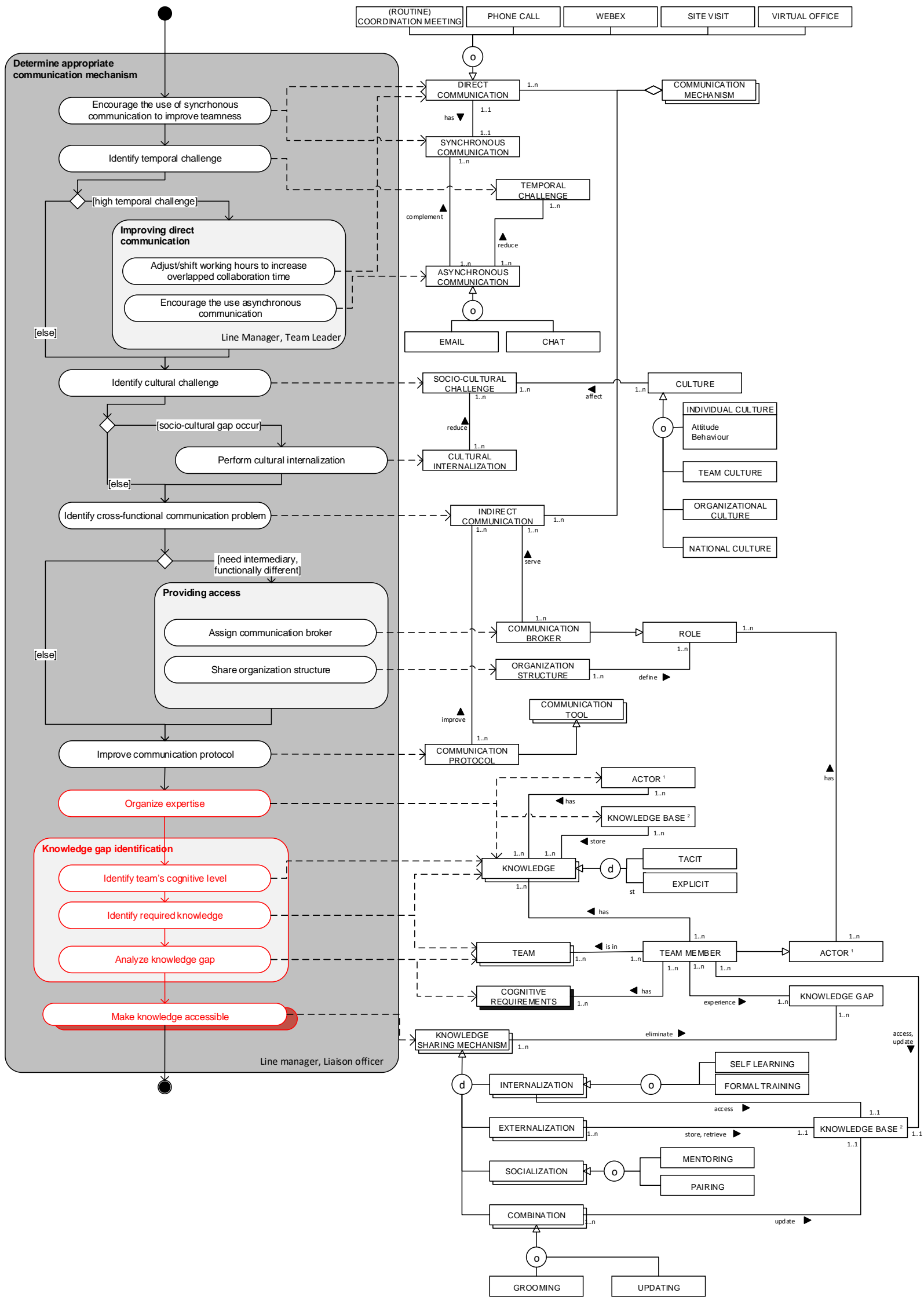


Figure 9-4 PDD of "Determine communication mechanism"

F.2 PDD of Task Coordination Method v.5 (Final Version)

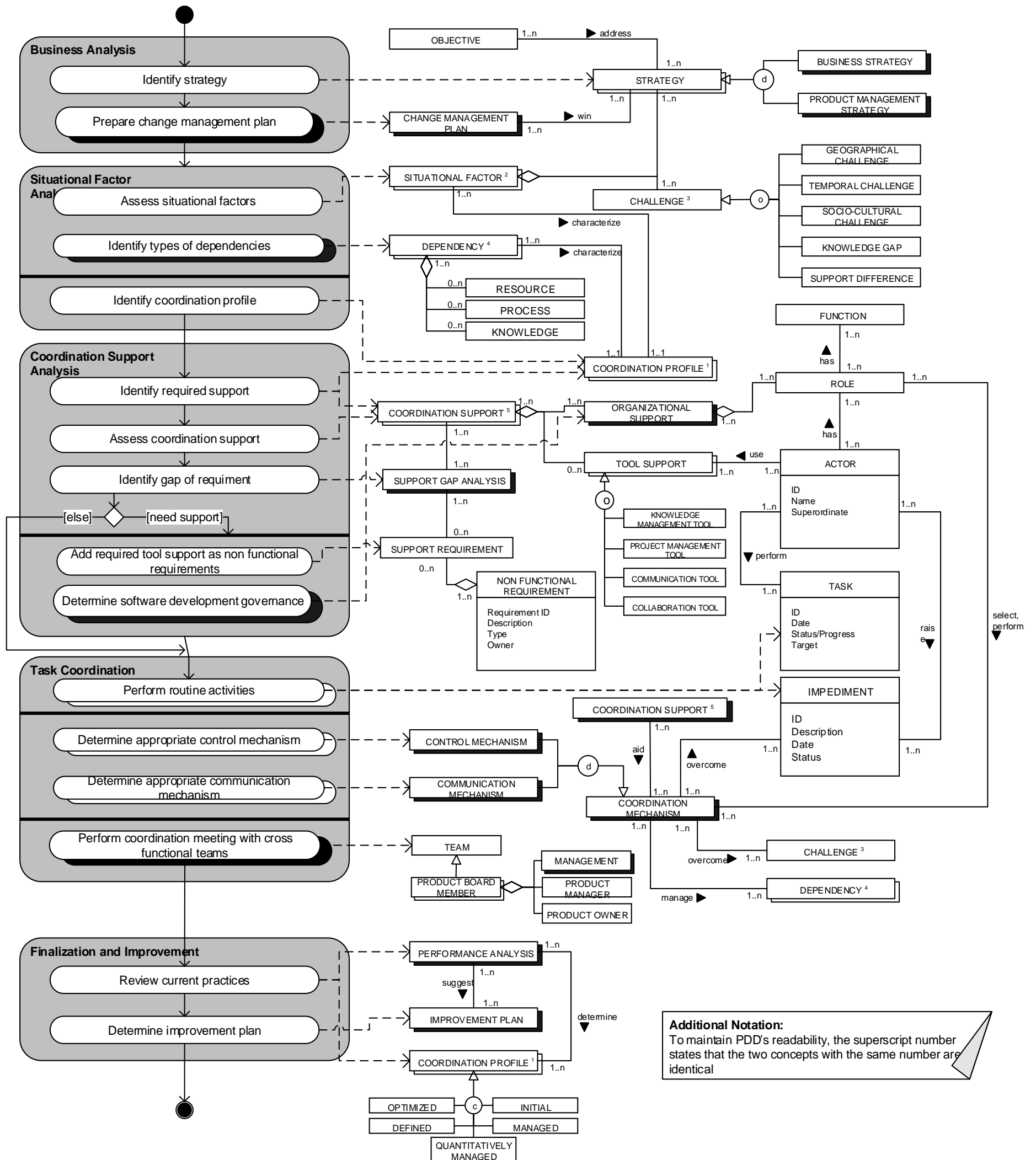


Figure 9-5 PDD of GSE task coordination Method (Main Method's Final Version)



## Appendix G. PDD Documentation

Table F-1 Table of Processes

Activity	Sub Activity	Description
Business Analysis	Identify strategy	The organization identifies BUSINESS STRATEGY, the business OBJECTIVE, and PRODUCT MANAGEMENT STRATEGY that affects to the practices of task coordination.
	Prepare change management plan	The organization prepare the approaches to support the employees for the organizational changes.
Situational Factor Analysis	Assess situational factors	The organization needs to recognize its l factors that could affect the practices in coordinating team members to synchronize their TASKs and WORK PRODUCTS.
	Identify types of dependencies	Coordinating tasks means managing inter-DEPENDENCYs among the TASKs. Thus, the organization must know how the TASKs and the WORK PRODUCTS are dependence each other and what kind of resources that are needed by other teams.
	Identify coordination profile	The organization should determine what level of its fluency in coordinating tasks (Table 6-3). The profiles are used to help organization in continuously measuring the performance in managing tasks among distributed members and improving their practices.
Coordination Support Analysis	Identify required support	The organization should determine its ORGANIZATIONAL SUPPORT (such as organizational structure, governance, or actors) and TOOL SUPPORT that can help managers, facilitators, or team members to perform task coordination.
	Assess coordination support	The organization evaluate the required COORDINATION SUPPORTs to perform task coordination
	Identify gap of requirement	The organization performs a comparison between the requirement and the availability of the supports to provide SUPPORT GAP ANALYSIS.
	Add required tool support as non-functional requirements	Deriving the SUPPORT REQUIREMENT from SUPPORT GAP ANALYSIS, when an organization recognizes missing TOOL SUPPORT, the missing supports should be added to non-functional requirements list (Schwaber, 2004).
	Determine software development governance	When the lack of support is related to the ORGANIZATIONAL SUPPORTs such as no clear authorities, functions, roles, the organization should declare the requirements in its software development governance.
<b>Activity</b>	<b>Sub Activity</b>	<b>Description</b>

Task Coordination	Perform routine activities	The routine activities are used to monitor the current project status, and processing occurred impediments. Some organizations choose to be more proactively acquiring the information before any impediment occurs by conducting a routine meeting (e.g. daily scrum meeting) or site visit.
	Determine appropriate control mechanism	The organization should identify the appropriate mechanisms that suit with the SITUATIONAL FACTORS. The chosen mechanisms can be one of the practices or a combination of them.
	Determine appropriate communication mechanism	The organization identifies the appropriate practices for (especially, not limited only to) horizontal interaction among distributed team members. Organization also identifies the appropriate practices to facilitate the KNOWLEDGE flows and to balance the KNOWLEDGE and expertise among TEAM MEMBERS.
	Perform cross-functional coordination meeting	The organization should also perform COORDINATION MEETING with cross functional teams that involves different teams within a project or involves PRODUCT MANAGEMENT BOARD.
Perform routine activities	Obtain periodic report	The MANAGER receives periodic report that present the latest status of the TASKs.
	Receive impediments	An ACTOR can report an IMPEDIMENT to the MANAGER
	Analyze task status	The MANAGER analyze the TASK status proactively by performing SITE VISIT, making a PHONE CALL, or during a COORDINATION MEETING.
	Obtain impediments	The MANAGER identifies any IMPEDIMENTs occur in the distributed TEAM.
	Analyze causality	The MANAGER analyze the source and the cause of the issue and report it in the ROOT CAUSE ANALYSIS
	Identify the needs of coordination	Based on the ROOT CAUSE ANALYSIS, the MANAGER identifies the needs of coordination mechanisms which will be defined in the next activity.
Determine appropriate control mechanism	Identify team self-organizing capability	The organization checks the team members' ability in managing the dependencies
	Supervise directly	If the team could not manage the dependencies by themselves, it is better that the MANAGER or the facilitator (such as Scrum Master) supervise the works directly.
	Facilitate self-organizing team	The MANAGER shares the authority in organizing TASKs and RESOURCEs to the TEAM.
	Provide standardization	The MANAGER provides the standardization of WORK PROCESSEs (or way of working) and the WORK PRODUCTs for the team.
<b>Activity</b>	<b>Sub Activity</b>	<b>Description</b>

Supervise directly	Initiate task	The MANAGER defines the TASKs that should be executed
	Allocate task	The MANAGER allocates the TASKs to the appropriate TEAM MEMBERS based on specific TASK ALLOCATION PLAN.
Facilitate self-organizing team	Set up meeting sessions	The MANAGER (or a facilitator such as Scrum Master) sets up a COORDINATION MEETING session. It is suggested to follow Agile principles to conduct daily meeting also planning, review, and retrospective meeting which are believed can help the organization to improve the teamness among distributed TEAM MEMBERS.
	Help team to identify issues	The TEAM should be able to identify the issues by themselves with the helps or support from the MANAGER.
	Encourage team to define solution	As well as identifying the issues, the TEAM should also be able to determine the appropriate solution to address the issues.
	Help team to organize task	Without the MANAGER's intervention, the TEAM have the responsibility of the TASK distribution, or in other words, TEAM MEMBERS must proactively take the appropriate TASKs
Provide standardization	Provide standardization of work processes	Standardizing work processes by providing work guidelines or tools that support the collaboration will lead to a faster work performance.
	Provide standardization of outputs	Standardizing the quality of work outputs and the reporting (e.g. documentation) helps the transition of chain processes easier and faster by reducing the dependency on knowledge and resources.
Determine appropriate communication mechanism	Encourage the use of synchronous communication to improve teamness	The organization (whatever the challenges are) should try to optimize the synchronous communication as much as possible; even it only has limited hours of overlapped working hours with the remote teams.
	Identify temporal challenge	The organization should determine the issues caused by the temporal challenges, such as difficulties in understanding the requirements because of limited collaboration/meeting time.
	Improving direct communication	The TEAM LEADER or MANAGER tries to improve the limited of synchronous communication.
	Identify cultural challenge	The MANAGER identifies any SOCIO-CULTURAL CHALLENGES that could harm the communication among distributed teams.
	Perform cultural internalization	The organization conduct an event that enable the dispersed teams with different cultural background to learn each other. There are several ways to perform cultural internalization such as SITE VISIT and team building
	Identify cross-functional	The MANAGER tries to identify whether any issues in the communication between the distributed teams that have different functional tasks.

	communication problem	
	Providing access	The organization provide the access to the remote team.
	Improve communication protocol	The organization can improve the communication protocol by providing a form or a structure of message for email or distributed documents that have been agreed upon by all teams.
	Organize expertize	The organization identifies the location of the expert and the expertise he or she possesses.
	Knowledge gap identification	The organization identifies the gap of knowledge among distributed teams.
	Make knowledge accessible	The organization selects the appropriate knowledge transfer approaches based on the types of the available and required KNOWLEDGE (Section 5.1.4).
Providing access	Assign communication broker	In a case where it is difficult to have DIRECT COMMUNICATION, and INDIRECT COMMUNICATION through a mediator can be done by assigning a COMMUNICATION BROKER.
	Share organization structure	By sharing the organization structure, team members can know easily to whom they want to talk with when they need to collaborate with (Deshpande et al., 2011)
Improving direct communication	Adjust working hours	Adjusting working hour from one or both dispersed locations will increase opportunity to have more collaboration time
	Encourage the use asynchronous communication	Organization can communicate with other team members at different countries to complement the limited direct synchronous communication such as sending the requirements through email or remind his/her colleagues to check the requirement updates at the SharePoint.
Knowledge gap identification	Identify team's cognitive level	The organization identifies the level of knowledge and expertise of the teams.
	Identify required knowledge	The organization identifies the required knowledge for the team to be able to perform assigned TASKs.
	Analyze knowledge gap	The organization compares the available and the required knowledge to determine the KNOWLEDGE GAP.
Finalization and Improvement	Review current practices	The organization performs a holistic PERFORMANCE ANALYSIS at the end of the project (in Agile it can be a review or retrospective meeting) and reviews the current coordination practices.
	Determine improvement plan	The organization determine the current COORDINATION PROFILE and plan to improve the future coordination practices.

*Table F-2 Table of Concepts*

Concept	Description
---------	-------------

STRATEGY	STRATEGY is a unified, comprehensive, and integrated plan that is designed to ensure that the basic objectives of the enterprise are achieved (Glueck, 1980, p. 9)
BUSINESS STRATEGY	The STRATEGY which is developed to achieve the OBJECTIVES in a business environment (Fitzgerald & Stol, 2017; Glueck, 1980)
PRODUCT MANAGEMENT STRATEGY	The STRATEGY which is defined in managing software productization that is aligned with BUSINESS STRATEGY (Fitzgerald & Stol, 2017; Fricker, 2012)
OBJECTIVE	OBJECTIVE is the goal that want to be achieved by an organization through its STRATEGY.
CHANGE MANAGEMENT PLAN	Change management in this context is related to the changes of organization's strategic initiatives or objectives. It means of perceiving the change in strategy, business processes, the software process management approaches that could impact the business performance (Nidumolu, 1996).
SITUATIONAL FACTOR	Various aspect that could impact to the configuration of a method (Brinkkemper et al., 1999; van de Weerd & Brinkkemper, 2009)
CHALLENGE	Particular issue that is associated to global software engineering caused by the dispersion of the team members (Olsson, Conchúir, et al., 2006).
GEOGRAPHICAL CHALLENGE	GSE CHALLENGE caused by the distance between distributed partners involved in the project (Šmite, 2007)
TEMPORAL CHALLENGE	GSE CHALLENGE characterized by the level of working hours overlay (Šmite, 2007).
SOCIO-CULTURAL CHALLENGE	GSE CHALLENGE caused by the difference level of social, ethnic, and cultural fit between teams from different national locations (Šmite, 2007).
KNOWLEDGE GAP	GSE CHALLENGE caused by the different level of KNOWLEDGE and expertise or difficulties to access the source of KNOWLEDGE among distributed teams (Kotlarsky et al., 2008)
CONTEXTUAL CHALLENGE	GSE CHALLENGE caused by the organizational heterogeneity in process maturity and inconsistency in work practices, methodology, or tools (Šmite, 2007)
DEPENDENCY	"Extent to which a unit's outcomes are controlled directly by or are contingent upon the actions of another unit" (Victor & Blackburn, 1987, p. 490)
RESOURCE	RESOURCE can be an ACTOR, a WORK PRODUCT, or an effort of an ACTOR that performs as a running TASK; or is produced or used in a running TASK (Crowston, 1994)
PROCESS	a series of actions or operations conducting to an end ("Process," 2017)
KNOWLEDGE	"The fact or condition of knowing something with familiarity gained through experience or association" ("Knowledge," 2017)
COORDINATION PROFILE	Organization capabilities categorization in managing coordination tasks among globally distributed team. The profiles are adopted from CMMI and task coordination pyramid (Section 5.2)
<b>Concept</b>	<b>Description</b>

COORDINATION SUPPORT	The infrastructure of task coordination in a form of ORGANIZATIONAL SUPPORT and TOOL SUPPORT
ORGANIZATIONAL SUPPORT	The support from organization infrastructure that can be form of ROLES, business FUNCTIONS, the organization structure itself, and the organization STRATEGY.
TOOL SUPPORT	The COORDINATION SUPPORT in a form of physical artifacts such as IT system or project management artifacts (e.g. burn-down chart)
ROLE	A position that an ACTOR gets by its virtue. ROLE is not always someone's job position.
FUNCTION	A task that should be performed by a ROLE.
ACTOR	The one who plays ROLE.
KNOWLEDGE MANAGEMENT TOOL	A tool that is used to facilitate knowledge sharing.
PROJECT MANAGEMENT TOOL	A tool or project artifacts that is used to manage or monitor the project.
COMMUNICATION TOOL	A tool that facilitates communication.
COLLABORATION TOOL	A tool that facilitate work collaboration.
SUPPORT GAP ANALYSIS	The analysis of task coordination support deficiency in a GSE project.
SUPPORT REQUIREMENT	The analysis of required support to facilitate task coordination.
NON-FUNCTIONAL REQUIREMENT	A list of requirements that are not related to the (developed) system's behavior (Schwaber, 2004).
TASK	The smallest unit of work that is assigned to the team member (Cossentino, Gaglio, Henderson-Sellers, & Seidita, 2006).
IMPEDIMENT	Anything that can slow down the team in performing their TASKs (Schwaber, 2004).
COORDINATION MECHANISM	An approach of task coordination.
CONTROL MECHANISM	A mechanism to manage dependencies among distributed team members.
COMMUNICATION MECHANISM	An organic coordination mechanism to manage dependencies through providing feedback and mutual adjustment (Van De Ven et al., 1976).
TEAM	A set of TEAM MEMBER
PRODUCT BOARD MEMBER	A group of people from with ROLES who become the steering committee of software productization.
MANAGEMENT	Or board of management, they who play the "C" role.
PRODUCT MANAGER	A ROLE (also can be a job position) who has responsibility in communicating the voice of customer (in this context the management board also business users e.g. Sales team) and realizing the product software roadmap.
PRODUCT OWNER	A ROLE who articulates the customer's voice into user stories.
PERFORMANCE ANALYSIS	An analysis of a project performance both technical and budget performances.
<b>Concept</b>	<b>Description</b>

IMPROVEMENT PLAN	A set of tasks or practices that should be performed in the future to have better performance than the current period.
INITIAL	A state where the company has not been specifically defined functions in business processes and organizational structure regarding GSE and tend to be reactive in dealing with problems in the coordination of tasks (Table 5-5).
MANAGED	A state where the organization has managed task coordination in GSE projects by using current organizational processes and structure (Table 5-5).
DEFINED	A state where the organization has been specifically defined functions in business processes and organizational structure regarding task coordination in GSE projects (Table 5-5).
QUANTITATIVELY MANAGED	A state where the organization has defined the process control and able to contextualize the information. The distributed team also have considered the workspace awareness (Table 5-5).
OPTIMIZING	A state where the organization also continuously improve the approach in managing task coordination in GSE projects (Table 5-5).
ROOT CAUSE ANALYSIS	An approach of problem solving by identifying the root causes of the problems
SPRINT BACKLOG	A list of tasks that should be accomplished within a sprint (Schwaber, 2004).
TASK ALLOCATION PLAN	A plan that determine the most optimize way to allocate tasks
CUSTOMER	In this context, CUSTOMERs are the business users that represents the real customer of a product software company, such as sales and marketing team.
ENGINEER	They who have the functions related to the product realization., for example developers and architects.
DIRECT SUPERVISION	A coordination approach that is achieved by having one individual take responsibility for the work of others (Mintzberg, 1979)
MUTUAL ADJUSTMENT	A coordination approach that is achieved by sharing the control of the work rests in the hands of the doers (Mintzberg, 1979)
COORDINATION MEETING	A formal meeting among (distributed) team members where coordination is also performed.
SOLUTION	A mean of problem solving.
DIRECT COMMUNICATION	A communication practice that involves active listening among partners.
PHONE CALL	Clear
WEBEX	A video conference product by Cisco (Cisco, 2017)
SITE VISIT	An official travel between a company to visit its remote facility or partner (vice versa)
VIRTUAL OFFICE	A formal or business office where the team members are separated by distance but be able to communicate through online communication network (Van Gameren, Van Solingen, & Dullemond, 2013).
SYNCHRONOUS COMMUNICATION	The communication among partners is done concurrently (Olsson, Conchúir, et al., 2006)
ASYNCHRONOUS COMMUNICATION	The communication among partners is done not in a real-time (Olsson, Conchúir, et al., 2006).
EMAIL	Clear
<b>Concept</b>	<b>Description</b>

CHAT	Clear
CULTURE	"The set of values, conventions, or social practices associated with a particular field, activity, or societal characteristic" ("Culture," 2017)
INDIVIDUAL CULTURE	Someone's ethical and professional behavior
TEAM CULTURE	The behavior of a team that built from the common characteristic of the team member.
ORGANIZATION CULTURE	A set of shared attitudes, values, goals, and practices that characterizes an institution or organization ("Culture," 2017)
NATIONAL CULTURE	Similar to the definition of CULTURE and ORGANIZATIONAL CULTURE, NATIONAL CULTURE is characterized by the common norms, behaviors, beliefs, and customs of a sovereign nation.
INDIRECT COMMUNICATION	The communication practice that is perform through a COMMUNICATION BROKER
COMMUNICATION BROKER	The person as the intermediary that provides linkages, knowledge access, and communication bridge between two (or more) dispersed team members.
ORGANIZATION STRUCTURE	Clear
COMMUNICATION PROTOCOL	A set of rules that is used in the asynchronous communication (such as email) that provide the message content in a structured way.
KNOWLEDGE BASE	A repository of KNOWLEDGE.
TACIT	A type of KNOWLEDGE that has not been externalized yet
EXPLICIT	A type of KNOWLEDGE that has been documented and provided in a physical way
TEAM MEMBER	Clear
COGNITIVE REQUIREMENT	The level of knowledge and expertise that should be owned by a knowledge worker.
KNOWLEDGE GAP	The difference between the available knowledge owned by the knowledge workers and the required knowledge to perform a particular task.
KNOWLEDGE SHARING MECHANISM	The mechanism of transferring knowledge into different form, or facilitating the flow of knowledge among knowledge workers.
INTERNALIZATION	The knowledge transformation from EXPLICIT to TACIT.
SELF LEARNING	A practice of INTERNALIZATION where TEAM MEMBERS learn the KNOWLEDGE from the available EXPLICIT KNOWLEDGE by themselves.
FORMAL TRAINING	A practice of INTERNALIZATION where TEAM MEMBERS learn the KNOWLEDGE from the available EXPLICIT KNOWLEDGE through a formal event (e.g. classical training) provided by the company.
EXTERNALIZATION	The knowledge transformation from EXPLICIT to TACIT.
SOCIALIZATION	The knowledge transformation from EXPLICIT to TACIT.
MENTORING	A SOCIALIZATION practice where an expert guides another TEAM MEMBER and provides advises throughout the task execution
<b>Concept</b>	<b>Description</b>


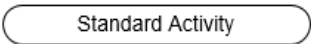


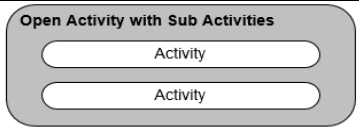

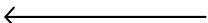



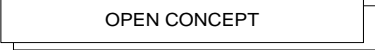

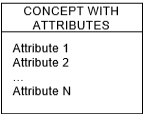
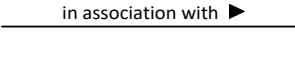


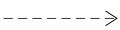


PAIRING	A SOCIALIZATION practice where two (or more) TEAM MEMBER with different level of knowledge or expertise work together at the same task that makes the less expert one can learn directly from the expert.
COMBINATION	The knowledge transformation from EXPLICIT to TACIT.
GROOMING	A COMBINATION practice where the KNOWLEDGE is enriched.
UPDATING	A COMBINATION practice where the KNOWLEDGE is renewed because of the latest version is outdated.



## Appendix H. PDD Notation

Table G-1 PDD's notations description (van de Weerd & Brinkkemper, 2009)

Notations	Descriptions
<i>Process View</i>	
	Initial state
	Standard activity: An activity that contains no further activities.
	Open activity: An activity that consists of a collection of sub-activities which are expanded in the same diagram or another diagram.
	Closed activity: A complex activity where its sub-activities are not expanded since it is known or not relevant in the specific context.
	Open activity with sub-activities. An activity that consists of a collection of sub-activities which are depicted inside it. This notation also is used to describe a set of unordered activities
	Branch: A state where the process is split into two or more routes based on specific criteria.
	Transition: A notation that explains the flow of the process.
	Forking and Joining. Forking and Joining use the same notation. Forking is used to start a set of concurrent activities, and Joining is used as the end state of parallelism.
	End state
<i>Deliverable View</i>	
	Standard concept: A concept that contains no further concepts.
	Open concept: A concept that consists of an aggregate of other concepts which are shown in the same or another diagram.
	Closed concept: A complex concept where its sub-concepts are not expanded in the same diagram since it is not relevant in the specific context.
	Concept with attributes: A concept (can be a form of standard, open, or closed concept) which describes its attributes.
	Relationship: A structural relationship that connects two concepts and specifies how concepts are linked to another.
	Aggregation: A specific type of relationship that represents the relation between a concept containing other concepts.
	Generalization: A relationship between a general concept and more specific concepts.
	Connection: Connecting process to delivered or utilized concepts.

## Appendix I. Expert Opinion Interview Protocol

### Methodological Support for Task Coordination on Global Software Engineering Project in a Product Software Organization

Interview Protocol  
Department of Information and Computing Science



Universiteit Utrecht

Utrecht, The Netherlands

Interviewee : \_\_\_\_\_  
Date & Time : \_\_\_\_\_  
Interviewers : Carolus Borromeus Widiyatmoko  
Research Supervisor : dr. Sietse J. Overbeek  
Prof. dr. Sjaak Brinkempper

First of all I want to thank you for your cooperation and taking the time to conduct this interview. The purpose of this interview is to gather information on the current practice of task coordination in distributed software engineering project in your organization.

After the presentation (10-15 minutes), in the following 25-30 minutes, we will run through this list in the form of an interview. If during the interview you ever feel uncomfortable or if you for any reason may wish not to answer, you are ever free to do so. This interview will be recorded, will only be used for this research, and will never be disclosed to third parties.

Then, we define a set of concepts as our method acceptance requirements which have been used in evaluating research artifacts from previous studies. We adopt concepts from Technology Acceptance Model (TAM) which are perceived usefulness and perceived ease of use (Polančič et al., 2010, p. 583; Wagenaar et al., 2017, p. 816). We also consider to evaluate the model based on the criteria in evaluating a method designed by method assembly approach (Brinkkemper et al., 1999) which are Completeness, consistency, efficiency, reliability, and applicability.

#### A. Interviewee Profile (5')

1. What is your job position and your job roles or functions in this organization related to the GSE projects?
2. How long (years of experiences) have you been working or involved in GSE projects / research?
3. How do you estimate the coordination experience level of this organization for managing interdependencies in GSE projects?

#### B. Assessment Criteria

Please answer the following questions short answer such as Yes or No, but suggest the interviewee to give his/her short reason why the answer is 'No'.

**Meta-modeling assessment**

1. Do you feel that the situational method contains all the method fragments that are referred to by other fragments in the situational method (Completeness)?
2. Do you think that all the activities, products, tools, and people do not contain any contradiction and are thus mutually consistent (Consistency)?
3. Do you think that the method is semantically correct and meaningful (Reliability)?
4. Do you think that the method can be perform at minimal cost and effort (Efficiency)?
5. Do you think that stakeholders can apply the method (Applicability)?

**Usefulness**

6. Do you think the method is useful for you?
7. Do you think that the method enable you to accomplish your tasks more effectively?
8. Do you think that the method increases your productivity?
9. Do you think that the method will help you to increase your experience level in coordinating tasks in GSE projects?

**Ease of Use**

10. Do you think that the method is clear and understandable?
11. Do you think that you would find the method easy to use?

**Behavioral Intention**

12. Do you think that you would have an intention to use the method?
13. Are you going to share the method to your colleagues or use the method for all the organization?
14. Will you fully use the capabilities of the method?

**C. Open Questions (10')**

For experts, elaborate their reasoning if they answered 'No' for 1-5.

Suggest the interviewee to answer concisely.

1. What are your initial thoughts on the method?
2. Are you going to bring up any other approaches for the method which are valuable for you or your organization?
3. What are the strength points of this method?
4. What are the weaknesses of this method?
5. Do you have any suggestions to improve the acceptability of this method?