FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Towards a Light-weight Distributed Software Development Process:

Empirically Driven Design of the Agile Distributed Adaptable Process Toolkit (ADAPT)

DISSERTATION

zur Erlangung des akademischen Grades

Doktor der technischen Wissenschaften

eingereicht von

DDipl.-Ing. Raoul Vallon, BSc
Matrikelnummer 0525496

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: Thomas Grechenig

Diese Dissertation haben begutachtet:

(Prof. Dr. Thomas Grechenig) (Prof. Dr. Rafael Prikladnicki)

Wien, 18.04.2016

(DDipl.-Ing. Raoul Vallon, BSc)

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Towards a Light-weight Distributed Software Development Process:

Empirically Driven Design of the Agile Distributed Adaptable Process Toolkit (ADAPT)

DISSERTATION

submitted in partial fulfillment of the requirements for the degree of

Doktor der technischen Wissenschaften

by

DDipl.-Ing. Raoul Vallon, BSc
Registration Number 0525496

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Thomas Grechenig

The dissertation has been reviewed by:

(Prof. Dr. Thomas Grechenig) (Prof. Dr. Rafael Prikladnicki)

Wien, 18.04.2016

(DDipl.-Ing. Raoul Vallon, BSc)

I'm not a great programmer; I'm just a good programmer with great habits.

Kent Beck, creator of XP and TDD

Deklaration

DDipl.-Ing. Raoul Vallon, BSc
Hernalser Hauptstraße 14/9, 1170 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbstständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

I hereby declare that I am the sole author of this thesis, that I have completely indicated all sources and aids used, and that all parts of this work - including tables, maps and figures - if taken from other work or from the internet, whether copied literally or by sense, have been labeled including a citation of the source.

Wien, 18.04.2016

Unterschrift Verfasser

Acknowledgment

First, I would like to thank Prof. Dr. Thomas Grechenig for his guidance, advice and support in this grand research adventure, making it possible to enter the world of academia, attend international research conferences and go on the research leave to Stanford.
Second, I would like to thank Prof. Dr. Larry Leifer and the whole designX lab of the Center for Design Research at Stanford University for the invitation to become a part of their lab for four months as a visiting researcher. It was a wonderful experience, both on a research-related and personal level and their input certainly enriched this dissertation. I greatly value and appreciate their advice on keeping the ADAPT framework tangible and accessible to both the researcher and the practitioner. I would also like to thank the Austrian Marshall Plan Foundation for making this research leave financially feasible.
I owe a great debt of gratitude to my fiancée Christiane for allowing me the time to concentrate on my dissertation, including many evenings, weekends and even some vacations, for providing motivation and support and also for proof-reading the thesis. I would also like to thank my family for enabling me to pursue my studies at TU Wien in the first place.
Furthermore, I would like to thank Dr. Brigitte Brem for her scientific support throughout the years and Dr. Alexander Zapletal for reviewing the final thesis draft and providing valuable feedback. I would also like to thank the anonymous reviewers who provided feedback to my submissions to various journals and conferences, especially those who also dedicated time to provide suggestions for improvement, thus making an impact on this research. On a related note, I am also thankful having been able to meet research leaders and colleagues from around the globe at international conferences, whose feedback also had an impact on this thesis.
Last but not least, I would also like to explicitly extend my gratitude to all participants of the multiple-case study, providing the empirical basis to the ADAPT framework, and to the experts who participated in focus groups or one-on-one interviews to evaluate the results of this dissertation and provided valuable and much appreciated feedback.

Abstract

Developing software in distributed environments is more complex as communication, coordination and control challenges arise. Adaptations to process models for collocated teams become necessary. There has been a growing interest in the past years in transferring agile values to distributed software development (DSD) to mitigate those challenges. However, while there are numerous empirical studies that report successful applications, there is to date no process framework providing a holistic approach for implementing agile practices in DSD environments. This thesis strives to fill that research gap by creating the ADAPT (Agile Distributed Adaptable Process Toolkit) framework. The design theory behind the framework includes five testable propositions to describe the nature of ADAPT. Related work is systematically analyzed in a mapping study, covering the 15-year time span of 1999 to 2014. The mapping study shows that - although the research field matured - empirical context of a case is rarely described to a full extent. This finding calls for a checklist to report empirical context in agile DSD. Based on the mapping study, a checklist is designed and then applied to drive the data extraction in this thesis' multiple-case study, which features three cases of varying distribution scenarios: cross town, no timeshift and continental. Using a grounded theory approach, 49 guidelines and 94 practice candidates are extracted in total from single-case analysis. These candidates are then compared cross-case to see common patterns emerge, which results in 10 guidelines and 29 full practices as well as 7 conceptual practices, together forming the first full iteration of the ADAPT framework as the output of this thesis. Two focus groups and ten expert interviews are held to evaluate both the design and the resulting ADAPT framework. Experts agree that the current state of the framework contributes to the grand challenge of applying agile values to DSD, and that it is worthwhile to continue with further iterations of the framework in future work, both growing the empirical basis and testing the current utility of the framework in a real-world implementation study.

Keywords. *Agile, Scrum, Distributed Software Development, Process Framework, Case Study, Systematic Mapping, Grounded Theory*

Kurzfassung

Verteilte Softwareentwicklung zeichnet sich durch größere Komplexität aus, da sowohl Kommunikation, Koordination als auch Kontrolle schwieriger zu bewältigen sind und bekannte Vorgehensmodelle adaptiert werden müssen. In den vergangenen Jahren haben sich daher Forschung und Praxis vermehrt mit der Problemstellung befasst, agile Prozesse auch für verteilte Standorte zu etablieren. Es gibt bereits mehrfach empirische Studien, die Erfolge vermelden, jedoch noch kein Prozess-Framework für eine gesamtheitliche Herangehensweise. Diese Arbeit will das vorliegende Forschungsproblem durch die Erstellung des ADAPT (Agile Distributed Adaptable Process Toolkit) Framework lösen. Der Designprozess des Frameworks leitet fünf testbare Aussagen her, die den Kern des Frameworks beschreiben. Weiters wird ein Systematic Mapping verwandter Arbeiten von 1999 bis 2014 durchgeführt. Die Ergebnisse des Mappings zeigen unter anderem, dass sich das Forschungsfeld zwar deutlich weiterentwickelt hat, aber der empirische Kontext in den meisten Studien nicht ausreichend beschrieben wird. Diese Erkenntnis führt dazu, das im Rahmen der Arbeit eine Checkliste erstellt wird, die die Datenextraktion für empirische Studien in verteilter agiler Softwareentwicklung definiert. Diese wird für die drei im Rahmen der Arbeit durchgeführten Fallstudien in den folgenden Verteilungsszenarien genutzt: innerhalb einer Stadt, keine Zeitverschiebung und kontinental. Durch Anwendung von Grounded Theory werden 49 Richtlinien und 94 Praktiken insgesamt als Kandidaten der einzelnen Fallanalyse extrahiert. Die weiterführende gesamtheitliche Analyse über alle drei Szenarien resultiert in insgesamt 10 Richtlinien und 29 vollumfänglichen Praktiken sowie 7 weiteren konzeptuellen Praktiken, die zusammen die erste volle Iteration des ADAPT Frameworks als Ergebnis dieser Arbeit darstellen. Zur Evaluierung von Design und Ergebnissen wurden zwei Fokusgruppen und zehn Experteninterviews durchgeführt, die sowohl den Beitrag des Lernframeworks für Forschung und Praxis als auch den in zukünftigen Iterationen leicht erweiterbaren modularen Aufbau unterstreichen.

Keywords. *Agil, Scrum, Verteilte Softwareentwicklung, Prozess-Framework, Case Study, Systematic Mapping, Grounded Theory*

Contents

CHAPTER 1

Introduction

The research motivation, challenges, objective, framework and outline have been presented and discussed at the 15th International Conference on Agile Software Development (XP 2014) in Rome and published as part of the joint PhD symposium report in the ACM SIGSOFT Software Engineering Notes. (Falessi et al., 2014)

Contents

1.1 Motivation

Agile software development has gained widespread popularity over the last decade in very different domains such as embedded software projects (Xie et al., 2012), mobile application development (Scharff and Verma, 2010) or aerospace (VanderLeest and Buter, 2009) and has found its way into global organizations and thus in multi-team and multi-site corporate environments on the scale of Intel (Chen et al., 2007), Microsoft (Begel and Nagappan, 2007), Yahoo! (Chung and Drummond, 2009) or SAP (Schnitter and Mackert, 2011). It is built around empowered and self-organizing teams with a strong focus on collaboration and communication supported by various agile practices including pairing, customer collaboration, stand-ups, reviews, retrospectives and the planning game (Šmite et al., 2010b).

Developing software in distributed environments has become a daily reality for many organizations as it offers benefits such as cost savings, access to large multi-skilled work forces and reduced time to market (Ó Conchúir et al., 2009). Agile software development may potentially improve collaboration in distributed environments as it relies strongly on frequent communication (Hossain et al., 2011b). Hence, agile practices have gained ground on distributed software development (DSD), e.g. (Sutherland et al., 2007; Paasivaara et al., 2009; Bannerman et al., 2012; Hildenbrand et al., 2008), and even global software development environments (GSD), e.g. (Hossain et al., 2011b; Cristal et al., 2008), in the last decade (Jalali and Wohlin, 2010, 2012a).

Distributed environments are more complex and several adaptations to process models for collocated teams are necessary (Hossain et al., 2011b; Batra, 2009), e.g. communication between team members needs other mechanisms (Dorairaj et al., 2012b; Korkala and Abrahamsson, 2007) and technical tool support plays a bigger role in the process (Dullemond et al., 2009; Niinimaki, 2011) as well as knowledge management and transfer (Dorairaj et al., 2012b). In general we can see a growing interest in transferring agile practices to distributed software development both as an active research field (Hossain et al., 2009; Jalali and Wohlin, 2010) and in practice, as the usage of distributed agile teams has more than doubled from 2012 to 2014: while 2012 35% of the respondents reported to work in geographically distributed agile teams, in 2014 (the latest available survey at the time of writing) the number increased to 80% (VersionOne, 2014).

Yet, although there are numerous empirical studies in the field, there has been no comprehensive framework for applying agile practices in DSD environments before the conduction of this thesis.

1.2 Research Challenges and Objective

Agile processes have been originally designed for collocated teams collaborating closely on a single site (Schwaber and Beedle, 2001). Distributed software development challenges one of agile software development's core strengths: team members need to interact and communicate on a daily basis to form self-organizing teams. However neither the leading agile process scrum (Schwaber and Beedle, 2001) nor Extreme Programming (XP) (Beck, 2000) were designed for teams working in distributed environments. Hence adaptations to the original process are necessary (Batra, 2009), but to date there is no standard process for applying agile practices to DSD (Alqahtani et al., 2013). The goal of these process adaptations is to transfer agile values, which produced excellent results in the last decade for collocated teams (Dingsøyr et al., 2012),

to DSD environments. This problem statement exhibits the following research challenges, formulated as research questions (RQ):

- RQ1. Why would distributed software development benefit from agile practices?[1]
- RQ2. What are suitable design components for building a distributed agile process framework?
- RQ3.
  a) What does the research landscape in the field look like in the 15 years of 1999 to 2014?
  b) What has changed in the later five years 2010 to 2014 in comparison to the former ten-year period of 1999-2009?
  c) What are common agile practices and distribution scenarios?
- RQ4.
  a) Which process design guidelines and practices can increase the chances of a successful agile process implementation in distributed environments?
  b) Do the different distribution scenarios affect the implementation of agile practices?

The research questions can be regarded, in the stated order, as a step-by-step road map to completing the research objective of this thesis, which reads:

Although numerous empirical studies have reported on the application of agile practices in different distribution scenarios, to date there has been no significant research effort to create a single comprehensive framework. To this end, the scientific achievement lies in designing such a framework called ***ADAPT*** *(****A****daptable* ***D****istributed* ***A****gile* ***P****rocess* ***T****oolkit) based on a multiple-case study to be conducted featuring different distribution scenarios (cross-town, no timeshift and continental) and discussed with regard to an extensive research on related work. The global scenario is explicitly out of scope for this thesis but the framework shall be designed to allow adding practices (from the global scenario, as well as others) in future work as well. Furthermore design guidelines will be provided to lead the tailoring of agile process implementations using the ADAPT framework and strengths and weaknesses evaluated by experts from both academia and industry.*

[1] RQ1 is considered an *introductory* research question because many empirical studies already exist to analyze the matter, but it is still posed to guide the analysis of the theoretical background in Chapter 2 and also to be in line with the remainder of the thesis' chapter design (cf. Table 1.1).

The framework is intended to be used by researchers to expand upon and practitioners to drive their process implementation. The framework is explicitly *not* considered to be an all-entailing "best practice guide" but a learning framework prepared for future iterations.

1.3 Research Framework and Outline

Based on the research challenges and objective, the following research framework in Table 1.1 was designed to carry out the research in this thesis. It also serves as a chapter outline to the remainder of this thesis.

Research Stage	**Chapter**	**RQs**	**Output**
Research Design	1	-	Research motivation, challenges, objective, framework, outline and contributions
Theoretical Background	2	RQ1	Definitions and explanations
Design Theory	3	RQ2	Design theory behind ADAPT, description of design components, 5 testable propositions
Systematic Mapping	4	RQ3	Overview of the 15-year period of 1999-2014, checklist for reporting empirical context in agile DSD
Single-Case Analysis	5	RQ4	Guideline and practice *concepts* from each respective case
Cross-Case Analysis	6	RQ4 (cont.)	ADAPT v1.0 (final set of cross-case guidelines and practices, connected to the challenge types)
Evaluation & Discussion	7	RQ1-RQ4	Discussion of results, evaluated in two focus groups and ten expert interviews, future work
Conclusion	8	-	Summarized conclusions

Table 1.1 – The research framework and chapter outline.

Chapter 2 explores the theoretical background behind the thesis, namely agile software development and distributed software development. The design theory behind the ADAPT framework is described in Chapter 3. Chapter 4 presents the systematic mapping study conducted to systematically analyze related work in the 15-year-period of 1999-2014 and also describes the checklist for reporting empirical context that is later applied in the multiple case study. For the case reports, first the three cases have been analyzed individually in Chapter 5 and then cross-case in Chapter 6 leading to the first full iteration of the ADAPT framework as the output of this thesis. The detailed research design for the multi-case study, including the steps for arriving at the final set of the ADAPT framework's guidelines and practices, can be found in Section 5.1. The discussion in Chapter 7 presents the evaluation of results from two

focus groups and 10 expert interviews and provides the final answers to the research questions within this thesis. Chapter 8 concludes the thesis.

1.4 Contributions

This thesis offers the following contributions to the research field of agile DSD.

- Empirical evidence from a long-term multiple-case study
- Cross-case analysis among different distribution scenarios
- A new learning framework for improving the implementation of agile practices in DSD that other researchers can expand upon and that can be used by practitioners to drive their distributed agile process implementation
- A systematic mapping of agile practices in DSD for years 2010-2014 in continuation of work by Jalali and Wohlin (2010) and thus covering the 15-year period of 1999-2014
- A checklist for researchers on reporting empirical context in agile DSD based on the results from the systematic mapping study
- Evaluation of research design and results by both research and industry experts in two focus groups, 10 expert interviews and several presentations at dedicated scientific international conferences

1.5 Publications

The following work has been published, or is in preparation to be published, during the course of this dissertation.

1. Vallon, Raoul and Grechenig, Thomas. Empirically Driven Design of the Agile Distributed Adaptable Process Toolkit (ADAPT): A Learning Framework. *In preparation*
2. Vallon, Raoul and Grechenig, Thomas. Trends and Directions of Applying Agile Practices in Distributed Software Development: A Systematic Mapping. *In preparation*
3. Vallon, R. and Grechenig, T. Ten Heuristics from Applying Agile Practices across Different Distribution Scenarios: A Multiple-Case Study. *Computer and Information Science*, 9(2):Online–First, May 2016
4. Vallon, R. Empirically Driven Design of the Agile Distributed Adaptable Process Toolkit (ADAPT). Technical report, Austrian Marshall Plan, 2015

5. Vallon, R., Wenzel, L., Brüggemann, M. E. and Grechenig, T. An Agile and Lean Process Model for Mobile App Development: Case Study into Austrian Industry. *Journal of Software*, 10(11):1245–1264, 2015

6. Vallon, R., Dräger, C., Zapletal, A. and Grechenig, T. Adapting to Changes in a Project's DNA: A Descriptive Case Study on the Effects of Transforming Agile Single-Site to Distributed Software Development. In *Agile Conference (AGILE), 2014*, pp. 52–60. IEEE, 2014

7. Vallon, R., Strobl, S., Bernhart, M. and Grechenig, T. Inter-organizational Co-development with Scrum: Experiences and Lessons Learned from a Distributed Corporate Development Environment. In *Agile Processes in Software Engineering and Extreme Programming. 14th International Conference, XP 2013, Vienna, Austria, June 3-7, 2013. Proceedings.*, volume 149 of *Lecture Notes in Business Information Processing*, pp. 150–164. Springer Berlin Heidelberg, 2013b

8. Vallon, R., Bayrhammer, K., Strobl, S., Bernhart, M. and Grechenig, T. Identifying Critical Areas for Improvement in Agile Multi-site Co-development. In *8th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE)*, pp. 165–172. SciTePress, 2013a

9. Vallon, R., Müller-Wernhart, M., Schramm, W. and Grechenig, T. Kombination von Agil und Lean in der Softwareentwicklung. *Springer Informatik-Spektrum*, In-Print 37(1):28–35, 2014, Online–First, 2012

10. Falessi, D., Oliveira, R., Taylor, K., Fontana, R. M., Power, K., Vallon, R., Giardino, C., Rejab, M. M. and Wang, X. Trends and emerging areas of agile research: the report on XP2014 PhD symposium. *ACM SIGSOFT Software Engineering Notes*, 39(5):26–29, 2014

CHAPTER 2

Theoretical Background

Contents

This chapter investigates definitions and theoretical background of this thesis. Section 2.2 tackles *RQ1. Why would distributed software development benefit from agile practices?* by investigating related work.

2.1 Agile Software Development

The content of this section has already been presented in (Vallon, 2012). Agile software development is regarded as an answer to the problem that even with exhaustive planning, the resulting software is seldom of high quality. One of the reasons is constantly changing requirements that are part of most of today's projects (Dogs and Klimmer, 2005). Hence agile processes try to use a more lightweight approach in planning to cope with changing requirements. Furthermore, to establish a common ground for all agile followers, the *Agile Manifesto* (Fowler and Highsmith, 2001) has been negotiated among 17 American software engineering thought leaders in 2001. The main part reads:

"We are uncovering better ways of developing software by doing
it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation

Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more." (Fowler and Highsmith, 2001, p. 35)

The Agile Manifesto points out that individuals and interactions are one of the key issues of software engineering from a process' perspective. This also implies that hierarchies are loosened compared to more traditional approaches (e.g. *Rational Unified Process*, cf. (Kruchten, 2004)) and planning needs to be done on a regular basis instead of following a strict plan in the (still frequently used) *Waterfall Model* (Royce, 1970). Flexibility and customer collaboration is increased and documentation kept to a minimum (Schatten et al., 2010), because it should not be used as a substitute for interaction (Highsmith, 2004).

Among the most widely used agile methodologies are *scrum* (cf. Section 2.1.1), Extreme Programming (XP) (cf. Section 2.1.2) or a combination of the two, such as *XP@Scrum* (cf. Section 2.1.3). However, it is important to truly adopt agile principles rather than strictly follow one of the agile methods or it will lead to "constant struggles and many other old school problems" (Fraser et al., 2006, p. 938). Although originally intended only for small-scale and not life-critical application (Cockburn and Williams, 2003), agile methodologies can be used in all kinds of software projects; they have even been applied successfully to disaster management after a terrorist attack (Nawaz and Zualkernan, 2009). Lean software development and its most famous follower kanban (cf. Section 2.1.4) also share many similarities with agile methods (Kniberg and Skarin, 2010) and are regarded as a process under the agile umbrella (Jalali and Wohlin, 2010), which is why they are discussed within this section. Kanban and scrum can also be successfully combined to a hybrid process in its own right (Vallon et al., 2012).

2.1.1 Scrum

The first presentation of scrum was held by Ken Schwaber (Schwaber, 1997) during a workshop at OOPSLA conference in 1995 (Sutherland, 1995). The name *scrum* originates from "the strategy used in rugby for getting an out-of-play ball back into play" (Schwaber and Beedle, 2001, p. 1). It has been chosen because "both [the game rugby and scrum] are adaptive, quick, self-organizing, and have few rests" (Schwaber and Beedle, 2001, p. 1). The most important properties of scrum can be derived from these similarities: high productivity, adaptivity, low risk and uncertainty resulting in increased comfort for practitioners (Grechenig et al., 2010). A summary of scrum history and papers by the co-creators Sutherland and Schwaber can be found in (Sutherland and Schwaber, 2007). The scrum overview in this section is based on

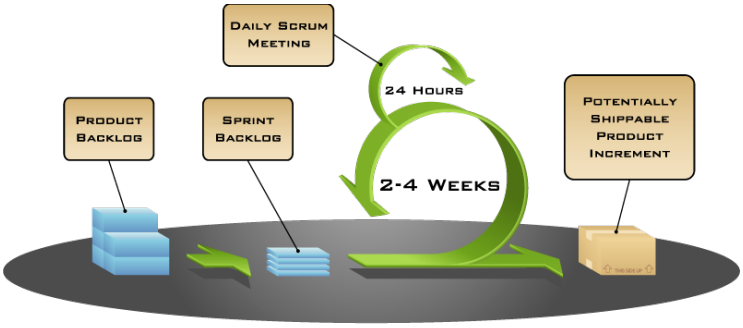


Figure 2.1 – Scrum process overview (Mountain Goat Software, 2005).

the sources (Schwaber and Beedle, 2001; Dogs and Klimmer, 2005; Pichler, 2008; Gloger, 2011). Figure 2.1 provides a quick overview of the scrum process.

Requirements in scrum are defined as *user stories*. A user story is a requirement written from the user's point of view (in a specific role), e.g. "As standard user I want to be able to create a new meeting" (Pichler, 2008). Phrasing user stories correctly is a very important issue as books have been written focusing explicitly on the matter (Cohn, 2004; Wirdemann, 2011). The *product backlog* is the collection of all user stories of the product to be developed. Scrum works with *sprints*, i.e. iterations of typically 2-4 weeks. At the beginning of each sprint, user stories of the product backlog are pulled into the *sprint backlog* by the team during the *sprint planning* meeting. The team commits to developing all the selected user stories within the sprint. Furthermore, there is the *daily scrum* meeting in which current progress and impediments are discussed in a very short manner. At the end of each sprint a product increment has to be created including all the user stories that have been developed during the sprint. The team and stakeholders will then reflect on the current product increment (*sprint review meeting*) as well as on the scrum process itself (*sprint retrospective*), which shall help improve the next sprint iteration.

Roles

Scrum defines three types of roles, *product owner* (PO), *scrum master* (SM) and the *team*. However, none of the three is a project manager in a traditional sense. The question is whether or not managers are needed and, if so, what they can or should do in agile environments. Anderson et al. (2003) argue that managers are definitely needed in agile environments, but their tasks may change. Ward Cunningham concludes that "a manager [in agile methods] does more oversight than day-to-day 'managing' of the programming activities" (Anderson et al., 2003, p. 276), which gives managers in agile environments more time to focus on the important administrative matters due

to self-organizing development teams. The following description shows how management tasks are divided among the three roles.

The *product owner* takes over some of the traditional management tasks but without leading the team (Pichler, 2008):

- Definition and management of requirements
- Release management and return on investment
- Close collaboration with the team
- Stakeholder management

He also serves as the link between the team and the customer, i.e. he communicates with the development team and represents the customer's interests, e.g. when defining user stories of the product backlog and setting their priorities. Regular meetings with the customer are absolutely necessary. The product owner works with the team during the whole project, i.e. requirements are constantly being refined and product increments are being reviewed at the end of each sprint. Yahoo! states that the product owner is the "single wringable neck" (Pichler, 2008), so he is solely responsible for the success or failure of a project. Thus the product owner needs to take many different interests into account (customer, marketing, service, etc.) and update the product backlog accordingly, i.e. add, refine or delete requirements during the course of the whole project, not just in the beginning. In bigger projects with multiple scrum teams the product owner role can be very complex and demanding, which is why Pichler (2008) advises to have one product owner for each team. The various product owners then form another team: the product owner team that may also include marketing, service or other representatives and one *chief product owner*. Gloger (2011) also states that in complex project situations several product owners are possible and that coordination among them needs to be done one level above the teams.

While the product owner is responsible for the success of the project, the *scrum master* is responsible for a working scrum process (Schwaber and Beedle, 2001). Pichler (2008) identifies the following tasks for a scrum master:

- Establish scrum as the process model in the team
- Support the team
- Ensure direct collaboration between product owner and team
- Remove impediments
- Help improve development methods

- Lead by serving

Greenleaf (2002) characterizes a *servant leader*, i.e. a leader without authority that supports the team. Gloger (2011) defines the scrum master as a powerless change manager, because he does not have any authority, yet needs to create and sustain a working scrum process. This difficult task must be taken seriously because "A dead scrum master is a useless scrum master!" (Ken Schwaber in (Gloger, 2011, p. 26)). Furthermore, the scrum master has to assure that the team does not trade quality for productivity (Schwaber, 2007). In conclusion, the scrum master has influence, but no power or authority regarding the team's organization (Pichler, 2008).

The *team* is fully self-organizing. While the product owner prioritizes the user stories, the team itself selects the user stories that it can commit to in the next sprint (following the product owner's prioritization). The team needs to be interdisciplinary (from architecture to testing) and work autonomously, i.e. it needs to be able to reach the sprint objectives without major external dependencies. If a user story is not ready for deployment at the end of the sprint, then it is neither the developer's nor the tester's fault. The whole team is held responsible. Lencioni and Schieberle (2014) argue that in good teams its members need to call each other to account to achieve goals and show their mutual respect.

Meetings

Scrum prescribes a variety of meetings that reflect the agile character of "individuals and interactions" (cf. Agile Manifesto (Fowler and Highsmith, 2001)).

At the beginning of each scrum project there is a *project planning* session where the vision, project staff and conventions (e.g. programming language, tools, etc.) are set. The product owner defines the first version of the product backlog and sets priorities for the user stories.

The product owner needs to have an updated and estimated backlog for the *sprint planning* meeting. The *estimation meeting* should be held at least once each sprint and should not exceed a total length of 90 minutes. The product owner can also use this meeting to present new backlog items to the team (Gloger, 2011). In contrast to traditional approaches, estimation is done on a team level (Wirdemann, 2011), i.e. "What can the team accomplish in one sprint?" instead of "How much can developer X implement or tester Y test in one sprint?". Moreover, estimations are conducted in reference to other user stories and by using *story points*, which is an abstract unit that the agile community has agreed upon (Gloger, 2011). The most widely used agile estimation method is Grenning's 2002 *Planning Poker*. Gloger (2011)

also introduced a new estimation method called *magic estimation*, because he argues that planning poker does not work well with bigger teams and backlogs. More on agile estimating can be found in (Cohn, 2005).

The *sprint planning* meeting takes place at the beginning of each sprint to decide on the sprint goal, i.e. which user stories from the product backlog will be put into the sprint backlog to be developed within this sprint. The team discusses possible ways of implementation, which improves the teams' understanding of the user stories similar to a requirements workshop (Gloger, 2011). The defined sprint goals are then kept in the sprint backlog.

The *daily scrum* is a daily stand up meeting that should take no longer than 15 minutes. The meeting is held standing up to enforce the short nature of the meeting. Every team member should state his status and problems (if any) shortly. More precisely, the scrum master will ask each team member the following three simple questions (Dogs and Klimmer, 2005):

1. What did you do yesterday?

2. What will you do today?

3. Are there any impediments in your way?

The scrum master takes notes of impediments (see *impediment chart* in Section 2.1.1) and will try to eliminate them as quickly as possible (Pichler, 2008). In bigger scrum teams it may help to focus the questions on individual user stories rather than individual team members to keep the meeting short and to the point (Davies and Sedley, 2010).

For larger projects with several scrum teams, the *scrum of scrums* meeting can be held daily as a project-wide stand up meeting that improves communication and coordination among several teams. Each team sends one team member to the scrum of scrums meeting. However, the scrum master should not be sent there too frequently in order for the team to remain self-organizing (Larman and Vodde, 2009).

The following questions need to be answered by each participant of the scrum of scrums (Pichler, 2008):

1. What did your scrum team do since the last scrum of scrums?

2. What will your scrum team do until the next scrum of scrums?

3. Are there any impediments in your scrum team's way?

In the *sprint review* meeting at the end of each sprint the developed product increment is presented. Participants of this meeting should be members of the management, the customer, user(s) and the product owner (Schwaber and Beedle, 2001). The scrum master moderates the meeting. However, it is an informal meeting and therefore it is not allowed to prepare presentations. The product increment is the main issue. Participants should collect information for the next sprint planning meeting by identifying strengths and weaknesses of the current product increment. Furthermore, it is reviewed, which user stories have been implemented during this sprint (ideally, all that have been selected for the sprint backlog). The product owner decides if they have been implemented adequately.

Without a *sprint retrospective*, teams would make the same mistakes over and over again (Kniberg and Skarin, 2010). The sprint retrospective should be held immediately after the sprint review meeting. It is dedicated to reflecting on the scrum process itself during the last sprint, e.g. level of collaboration within the team and room for improvement in general (Schwaber, 2004). The objective is to increase productivity and efficiency of the team as well as overall software quality (Derby et al., 2006). After the sprint retrospective the sprint is formally completed.

Artifacts

The following artifacts support the scrum process. Product backlog and sprint backlog are the core components of every scrum implementation. Additionally, the burndown chart is an important tool to track progress within a sprint. The impediment chart is a valuable optional utility (Pichler, 2008).

The *product backlog* is created during project planning at the beginning of a scrum project. It is a list of user stories that should become part of the product. However, in contrast to traditional product specifications, it is intentionally kept incomplete (Schwaber and Beedle, 2001). This is part of agile thinking because the product backlog needs to be under constant development and refinement by the product owner. New user stories can be added or old ones deleted in every phase of the project. User stories are usually not only written by the product owner (but also by the team itself or other stakeholders e.g.), especially in bigger projects (Gloger, 2011). It is very important that the product owner keeps an updated prioritization of the items (i.e. user stories) in the product backlog at all times.

At the beginning of each sprint the team selects user stories from the product backlog to be added to the *sprint backlog*. The team then divides the user stories into tasks that will be worked on during the sprint. The sprint backlog

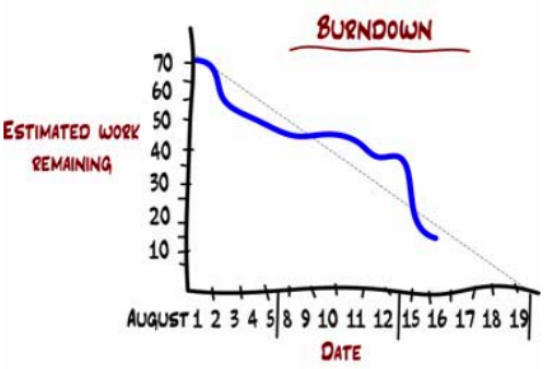


Figure 2.2 – Scrum burndown chart (Kniberg and Skarin, 2010).

may not change during the sprint with exception of new subtasks being added (Schwaber and Beedle, 2001), because the team has committed to completing all tasks in the sprint backlog within the sprint's duration.

The *burndown chart* is a tool to keep track of activities within a sprint (Dogs and Klimmer, 2005). More precisely, it shows the day-to-day progress of the sprint (Schwaber and Beedle, 2001) by comparing estimated and actual effort over time. Figure 2.2 shows an exemplary burndown chart. The ideal burndown describes the ideal (linear) sprint progress (Pichler, 2008). By summing up efforts (see y-axis of Figure 2.2), it is possible to check if the actual burndown of work is above or below the estimated burndown that is shown as a straight line in Figure 2.2. This way actual progress (burndown) is compared to the estimated one.

The *impediment chart* contains a short description of impediments as well as the date of first occurrence and removal (Pichler, 2008). The scrum master should update it at the end of each daily scrum meeting. It is the scrum master's duty to deal with the removal of impediments.

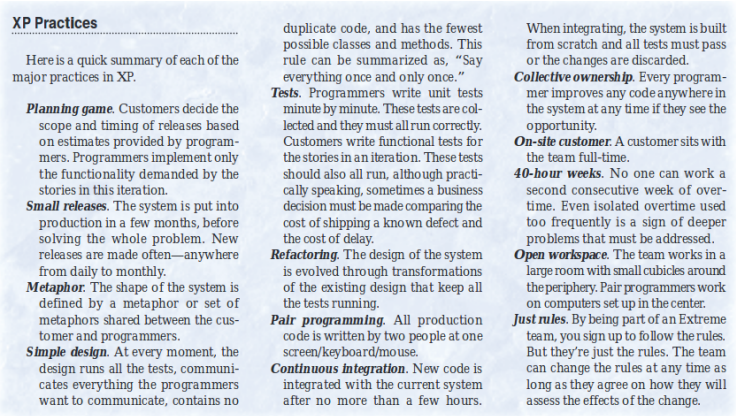
2.1.2 Extreme Programming (XP)

Extreme Programming goes back to Kent Beck with (Beck, 2000). He describes it as:

> "What is XP? XP is a lightweight, efficient, low-risk, flexible, predictable, scientific, and fun way to develop software. It is distinguished from other methodologies by
>
> - Its early, concrete, and continuing feedback from short cycles.

- Its incremental planning approach, which quickly comes up with an overall plan that is expected to evolve through the life of the project.
- Its ability to flexibly schedule the implementation of functionality, responding to changing business needs.
- Its reliance on automated tests written by programmers and customers to monitor the progress of development, to allow the system to evolve, and to catch defects early.
- Its reliance on oral communication, tests, and source code to communicate system structure and intent.
- Its reliance on an evolutionary design process that lasts as long as the system lasts.
- Its reliance on the close collaboration of programmers with ordinary skills.
- Its reliance on practices that work with both the short-term instincts of programmers and the long-term interests of the project."

(Beck, 2000, Preface p. X)



XP Practices

Here is a quick summary of each of the major practices in XP.

Planning game. Customers decide the scope and timing of releases based on estimates provided by programmers. Programmers implement only the functionality demanded by the stories in this iteration.

Small releases. The system is put into production in a few months, before solving the whole problem. New releases are made often—anywhere from daily to monthly.

Metaphor. The shape of the system is defined by a metaphor or set of metaphors shared between the customer and programmers.

Simple design. At every moment, the design runs all the tests, communicates everything the programmers want to communicate, contains no duplicate code, and has the fewest possible classes and methods. This rule can be summarized as, "Say everything once and only once."

Tests. Programmers write unit tests minute by minute. These tests are collected and they must all run correctly. Customers write functional tests for the stories in an iteration. These tests should also all run, although practically speaking, sometimes a business decision must be made comparing the cost of shipping a known defect and the cost of delay.

Refactoring. The design of the system is evolved through transformations of the existing design that keep all the tests running.

Pair programming. All production code is written by two people at one screen/keyboard/mouse.

Continuous integration. New code is integrated with the current system after no more than a few hours. When integrating, the system is built from scratch and all tests must pass or the changes are discarded.

Collective ownership. Every programmer improves any code anywhere in the system at any time if they see the opportunity.

On-site customer. A customer sits with the team full-time.

40-hour weeks. No one can work a second consecutive week of overtime. Even isolated overtime used too frequently is a sign of deeper problems that must be addressed.

Open workspace. The team works in a large room with small cubicles around the periphery. Pair programmers work on computers set up in the center.

Just rules. By being part of an Extreme team, you sign up to follow the rules. But they're just the rules. The team can change the rules at any time as long as they agree on how they will assess the effects of the change.

Figure 2.3 – The 12 Extreme Programming (XP) practices (Beck, 1999).

Figure 2.3 shows the 12 XP practices in a nutshell. The practices were not new at the time and go back to a diverse set of authors: Wood and Silver (1995); Martin (1991); Stapleton (1997); Alexander (1979); Takeuchi and Nonaka (1986); Cunningham (1996); Jacobson (1994); Gilb and Finzi (1988); Boehm (1988); Thomas (1998); Lakoff and Johnson (1998); Coyne (1995); Coplien (1998); DeMarco and Lister (1999). However, the combination of the practices to a new best practice process with short iterations to cope with

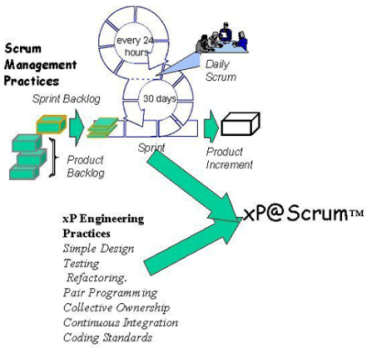


Figure 2.4 – XP@Scrum process (Mar and Schwaber, 2002)

changing requirements was novel and paved the way for the Agile Manifesto (Fowler and Highsmith, 2001) and other agile methodologies such as scrum (cf. Section 2.1.1).

2.1.3 XP@Scrum

Nowadays, it is quite common (cf. Chapter 4) to find a scrum process combined with XP development practices. The concept is simple: "XP focuses on engineering practices, and scrum on managerial and organizational aspect" (Vriens, 2003, p. 122). The term *XP@Scrum* was coined by Mar and Schwaber in 2002. Other work on combining scrum and XP include the experience report from at Philips (Vriens, 2003) and the book (Kniberg, 2007). Figure 2.4 shows the first published combination of XP and scrum. The XP@Scrum processes uses a general scrum process (cf. Section 2.1.1 extended by XP engineering practices (cf. Section 2.1.2)).

2.1.4 Lean Software Development and Kanban

To understand lean software development, its origins need to be investigated first. *Lean production* (also *lean management* or *just-in-time production*) originates from automobile industry, namely Taiichi Ohno's *Toyota Production System* in (Ohno, 1978). Toyota developed the new production model after World War II in the 1950s. The new lean production approach stood in stark contrast to commonly found mass production among competitors in automobile industry that started to stagnate in both the US and Europe at the time (Jones et al., 1990). The central assumption of lean production is to align production to what provides value for the customer and thus avoid rework. Everything that does not provide value to the customer is waste and needs to be eliminated (Ohno, 1988; Womack and Jones, 1996). Toyota observed that costs per unit were lower at smaller production levels than in bigger ones, which provided a competitive edge over mass production (Jones et al., 1990):

- Enormous inventory levels of mass production were diminished (beginning of just-in-time production)
- Defects could be spotted more easily at smaller production levels and be dealt with immediately

However, this new lean approach demanded highly qualified and motivated personnel, since the anticipation and correction of defects requires one's initiative before a blockage in the workflow occurred. Otherwise the whole production flow may stall (Jones et al., 1990). The lean production approach that revolutionized automobile industry can also be applied to software development. Anderson (2003) and Poppendieck and Poppendieck (2003) pioneered in the field in 2003 by putting great effort into introducing lean principles to software development.

The underlying lean principles in the Toyota Production System are (Womack and Jones, 1996):

1. Precisely specify value by specific product.
2. Identify the value stream for each product.
3. Make value flow without interruptions.
4. Let the customer pull value from the producer.
5. Pursue perfection.

However, the goal is not to bring manufacturing or the Toyota Production System to software engineering: "The Toyota Production System is Lean, but Lean is not the Toyota Production System. We are not trying to make software development look more like manufacturing, because Lean is not about manufacturing. Lean is about value streams." (Ladas, 2008, p. 13).

These principles have been incorporated into software engineering by means of the process *kanban* (see following chapter). Still, kanban is not the only lean software development process. Janes and Succi (2009) point out that two other known approaches to software development are also based on the lean pull principle (in contrast to traditional push approaches):

- *Test Driven Development* (TDD) by Beck (2003), i.e. writing test cases before code
- *Goal-driven Software Development* by Schnabel and Pizka (2006), i.e. defining goals before setting requirements and using these goals to pull requirements and their priorities

Kanban

The word kanban is Japanese for signal card and originates from the Toyota Production System (Ohno, 1988), in which kanban cards (also called *kanbans*, i.e. single work pieces) have been used to signal demand in the production flow. The function of kanban in software development is very similar: "Within software development, kanbans are used to 'pull' user stories into development. By limiting the amount of kanbans that are available one can limit the amount of user stories currently developed, i.e., the 'work-in-progress' or in other words, the amount of code that is not finished yet." (Janes and Succi, 2009, p. 2). That is the kanban process in a nutshell. A more precise definition follows. David J. Anderson has introduced kanban to software development in 2007 at the *Lean New Product Development conference* (Anderson, 2010). One year later, already six presentations have been held at the *2008 IEEE Agile conference* (Anderson, 2010), which shows the growing interest and need for kanban in software development. Kanban's recipe for success includes six steps (Anderson, 2010):

- Focus on quality
- Reduce work-in-progress
- Deliver often
- Balance demand against throughput
- Prioritize
- Attack sources of variability to improve predictability

Nevertheless kanban can be broken down into only three simple rules. The following description is based on (Kniberg and Skarin, 2010) and (Vallon, 2011).

Visualize the Workflow: Work is split into pieces and each item is written on a card and put on a kanban board that is divided into named columns to illustrate the workflow (Kniberg and Skarin, 2010). The kanban board is used to visualize the workflow that each item has to run through (Vallon, 2011). There are both physical and electronic kanban boards. Anderson (2010) argues that both have their right to exist, i.e. physical boards provide a better psychological effect while electronically ones simplify the creation of metrics and reports. An example kanban board and workflow is shown in Figure 2.5.

Limit Work In Progress (WiP): Limiting the work in progress (or WiP) means to "assign explicit limits to how many items may be in progress at

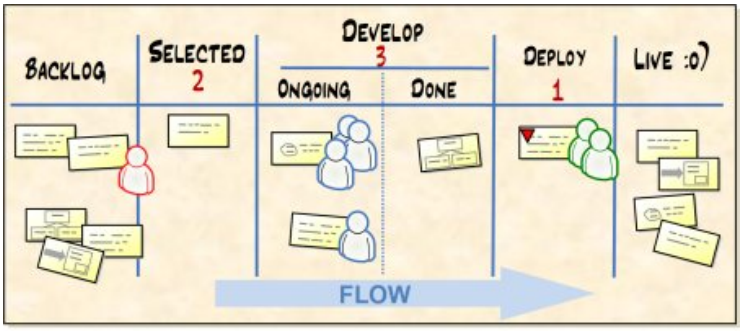


Figure 2.5 – Example kanban board (Kniberg and Skarin, 2010).

each workflow state" (Kniberg and Skarin, 2010, p. 4). By setting WiP limits, bottlenecks as well as idle time within the workflow can be identified and visualized on the kanban board. Furthermore, one has to actively work on solutions for staying within WiP limits and thus try to anticipate future blockages to improve the overall flow rate (i.e. lead time) (Vallon, 2011). The usage of WiP limits is demonstrated in Figure 2.5.

Measure the Lead Time: The *lead time* (also *cycle time*) is the average time to complete one work item (Kniberg and Skarin, 2010), i.e. it is the time that is needed for one item to complete all steps of the workflow. It is the most important metric in kanban, because the goal is "to optimize the process to make lead time as small and predictable as possible" (Kniberg and Skarin, 2010, p. 5). Figure 2.5 illustrates the application of all three rules by means of a kanban board that visualizes the following simple workflow:

$$Backlog \Rightarrow Selected \Rightarrow Develop\,(Ongoing/Done) \Rightarrow Deploy \Rightarrow Live$$

Each work item (illustrated as kanban cards) runs through the whole workflow from *Backlog* to *Live* in a certain time. The average time of all completed kanban cards, i.e. work items, is the lead time. The numbers in Figure 2.5 are WiP limits. The number "2" in the *Selected* column denotes that only two work items at a time may be pulled into that step of the workflow. In case of *Develop* we have a shared column and thus shared WiP limit of "3" for *Ongoing* and *Done*. This forces developers to care for deployment of developed items because otherwise they cannot start with new items due to the WiP limit. Thus deployment on a regular basis comes naturally, which usually helps to improve the overall quality of the product.

Roles, Meetings and Artifacts

Kanban does not specify any roles, meetings or document types. This does not mean that kanban works without any roles but that it remains free to add whatever roles seem to fit. However, "the general mindset in both scrum and

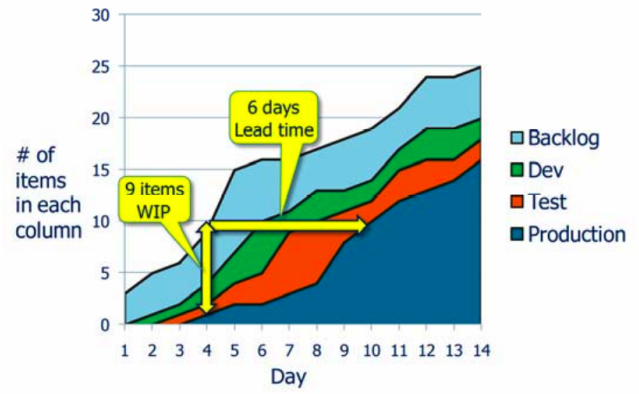


Figure 2.6 – Kanban cumulative flow diagram (Kniberg and Skarin, 2010).

kanban is 'less is more'. So when in doubt, start with less." (Kniberg and Skarin, 2010, p. 11). Scrum's daily stand up meeting may also be found in kanban teams, although it is not obligatory (Vallon, 2011).

One instrument that deserves to be mentioned is the *cumulative flow diagram* (CFD). It is also not prescribed by kanban, but can help to present the correlation of WiP limits and lead time. An example is provided in Figure 2.6. The horizontal yellow arrow in Figure 2.6 shows the amount of time that the work item needed in each phase of the workflow, i.e. each column of the kanban board, until it reached production. The vertical yellow arrow in Figure 2.6 shows the number of work items in each phase. Thus the correlation of WiP and lead time can be identified at any given moment in time using the CFD, which may help to find the right WiP limits in order to reduce lead time.

The only artifact that really needs to be part of every kanban process is the kanban board, because it is its very central component.

2.2 Distributed Software Development and Agile Practices

This section first covers background on DSD in general and then concludes with the rationale behind adding agile practices to DSD in Section 2.2.4. The coverage is very compact, as the systematic mapping study of Chapter 4 provides an extensive overview of the application of agile practices in DSD in related work over the 15-year-period of 1999-2014.

2.2.1 Distributed Software Development

Distributed software development (DSD) has become a daily reality in today's software engineering (VersionOne, 2014) and has been an evolving research field for more than a decade (Carmel, 1999). In 2006 a conference was premiered to focus exclusively on the subject: *IEEE International Conference on Global Software Engineering* (ICGSE). Global software engineering deals with all kinds of geographically distributed software development teams, not only globally distributed ones, as defined by Šmite et al.:

> "Global software engineering: development of a software artifact across more than one location" (Šmite et al., 2014, p. 122)

Hence the terms *DSD* (distributed software development) and *GSD* (global software development) can be used interchangeably. This thesis uses the term DSD exclusively.

2.2.2 Benefits

Benefits include cost savings (sometimes actively asked for by the client (Klimpke et al., 2011)), access to large multi-skilled work forces, reduced time to market (Ó Conchúir et al., 2009) and the possibility to follow critical-path tasks around the clock (Herbsleb and Moitra, 2001) as well as the possibility to locate the development closer to the customer (Damian and Moitra, 2006) and also higher flexibility (Klimpke et al., 2011).

2.2.3 Challenges

Several challenges emerge in developing software with distributed teams in a multi-site environment. The flow of information across sites is more restricted and there is a lack of informal conversation (Herbsleb and Mockus, 2003) and group chat and instant messaging may not be for everybody (Herbsleb et al., 2002). Further issues include physical distances and time zones, loss of "teamness", culture differences (Battin et al., 2001), strategic issues, process differences, knowledge management and technical ones (Sengupta et al., 2006). Ågerfalk et al. (2005) have presented a framework for clustering DSD issues in *communication*, *coordination* and *control* in the three dimensions of *temporal*, *geographical* and *socio-cultural distance*. Mitigation strategies include reducing intensive collaboration, cultural and temporal distance (Carmel and Agarwal, 2001) and also the application of agile practices (cf. Section 2.2.4).

2.2.4 Why Agile DSD?

Section 2.2.3 showed that working in distributed development environments exhibits several challenges. Figure 2.7 lists DSD challenges and shows, how they can possibly be mitigated with the help of agile practices.

Challenges in Distributed Development [2,3,5]		Characteristics of Agile Development [1,4]	New Challenges in Agile Distributed Development
Communication challenges	• Difficult to initiate communication • Misunderstanding/miscommunication • Dramatically decreased frequency of communication • Increased communication cost—time, money, and staff • Time difference	• Lack of formal communication • Increased demand for informal communication	*Communication need vs. communication impedance*
Lack of control	• Difficult to control process and quality across distributed teams	• Lightweight process • Ongoing negotiation • Reliance on skilled people	*Fixed vs. evolving quality requirements* *People vs. process oriented control*
Lack of trust	• Lack of trust between distributed team members • Lack of team morale	• Cohesive team • Trust built progressively • Short commitment	*Formal vs. informal agreement* *Lack of team cohesion*

Figure 2.7 – Characteristics of DSD and agile practices compared (Ramesh et al., 2006).

In fact, several authors have pointed out that agile practices potentially mitigate the challenges faced in DSD environments. Since 2004 we can see an increasing research interest in applying agile practices to DSD (Jalali and Wohlin, 2010). Schwaber (2004) presented mechanisms of scaling scrum, which also touched geographically distributed environments. Early studies include e.g. Ramesh et al. (2006), who conducted a multiple-case study and concluded that "careful incorporation of agility in distributed software development environments is essential in addressing several challenges to communication, control, and trust across distributed teams" (Ramesh et al., 2006, p. 46). Paasivaara and Lassenius (2006) argue that the strong emphasis on frequent communication can also in DSD be regarded as a strength and that "the short iterations, frequent builds, and continuous integration [...] bring transparency of work progress to all partners" (Paasivaara and Lassenius, 2006, p. 112). A notable finding by Hossain et al. (2009) was that scrum, the most popular agile process, can be used to mitigate DSD risks, but it needs to be extended. In later work, Hossain et al. (2011b) also investigated in their multiple-case study how scrum practices have been successfully applied to distributed environments. There are also several books (Eckstein, 2010; Šmite et al., 2010b; Woodward et al., 2010) focusing exclusively on the application of agile practices in DSD environments.

2.3 Conclusion

This chapter presented the theoretical background of this thesis, namely of agile software development, distributed software development and the combination of the two. In conclusion to this chapter, the first research question of this thesis is answered:

RQ1. Why would distributed software development benefit from agile practices?

Based on the investigation of related work in this chapter, it can be concluded that typical challenges found in distributed software development environments such as communication, coordination and control challenges (cf. Section 2.2.3) can be mitigated by the implementation of agile practices (cf. Section 2.2.4) due to the focus on frequent communication and short iterations, among others, which potentially increases transparency for all partners involved and allows organizations to make better use of the benefits in distributed software development (cf. Section 2.2.2).

CHAPTER 3

Design Theory

The design theory has been finalized during a research visit at Stanford University at the Center for Design Research of Professor Dr. Larry Leifer and has been published in a report for the Austrian Marshall Plan Foundation (Vallon, 2015).

Contents

Before diving into the design theory for the ADAPT framework, the status quo of frameworks in the field is presented. Damian and Zowghi (2003) developed an issue-based model focusing on requirements engineering in distributed environments. Ågerfalk et al. (2005) worked on a framework of distributed development issues (which also appeared in (Ågerfalk and Fitzgerald, 2006)). Hossain et al. (2011a) presented a research framework which maps DSD challenges and mitigation strategies and discusses how scrum practices could be implemented in practice based on a systematic literature review. However, past frameworks are described at a high level of abstraction, giving only exemplary advise on how to implement the practices. The authors also conclude that "there is a substantial need for research to 'catch up' and support the needs of practice" (Hossain et al., 2011a, p. 100). The ADAPT framework aims to provide detailed advice on how to implement agile practices successfully based on empirical evidence gathered from a multiple-case study (cf. Chapters 5 and 6).

Section 3.1 investigates suitable design research theory to achieve that goal. Section 3.2 looks at how related papers have dealt with framework design and development and derives essential aspects for the creation of the ADAPT framework. The process for implementing the ADAPT framework is drafted in Section 3.3. Section 3.4 concludes the report by presenting the design components based on the former investigation to tackle the research question:

RQ2. What are suitable design components for building a distributed agile process framework?

3.1 Design Research

Developing the ADAPT framework is a constructivist approach in the sense that the developed artifact is the chief output to the research (Gregor and Jones, 2007). According to Gregg et al. (2001), the ADAPT framework would be classified as an incremental extension and/or generalization of an existing concept (applying agile to DSD), based on descriptive details (practices and guidelines extracted from multiple-case study) and is without implementation at this stage (subject of future work, cf. Section 7.7). The design theory is described with the components developed by Gregor and Jones (2007), which build on top of Aristotle's writing on the four explanations of any *thing* explanation (Falcon, 2014) (literal translation from Greek, see (Hooker, 1996)):

- The material cause: "that out of which", e.g., the bronze of a statue.
- The formal cause: "the form", "the account of what-it-is-to-be", e.g., the shape of a statue.
- The efficient cause: "the primary source of the change or rest", e.g., the artisan, the art of bronze-casting the statue, the man who gives advice, the father of the child.
- The final cause: "the end, that for the sake of which a thing is done", e.g., health is the end of walking, losing weight, purging, drugs, and surgical tools.

The four causes apply "to everything that requires an explanation, including artistic production and human action" (Falcon, 2014). Gregor and Jones (2007) expand on the four causes and define eight components as essential to the anatomy of design theory (six core and two additional ones, cf. Table 3.1).

Component	Description
Core components	
1) Purpose and scope (the causa finalis)	"What the system is for", the set of meta-requirements or goals that specifies the type of artifact to which the theory applies and in conjunction also defines the scope, or boundaries, of the theory.
2) Constructs (the causa materialis)	Representations of the entities of interest in the theory.
3) Principle of form and function (the causa formalis)	The abstract "blueprint" or architecture that describes an IS artifact, either product or method/intervention.
4) Artifact mutability	The changes in state of the artifact anticipated in the theory, that is, what degree of artifact change is encompassed by the theory.
5) Testable propositions	Truth statements about the design theory.
6) Justificatory knowledge	The underlying knowledge or theory from the natural or social or design sciences that gives a basis and explanation for the design (kernel theories).
Additional components	
7) Principles of implementation (the causa efficiens)	A description of processes for implementing the theory (either product or method) in specific contexts.
8) Expository instantiation	A physical implementation of the artifact that can assist in representing the theory both as an expository device and for purposes of testing.

Table 3.1 – The eight components of design theory as defined by Gregor and Jones (2007).

3.2 Framework Design

In the introduction of this chapter known frameworks to combine agile and DSD have been discussed with the result that there is only one in (Hossain et al., 2011a), which lacks design methodology, detailed and ready-to-use insight. This section investigates related design research on building frameworks in general to deduct knowledge for building the ADAPT framework.

The term *framework* within this thesis is understood as defined by Wild et al.:

> "A framework can be seen to be a general set of concepts [...]. It is not tightly organized enough to be a predictive theory. It aims to sketch out the general concepts of a field of enquiry & the possible relationships between them." (Wild et al., 2009, p. 147)

More specifically, the ADAPT framework is a *process framework*, understood within this thesis as defined by Sorathia et al.:

> "[...] it integrates various elements involved in different phases of the software development life-cycle. Once the process is well defined, the individual teams can utilize required process subsets or

the entire process and also may customize these to meet individual requirements." (Sorathia et al., 2010, p. 297)

Table 3.2 illustrates that the ADAPT framework is designed to support a process instantiation (Gregor and Jones, 2007), i.e. to allow the practitioner to derive a concrete process implementation for his specific DSD environment based on and guided by the information provided within the ADAPT framework.

	ADAPT Framework	**Process Instantiation**
Artifact type	Abstract artifact	Material artifact (instantiation)
Description	A framework for driving agile DSD process implementations including challenges, guidelines and practices	The concrete instantiated process implementation, led by ADAPT's guidelines and utilizing a subset of the ADAPT's practices

Table 3.2 – ADAPT framework vs. a concrete process instantiation (inspired by (Gregor and Jones, 2007)).

The framework consists of challenges (cf. Section 3.2.2), guidelines (cf. Section 3.2.3) and effective practices (cf. Section 3.2.4). The ADAPT framework is by design similar to (Soundararajan et al., 2012) in the way that it uses a three-layered setup and links principles (ADAPT uses guidelines) to practices. The framework is based on empirical evidence only. As such it provides an overview of what worked in which distribution scenarios in a description-oriented fashion (in contrast to being prescription-oriented) (Van Aken, 2005). Future research may include the development of prescription-driven practices (cf. Section 7.7).

Figure 3.1 shows the schematic outline of the ADAPT framework. The three challenge types (cf. Section 3.2.2) are linked to several guidelines, which in turn cover several practices that help accomplish a guideline's objective and thus mitigate the challenges.

Grounded theory, more specifically a combination of *open coding* and *axial coding* by Strauss et al. (1998), is used for theory building from the case study research in this thesis. Conceptual practices and guidelines are derived from coding the three case studies (cf. Chapters 5 and 6) to ensure empirical grounding of all elements within the ADAPT framework. The guidelines and practices are only considered for the ADAPT framework if they have empirical evidence in at least two of the analyzed three cases. Additionally further support from related empirical studies is sought (cf. Chapter 4) and discussed (cf. Chapter 7) to strengthen the emerging theory. The framework is thus designed iteratively: preliminary *concepts*, i.e. guidelines and practices, are extracted after each respective case analysis and then aggregated cross-case to find emerging patterns. The design and results are evaluated in two focus

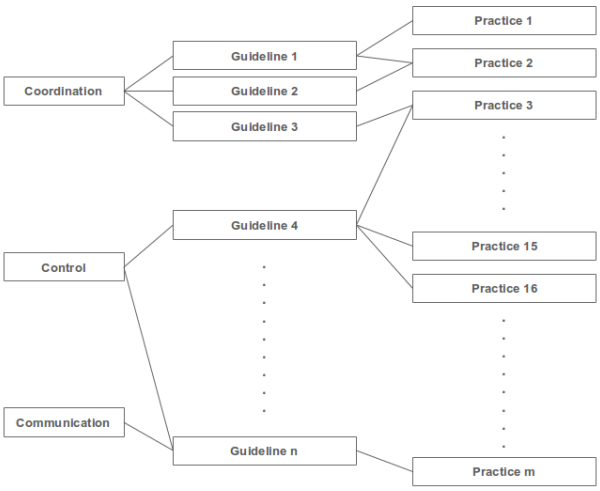


Figure 3.1 – Schematic outline of the ADAPT framework: challenges, guidelines and practices.

groups and 10 expert interviews (cf. Chapter 7). The in-depth explanation of the used approach to theory building is presented in the research design of the multiple-case study in Section 5.1.

3.2.1 Software Process Tailoring

Research efforts in software process tailoring go back to the 1980s (Akbar et al., 2011b), but it is still a topic that has not been extensively researched (Martínez-Ruiz et al., 2012). It is a necessity in both traditional processes, e.g. Rational Unified Process (Hanssen et al., 2005), and agile ones, e.g. XP (Mirakhorli et al., 2008) or scrum (Kniberg, 2007; Kniberg and Skarin, 2010). Pedreira et al. (2007) distinguish between formal and informal approaches to software process tailoring and argue that formal approaches may be better for large organizations with a planned and strictly managed process, while small and medium-sized organizations may benefit from a simple and pragmatic process. The informal process tailoring approach suits lightweight agile thinking better and is part of the process tailoring using the ADAPT framework. Software process tailoring can be done at different levels such as e.g. the organizational and project level (Pedreira et al., 2007). Although it is acknowledged that context consists of both organizational and project-based parts (Xu and Ramesh, 2003), the ADAPT framework will focus on project-based tailoring due to the argument that each project is unique even within the same organization.

Applying agile practices to DSD is no silver bullet solution, the process implementation has to be tailored iteratively and correctly to the individual

project's needs. Failure to do so will not produce better results, as Alqahtani et al. (2013) showed: 75% of the studies report a lack of communication and collaboration in agile DSD. Dumitriu et al. (2011) argue that DSD and agile software development are two extremes (distribution and collocation) that are not easy to integrate, so tailoring must be seen as finding an optimal compromise between the two in order to allow agile practices to reduce the consequences of geographical, temporal and socio-cultural distance. It is also very important that the rationale behind the practice is understood for a successful process tailoring (Šmite et al., 2010b).

3.2.2 DSD Challenge Categories

There have been different categorizations of challenges in distributed software development in related work such as (Kajko-Mattsson et al., 2010; Mudumba and Lee, 2010; Sriram and Mathew, 2012). The ADAPT framework follows the most established approach in the field of DSD to classify challenges in categories *coordination*, *control* and *communication* (Carmel, 1999; Carmel and Agarwal, 2001; Ågerfalk et al., 2005; Ågerfalk and Fitzgerald, 2006; Holmström et al., 2006; Pries-Heje and Pries-Heje, 2011; Hossain et al., 2011a). The categories are described as follows.

> "*Coordination* is the act of integrating each task with each organizational unit, so the unit contributes to the overall objective. Orchestrating the integration often requires intense and ongoing communication.
>
> *Control* is the process of adhering to goals, policies, standards, or quality levels. Controls can be formal (such as budgets and explicit guidelines) or informal (such as peer pressure). We recognize today that, for knowledge workers, coordination and control have in many ways blended together.
>
> *Communication* is a mediating factor affecting both coordination and control. It is the exchange of complete and unambiguous information-that is, the sender and receiver can reach a common understanding." (Carmel and Agarwal, 2001, p. 23)

Carmel and Agarwal (2001, p. 23) state that "coordination and control have in many ways blended together". Hence, for disambiguation of the two terms within this thesis, further elaboration is required: coordination and control can be seen as two sides of the same coin (Nurmi et al., 2005), which is the process of managing dependencies among activities (Malone and Crowston, 1994). The extreme of each side would be *organic coordination* (cooperative, informal and decentralized) and *mechanistic control* (controlling, formal and

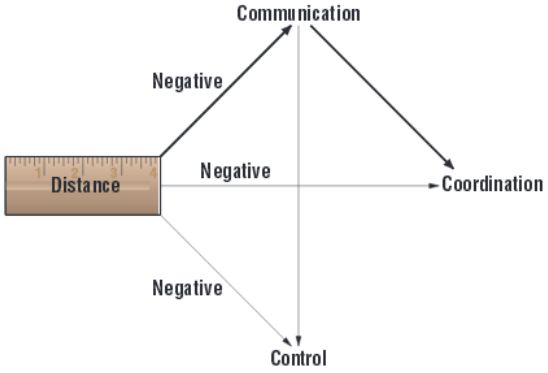


Figure 3.2 – Challenge categories by Carmel and Agarwal (2001): Impacts of distance in distributed software development.

centralized) (McCann and Galbraith, 1981). Another important distinction is that coordination is the work of dependent parts towards a common goal (Nurmi et al., 2005), while control is needed when the goals of individual stakeholders differ from those of the larger overall entity (Sabherwal, 2003). Figure 3.2 shows the original draft of the challenge types by Carmel and Agarwal (2001). It illustrates well that communication is a mediating factor for both coordination and control.

These three challenge types are the top layer in the ADAPT framework (cf. Figure 3.1), overspanning guidelines (cf. Section 3.2.3) and practices (cf. Section 3.2.4). Pries-Heje and Pries-Heje (2011) and Hossain et al. (2011a) have both worked with the CCC model (coordination, control, communication) in distributed agile environments, which underlines its applicability and relevance to the ADAPT framework.

3.2.3 Design Guidelines

Design guidelines are the second layer in the ADAPT framework (cf. Figure 3.1) and overarch the practices. Similar to (Soundararajan et al., 2012), the relationship between guidelines and practices is *N to M*, which means that a guideline can be linked to multiple practices and vice versa. The guidelines are treated as *constructive heuristics* (Heeager and Rose, 2014) and emerge from case study research. The design guidelines specifically aim at guiding the practitioner to build and improve his individual process instantiation.

3.2.4 Effective Practices based on Context

The goal of this framework is to provide effective practices that have a successful empirical grounding. The identified practices will not be called *best*

practices as no practice can be "best" in every context (Ambler, 2011). Inspired by (Ambler, 2002), the practices to be identified are called *effective practices*. They are regarded as *effective* because they rely on successful empirical implementations as evidence and are thus seen to meet a goal with higher probability and fewer risks involved (Schatten et al., 2010). The practices can be regarded as *method fragments* (Baskerville and Pries-Heje, 2013). It is a major concern that the practices are detailed enough and not too simple in their description in order to be of practical usability (Baskerville and Pries-Heje, 2013).

It seems to be agreed within the research community that context is a major concern for every type of case study research, although different ways of reporting context have been proposed (Kitchenham et al., 1995; Runeson and Höst, 2009; Petersen and Wohlin, 2009; Jalali and Wohlin, 2010, 2012a). This thesis follows and expands on the checklist for reporting context by Jalali and Wohlin (2010, 2012a) as it has already been applied in an extensive systematic review of empirical studies. Practices within the ADAPT framework will be reported including their context of application. This criterion leads to the constraint that no theoretical practices will be part of the ADAPT framework, a successful empirical application is the minimum requirement (sine qua non) for being considered for inclusion in the framework. A practice is the smallest element of the ADAPT framework in the hierarchy of challenges, guidelines and practices. The focus is on providing effective and tangible (i.e. detailed) practices to be used by the practitioner. The practices may evolve into a *pattern language* (Alexander et al., 1977) in future work based on the ADAPT framework, once a significant amount of good empirical research (including a rich description of the study context and background) has been done on the subject.

The practices are sought to emerge from varying distribution scenarios (Prikladnicki et al., 2003), which are:

- Cross Town Scenario (case 1 of the multi-case study, cf. Section 5.2)
- No Time Shift Scenario (case 2 of the multi-case study, cf. Section 5.3)
- Continental Scenario (case 3 of the multi-case study, cf. Section 5.4)
- Global Scenario (out of scope for this thesis)

As has been stated in Section 1.2, the global scenario is out of scope for this thesis, but the framework shall be designed to allow future extension for this scenario. The rationale behind this decision is that the global scenario adds another layer of complexity to the research problem with massive distance, time and socio-cultural differences between the development sites. Following

agile values, the ADAPT framework will evolve iteratively, with this thesis describing the first full iteration of the framework. The global scenario may be added in a future iteration (cf. Section 7.7).

3.3 Process Design

Process design (Aken, 2004) is necessary because "professionals need to know how to apply the knowledge in their own unique and specific cases" (Gregor and Jones, 2007, p. 322). The practices presented in the ADAPT framework can be consumed in a "supermarket approach" (Baskerville and Pries-Heje, 2013), i.e. it is advised to implement a minimal set to satisfy all guidelines, but the framework users, i.e. the practitioners, decide which of the practices to select. The term *process design* within this thesis is regarded as the way of arriving from the general ADAPT framework at the concrete process instantiation. It is an iterative process, where practices should be evaluated and then modified or replaced in regular retrospective-type meetings after each sprint. In order to maintain the self-organization of teams, the decision on what practices to select should be a majority vote (bottom up) rather than a (top down) management decision to achieve a better level of acceptance and motivation to change. Figure 3.3 illustrates the proposed iterative agile process design:

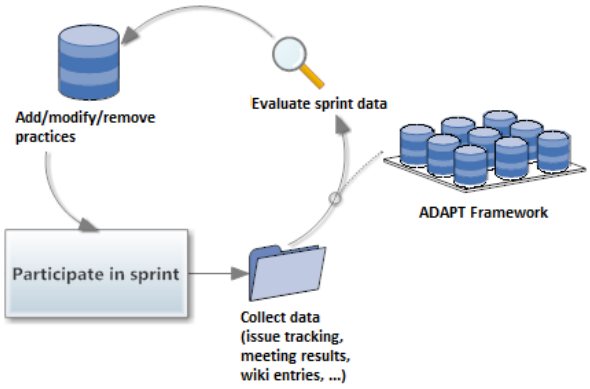


Figure 3.3 – Using the ADAPT framework for process design.

1. Refer to the ADAPT guidelines to find suitable (linked) practices to add for the next sprint such that all guidelines are covered (i.e. at least one practice per guideline is part of the process implementation[2]).

2. Conduct the sprint iteration and collect data to evaluate the practices[3] by means of e.g. burndown charts, bug counts or lead time.

[2]It is good agile practice to start with less and add more later in the following sprints.
[3]Naturally this would be a typical task for the scrum master or an agile coach, if available.

3. Evaluate data and reflect on the process implementation as a team effort (e.g. during retrospective): start over with the first step but now consider not only adding, but also modifying or removing practices while trying to find a balance between the CCC challenge types coordination, control and communication.

3.4 Design Components

This final section builds on top of the prior discussion in this chapter and defines the design components in Table 3.3 to complete the presentation of the design theory for the development of the ADAPT framework within this thesis. Five testable propositions (TP) are also part of the design components and are described in Table 3.4. Each TP features a rationale and a means of verification, as each TP is regarded a truth statement according to the definition by Gregor and Jones (2007). The objective of the *verification* in this context is understood as in software testing to verify if the "product" (in this case the ADAPT framework) is built right according to this chapter's design theory, in contrast to *validation*, i.e. if it is the right product (Boehm, 1989), which is also partly covered due to the extensive research of related work (cf. Chapter 4) and the fact that some of the interviewed experts (cf. Chapter 7) are practitioners and thus possible future end users. This definition is also in line with the IEEE Standard 610.12-1990 (IEEE, 1990, p. 81), stating that verification is the "process of evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase". So the verification column of Table 3.4 describes the intended means of verification to verify if this chapter's up-front design theory is in fact implemented in the first full iteration of the ADAPT framework as the outcome of this thesis. The discussion of testable propositions is presented in Chapter 7.

3.5 Conclusion

This chapter defined the design theory behind the construction of the ADAPT framework and answers:

RQ2. What are suitable design components for building a distributed agile process framework?.

Component	Description
1) Purpose and scope	The aim is to develop a framework for applying agile practices effectively to distributed software development.
2) Constructs	The framework is represented by a three-layered hierarchy of challenge types (CCC), design guidelines and effective practices.
3) Principles of form and function	A process framework is provided to aid practitioners and researchers in tailoring agile practices to the respective unique distributed environment.
4) Artifact mutability	The design process supports continuous construction cycles, allowing the practitioners to add, modify or remove practices as the project (and thus empirical feedback) progresses. The framework is designed to be open for integrating additional effective practices and guidelines as the research field evolves over time, possibly introducing a more prescriptive nature in future work, that allows the addition of the out-scoped global scenario. Hence, it is regarded as a learning framework, designed with future iterations in mind.
5) Testable propositions	The ADAPT framework should satisfy the five propositions TP1 to TP5, which are presented in Table 3.4.
6) Justificatory knowledge	The framework is grounded in current research on agile software development and distributed software development. The design theory has been presented and improved during a four-month research visit at the Center for Design Research at Stanford University and is evaluated in two focus groups and ten expert interviews (cf. Chapter 7).
7) Principles of implementation	The process design (how to arrive from the generic ADAPT framework at the concrete process instantiation) is an iterative process utilizing agile feedback loops at the process level and is described in detail in Section 3.3.
8) Expository instantiation	The printed Table 6.3 as well as the practice cards in Section 7.7 serve as a physical implementation of the artifact to represent the theory and can be used as an expository device and for purposes of testing. However, actual real-world testing (an implementation study) is out of scope for this thesis and planned for future work.

Table 3.3 – The eight design components of the ADAPT framework's design theory.

Propositions	Rationale	Verification
TP1 (empiric). Each practice of the ADAPT framework is grounded in empirical evidence.	The ADAPT framework is not a silver bullet solution, but it is a set of tools based on empirical evidence showing what worked in which context.	The empirical context of all practices and guidelines must be fully specified according to the checklist defined in Chapter 4.
TP2 (iterative process tailoring). The ADAPT framework allows a simple, pragmatic and iterative process tailoring (rather than planned and strictly managed).	Process tailoring should be part of any agile implementation.	Evaluation of the process design (cf. Section 3.3) is conducted through two focus groups and ten expert interviews.
TP3 (accessible to different scenarios). The ADAPT framework supports project-based process tailoring (rather than organization-based).	Even within the same organization each project is unique.	The multiple-case study needs to feature different distribution scenarios (outscoped: global scenario) and thus different project-based environments.
TP4 (tangible). The ADAPT framework provides tangible and detailed advice to the practitioner.	In order to be of practical use the practices must provide enough detail.	Evaluation is conducted through two focus groups and ten expert interviews.
TP5 (easily extensible). The ADAPT framework is easily extensible.	In order for the ADAPT framework to further evolve and improve, practices and guidelines need to be extensible.	Evaluation is conducted through two focus groups and ten expert interviews. At least one possible way of expanding the framework must be scheduled for future work.

Table 3.4 – Testable propositions for the design theory of the ADAPT framework.

Table 3.3 shows the design components *purpose and scope*, *constructs*, *principles of form and function*, *artifact mutability*, *testable propositions*, *justificatory knowledge*, *principles of implementation* and *expository instantiation*. At the heart of it five testable propositions have been derived (cf. Table 3.4) covering the following attributes of the ADAPT learning framework: empiric (TP1), iterative process tailoring (TP2), accessible to different distribution scenarios (TP3), tangible (TP4) and easily extensible (TP5).

CHAPTER 4

Agile Practices in DSD: Systematic Mapping of Fifteen Years

At the time of finalizing this dissertation, this chapter's work was in preparation for publication (to appear, cf. Section 1.5).

Contents

This chapter presents the results of a systematic mapping study on the use of agile practices in distributed software development (DSD) over the fifteen year period of 1999-2014. The mapping study is built on top of research by Jalali and Wohlin (2010, 2012a), who investigated studies published between 1999 and 2009 and their analysis is extended to years 2010-2014 in this chapter and thus also following the call of Hanssen et al. (2011) who identified the need for systematic reviews covering agile DSD for years 2008 and newer in their tertiary study.

Up-to-date maps and trends are provided with regard to the research previously conducted. A special focus is put on comparing the progress in the five-year period of 2010 to 2014 to the results of Jalali and Wohlin (2010,

2012a). This approach allows covering a full fifteen year time span and investigates how agile practices may or may not have evolved in DSD environments. The systematic search led to 95 included relevant studies (cf. Appendix A.2) for the extended analysis to years 2010-2014, for which a full-text analysis is conducted.

The remainder of the chapter is organized as follows. Section 4.1 provides an overview of related systematic mapping studies. Section 4.2 explains the detailed study design and procedure. Section 4.3 presents the results of the systematic mapping, also in comparison with previous research by Jalali and Wohlin (2010, 2012a). A summary is presented in Section 4.3.4 and Section 4.4 discusses implications for research and practice and Section 4.5 provides the conclusion.

4.1 Related Systematic Literature Reviews and Mappings

VersionOne publishes a *state of agile* survey each year, the latest 2014 edition (VersionOne, 2014) shows that the usage of distributed agile teams has more than doubled from 2012 to 2014: while 2012 35% of the respondents reported to work in distributed agile teams, in 2014 the number increased to 80%. This shows that the use of agile practices in DSD is a very recent, relevant and fast-evolving challenge to both the researcher and the practitioner and deserves further research attention. This section explores the history of systematic mappings and literature reviews in software engineering (SE), previous studies on DSD and on agile practices in DSD in particular.

4.1.1 General Remarks on Systematic Mapping and Literature Reviews in SE

When a research area reaches a certain level of maturity it becomes more important to summarize the growing amount of past findings and provide overviews (Petersen et al., 2008). Systematic literature reviews (SLR) have gained attention earlier than systematic mapping studies in SE research, such as in (Kitchenham and Charters, 2007; Dybå et al., 2006; Hannay et al., 2007; Kampenes et al., 2007), while systematic mappings had been widely neglected until (Bailey et al., 2007). Both systematic mappings and SLR aim to aggregate knowledge from previous research. Systematic mapping studies provide a coarse-grained overview. Full-text analysis is not required; the mapping can be done based on abstracts or by also studying further parts of the full text such as introduction and conclusion (Petersen et al., 2008). However, for this systematic mapping study full-text analysis was necessary to extract all of the agile practices.

4.1.2 Agile Practices in DSD

Distributed software development is a growing research field as several systematic reviews account for (Verner et al., 2012; Marques et al., 2012; Raza et al., 2013) and all seem to agree that there is a need for more primary studies in DSD research. Most relevant to this thesis' line of research are systematic mapping and review studies focusing on the application of agile practices in DSD, which is covered in this section.

Jiménez et al. (2009) conducted an SLR on challenges and improvements in distributed software development. Agile methodologies were identified as one success factor in DSD. However, agile is neither a focus of the study nor are any concrete practices listed. Hossain et al. (2009) conducted an SLR on scrum and DSD. To the best of the author's knowledge it is the first SLR addressing agile practices in DSD, although the focus is limited solely to scrum. Hossain et al. (2011a) also formulated a research framework for scrum in DSD. Jalali and Wohlin were the first to deliver an extensive overview of agile DSD by conducting a systematic mapping study (Jalali and Wohlin, 2010) and a systematic review (Jalali and Wohlin, 2012a). Both studies cover years 1999-2009 and serve as the base to be extended for recent years in this thesis' systematic mapping study. Sriram and Mathew (2012) give an overview of agile and DSD. However, the short paper does not explain the research design nor provide details about the execution of the study. Hence it does not classify as a systematic review. Kuhrmann et al. (2013) researched the use of agile artifacts in DSD and conducted a systematic mapping study. It can be seen that XP and scrum artifacts have been almost exclusively used in DSD within the studies analyzed by the authors, followed by Agile Unified Process and Kanban. There is also an agile-related tertiary study by Hanssen et al. (2011) summarizing twelve SLRs in DSD by looking through an agile lens. The authors conclude that agile is a frequent topic in DSD, but many publications lack proper research design and rather have the character of industrial reports. Furthermore the authors call for a new SLR for agile in DSD to cover publications of 2008 and newer, which is addressed in this thesis' systematic mapping study. One of the newer published studies in the area of agile DSD is by Yin and Ma (2013), who conducted a thematic review. However, the short paper does either not follow a systematic approach or it is not explained in the paper. It also does not list referenced studies, but rather summarizes trends without referencing specific studies. The newest systematic review on the matter is, to best of the author's knowledge, by Rizvi et al. (2015) and covers literature until 2012. However, in contrast to this thesis' systematic mapping study, it does not focus on actual agile practices.

In conclusion, there have been a number of studies dealing with reviewing agile methods in DSD. However, *systematic* mapping and literature review

studies specific to the application of agile practices and DSD are limited to only (Hossain et al., 2009) and (Jalali and Wohlin, 2010, 2012a), which only cover literature up to the year of 2009. Hence the identified research gap is to investigate the drastically increasing usage of agile practices in distributed environments (VersionOne, 2014) in the last years and research is built on top of (Jalali and Wohlin, 2010, 2012a) to be able to provide a mapping of a fifteen-year period, effectively covering agile back to its origins. Implicitly, the call of Hanssen et al. (2011) is answered for the need to study years of 2008 and newer in the area.

4.2 Study Design

Since this thesis' systematic mapping study expands on (Jalali and Wohlin, 2010, 2012a), a similar research design is followed. The detailed design including minor changes to the original one of Jalali and Wohlin (2010, 2012a) is presented in the remainder of this section. The guidelines for systematic mapping study design by Petersen et al. (2008) and Kitchenham and Charters (2007) were consulted as well as experiences drawn from (da Silva et al., 2010) for study design, in which two other supporting researchers participated. The supporting researchers assisted in the execution of this thesis' systematic mapping study and the author appreciates their contribution. All work which is not explicitly designated in the following sections as being attributed to the supporting researchers has been conducted solely by the author of this thesis (about 90%).

4.2.1 Research Steps

1. Planning
    - Identification of the need for a systematic mapping (Section 4.1)
    - Specifying the research questions (Section 1.2)
    - Developing a review protocol (Section A.3)
    - Evaluating the review protocol (carried out by supporting researcher)
2. Conducting
    - Identification of primary studies (Sections 4.2.2 and 4.2.3)
    - Selection of primary studies (Sections 4.2.4 and 4.2.5)
    - Study quality assessment (Section 4.2.6)
    - Data extraction (Section 4.2.7)
    - Data synthesis (Section 4.2.7)
3. Reporting

- Formatting the main report
- Specifying publication mechanisms

4.2.2 Search Terms

To achieve comparable results to (Jalali and Wohlin, 2010, 2012a), the same search terms and constraints were used except for the term "open source", which produced a lot of irrelevant results during the pilot search in IEEE Xplore (http://ieeexplore.ieee.org/) and was dropped. The publication year was set to 2010-2014 (effectively 08/2014 as the final search has been conducted that time) and the written language to English. Search was limited to abstract, keywords and title. Although there are some books touching the area such as (Eckstein, 2013; Šmite et al., 2010b; Woodward et al., 2010), books were excluded on all database searches as they are generally not peer-reviewed and also in alignment with (Jalali and Wohlin, 2010, 2012a). The search string basically looks for *agile* AND *distributed software development*, with synonyms or variants of both terms separated by an OR-operator, as follows:

(agile OR scrum OR "extreme programming" OR "pair programming" OR "lean development" OR "lean software development")

AND

("distributed software development" OR "distributed software engineering" OR "global software development" OR "global software engineering" OR gse OR gsd OR "dispersed team" OR "spread team" OR "virtual team" OR offshore OR outsource)

4.2.3 Resources Searched

All databases of (Jalali and Wohlin, 2012a) were searched, which includes one more database compared to (Jalali and Wohlin, 2010):

- ACM Digital Library (http://dl.acm.org/)
- AIS (http://aisel.aisnet.org)
- Compendex (http://www.engineeringvillage.com)
- IEEE Xplore (http://ieeexplore.ieee.org/)
- INSPEC (http://apps.webofknowledge.com)
- Scopus (http://www.scopus.com)

Additionally to the six databases, the most relevant conferences were double-checked manually, *Agile Conference* (AGILE 2010 to 2014), *International Conference on Agile Software Development* (XP 2010 to 2014) and *International Conference on Global Software Engineering* (ICGSE 2010 to 2014) to make sure that nothing is missed from these venues.

4.2.4 Study Selection Criteria

Primary studies were selected according to the following criteria:

1. Inclusion Criteria
    - The study directly relates to the research questions, i.e. addresses agile practices in DSD
    - The study is available via Vienna University of Technology library service, Stanford University library service (both accessible to the author) or is freely available on the web
2. Exclusion Criteria
    - Duplicated or repeated studies
    - Studies not presenting results or work in progress papers
    - Experience reports
    - Books
    - Theses
    - Workshop papers

4.2.5 Study Selection Process

The selection process is illustrated in Figure 4.1. The author searched all six databases with the pre-defined search string and the noted limitations. For databases that allowed exporting the search results in CSV-format, the author wrote a small program to automatically aggregate the findings with the existing spreadsheet base of potentially relevant studies. The program's logic followed the pseudo code in Algorithm 1 and had to be adjusted to the varying format of the CSV-file of the respective database.

With that semi-automatic procedure results of all six databases plus a manual search of conferences AGILE, XP and ICGSE (cf. Section 4.2.3) were aggregated in a single spreadsheet. This final search resulted in 288 entries. In the second step, the author excluded theses, workshop papers and experience reports. It was however unlikely to filter all experience reports at this point of the process, the final decision had to be made after the full-text analysis. The author then classified the papers with regard to their relevance to the

Algorithm 1 Search Helper in Pseudo Code

```
1: for each search entry of the CSV exported from database do
2:     if search entry does not exist in spreadsheet then
3:         add new row for the search entry in spreadsheet
4:     end if
5:     append database-name to database-column of search entry
6: end for
```

research questions based on title, keywords and abstract. The categories used for the classification were *relevant*, *maybe relevant* and *irrelevant*. In the next step one of the supporting researchers double-checked all papers in categories maybe relevant and irrelevant and formed his own decision. An agreement was sought using the following logic described in Table 4.1.

Author's Decision	**Supporting Researcher's Decision**	**Resulting Action**
irrelevant	irrelevant	exclude study
maybe relevant	irrelevant	exclude study
irrelevant	maybe relevant	exclude study
maybe relevant	maybe relevant	include study (for further analysis in step 6)

Table 4.1 – Steps 5 of the six step inclusion process.

The agreement resulted in 152 potentially included studies. The next step was to download all studies through the university library services. The final step, the full-text analysis, resulted in the exclusion of another 57 studies. 90% of the full-text analysis has been conducted by the author and the remaining 10% by a supporting researcher in order to review the final extraction procedure. These 10% have been double-checked by the author to ensure a consistent handling of all studies. Copies of the spreadsheet have been saved after all stages for verification purposes. The final set comprised 95 included studies for the five-year period of 2010 to 2014 (cf. Appendix A.2), which is a greater amount than before in Jalali and Wohlin (2012a) with 81 included studies for the ten-year period of 1999-2009.

4.2.6 Study Quality Assessment Criteria

To ensure quality of studies only peer-reviewed conference papers and journal articles are included, i.e. no books, theses, workshop papers, experience reports and work in progress papers with incomplete results. Some of these criteria, however, could only be fully applied during the full-text analysis.

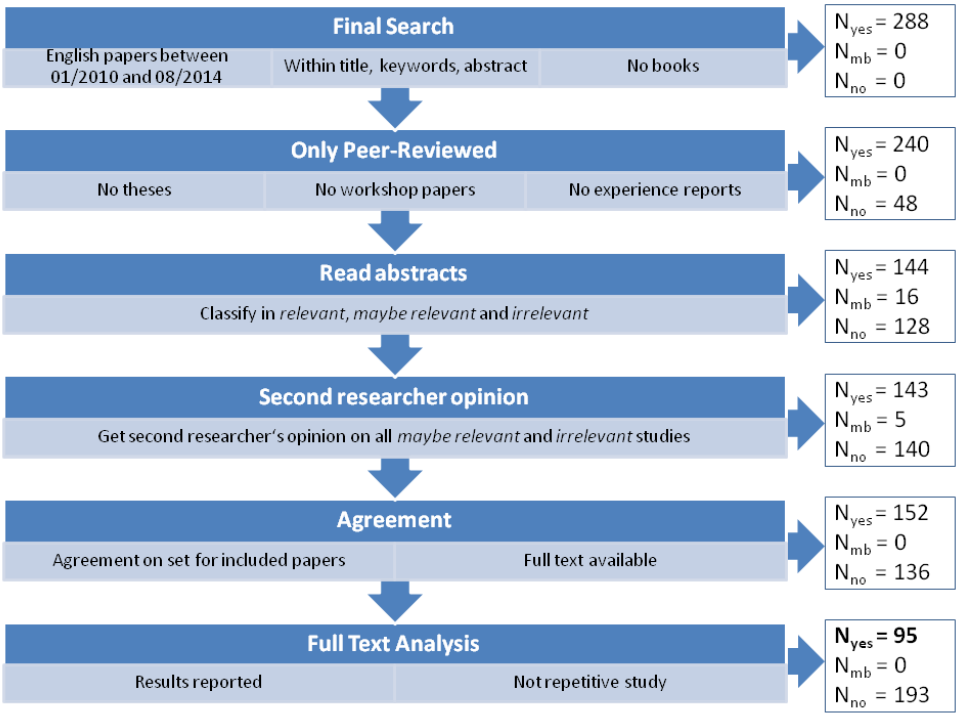


Figure 4.1 – Six step inclusion process: $N_{yes}/N_{mb}/N_{no}$ show the amount of relevant/maybe relevant/irrelevant studies after each respective step.

The classification of included studies was done with regard to the following research types, inspired by Wieringa et al. (2006):

Solution Proposal: Proposal of a novel solution technique without a full-blown validation, but may offer a proof of concept or a small example
Validation Research: Investigation of the properties of a solution that has not yet been implemented in practice, methods may include e.g. experiments, simulation or prototyping
Evaluation Research: Analysis of a problem or the implementation of a technique in practice by means of e.g. a case study, field study or a survey
Philosophical Papers: Sketch of a new way of looking at things, e.g. a new conceptual framework
Opinion Paper: Statement of the author's personal opinion
Experience Paper: Listing of the author's personal experience in an anecdotal way, may also be often written by industry practitioners

Experience reports were excluded from the final set of included studies for further analysis, but still tracked to provide a systematic map of all research types, since they were included in (Jalali and Wohlin, 2012a).

4.2.7 Data Extraction and Synthesis

Data extraction and synthesis was initially done based on title, keywords and abstract and then continued with a full-text analysis for the final set of 95 included studies (cf. Appendix A.2). The Appendix A.3 shows the complete layout of the data extraction spreadsheet. *All* geographically distributed software development teams were included, not only globally distributed ones. This aligns with (Jalali and Wohlin, 2010, 2012a) and also with the definition of global software engineering by Šmite et al. (2014): "development of a software artifact across more than one location". The classification of Jalali and Wohlin (2010, 2012a) was extended for the distribution types, which was inspired by (Šmite et al., 2014) to the following definition:

Location: Offshore (different countries) > Onshore (same country) or Unclear
Legal entity: Outsourcing (different organizations) > Insourcing (same organization) or Unclear
Geographic distance: Far (flight time 2 hours and more) > Near (flight time less than 2 hours) or Unclear
Temporal distance: Large (more than 4 hours) > Small (4 hours or less) or Unclear

If a study falls into several distribution types, e.g. a multi-site environment with two sites involved in onshoring and one site in offshoring, the most complex category will be assigned (as denoted by the ">" sign), i.e. "offshoring" for that example. Šmite et al. (2014) propose to have different notions for geographic and temporal distance regarding offshoring and onshoring. While there is clear benefit to specify contextual information more accurately that way for a specific case, it is considered sufficient for this systematic mapping to use the more coarse-grained view, as defined for offshoring in (Šmite et al., 2014), for *both* offshoring and onshoring.

Furthermore, there is also a distinction between *distributed team* and *virtual team* in the study of Jalali and Wohlin (2010). The distinction made is that distributed teams work on independent tasks while virtual teams work jointly on the same tasks. Since no reference is cited for this definition, support was sought in literature and found in a recognized literature work by Lipnack and Stamps (1997) on virtual teams: "A virtual team, like every team, is a group of people who interact through interdependent tasks guided by common purpose. Unlike conventional teams, a virtual team works across space, time, and organizational boundaries with links strengthened by webs of communication technologies". Hence there is support for the definition of a virtual team, but not for the noted distinction to distributed teams. The terms distributed and virtual were used interchangeably, alongside dispersed, in both the work of

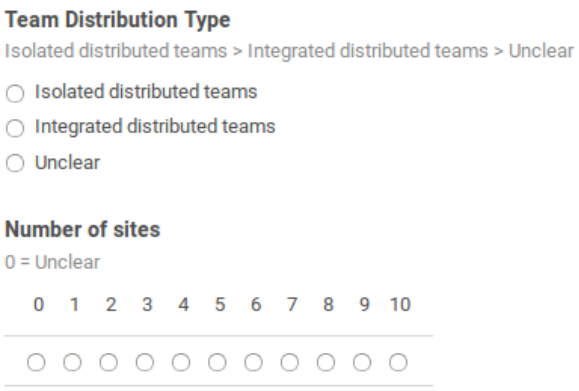


Figure 4.2 – A snippet of the implemented web form that has been used, showing parts of the empirical data extraction for distributed software development.

Lipnack and Stamps (1997) and other related ones (cf. Section 4.1). It is acknowledged that there is a necessary distinction between the two as inspired by Sutherland et al. (2007) and in this systematic mapping they are defined as follows.

Team distribution type: Isolated distributed teams > Integrated distributed teams or Unclear. *Isolated distributed team:* Team members are spread in different locations and work remotely on independent tasks, i.e. one team does not span across more than one site. This is regarded as the corresponding term to Jalali and Wohlin's (2010) *distributed team*. *Integrated distributed team:* Team members are spread in different locations and work jointly on the same tasks, i.e. the team is integrated over multiple (2+) sites. This is regarded as the corresponding term to Jalali and Wohlin's (2010) *virtual team*.

Other than that, similar data was extracted from each paper as Jalali and Wohlin did to be able to compare results: The overall *project size* is also defined as in (Jalali and Wohlin, 2010) with: Large > 50 persons ≥ Medium > 20 persons ≥ Small or Unclear. *Project duration* was defined as: Long (more than 7 months) > Medium > Short (less than 1 month) or Unclear. Like (Jalali and Wohlin, 2010), the knowledge areas are based on SWEBOK (Abran et al., 2004). If a study featured multiple cases, the agile practices have been extracted separately for each case.

The full data extraction scheme is listed in the Appendix A.3. The author implemented a web form via Google forms (https://docs.google.com/forms) which facilitated and improved the data extraction process. Figure 4.2 shows a small sample of the web form implementation.

4.3 Results

The final search based on abstracts, keyword and title led to 152 papers. After the full-text analysis another 57 studies had to be excluded due to the following reasons: replicate study (17), no agile DSD focus (16), study not available (13), experience report (5), no results (2), book (2) and not in English (2). This resulted in 95 included studies for the period of 01/2010 to 08/2014 (cf. Appendix A.2). Without looking at the full text, as many as 57 (37.5%) false positives would have been part of the set of 152 potentially included studies. It is thus argued that even a systematic mapping study is not feasible without doing either a complete full-text analysis or studying the full text adaptively.

4.3.1 Research Settings

Table 4.2 shows the trend of publications for the five-year period 2010-2014 and Figure 4.3 covers the trend in paper count for the total time span of 1999-2014. Compared to years 1999-2009 (Jalali and Wohlin, 2012a), there is an increase in total paper count, which indicates bigger interest in the subject of combining agile with DSD. There is also a trend evolving of around 20 publications on the subject per year, which started in 2008 and seems to continue. This thesis' systematic mapping study ends with 08/2014, so the data point for 2014 cannot be considered accurate, since the remainder of 2014 (09/2014-12/2014) is missing. However, since the most prominent conferences have been searched manually for the full year of 2014 (ICGSE, XP and AGILE conferences), the data point indicates that 2014 indeed had less publications on the subject.

The analysis also included the publication coverage in the respective databases. It is noteworthy that 84.21% of the included studies could be covered using solely the Scopus database and a manual search of ICGSE, XP and AGILE conferences. When additionally extending the search to the Compendex database even 94.74% of included studies can be found. This shows that close to 95% of this thesis' systematic mapping study's included papers could have been found using only two databases, Scopus and Compendex, instead of six. The remaining four (ACM, INSPEC, IEEE and AIS) only added another 5%. Scopus also had the highest count (11) of unique studies which could not be found in any of the other databases. The least performing database regarding the used search terms was AIS, which only resulted in a single unique study, yet in a lot of false positives regarding the term *agile* (not concerning agile SE, but the regular English word "agile" in other contexts).

Figure 4.4 provides an overview of the most active researchers on agile DSD by country and university. The top three countries are USA (13 publications),

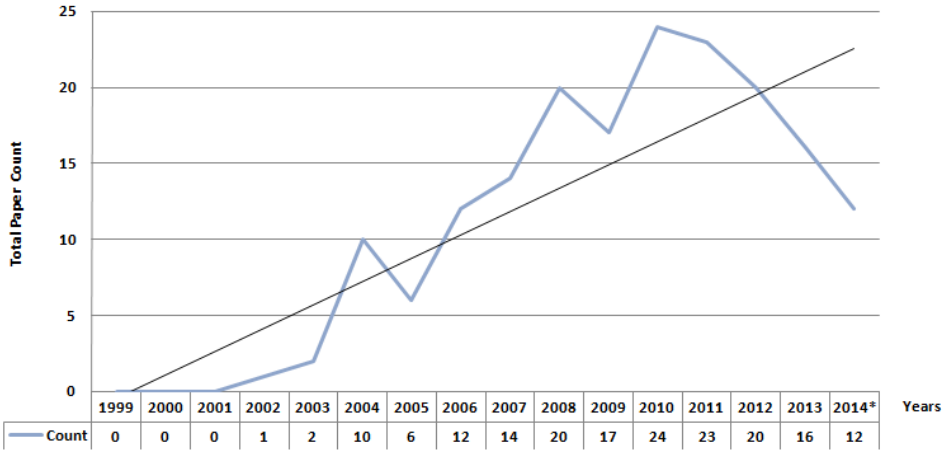


Figure 4.3 – Total paper count and a trend line for agile practices in DSD for the fifteen years of 1999-2014. Data from 1999 to 2009 is from (Jalali and Wohlin, 2012a). Year 2014 does not account for the full year as the search has been conducted in 08/2014 (plus AGILE2014 and ICGSE2014 conferences).

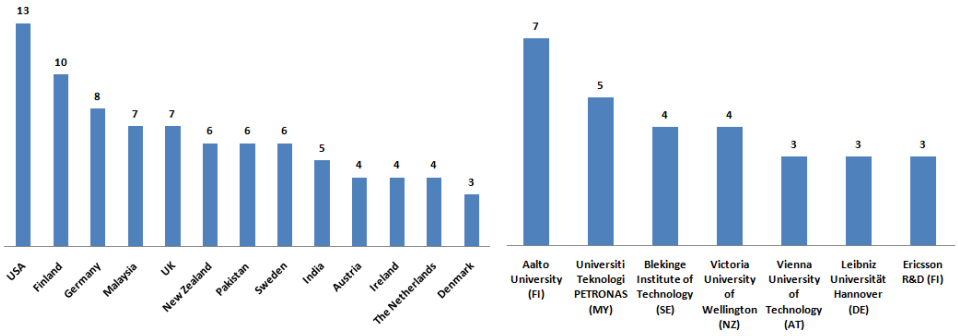


Figure 4.4 – Researchers' affiliations (country and university): Only counts of 3 and more are included in this overview. Only years 2010 to 2014 are covered because this type of mapping was not covered in (Jalali and Wohlin, 2010, 2012a).

Finland (10) and Germany (8). The top three universities are Aalto University (7 publications), University Teknologi PETRONAS (5) and Blekinge Institute of Technology (4). In general, 86.82% of authors in the included studies were affiliated with universities and 13.18% with external research centers or industry, which can be reasoned with the fact that experience reports, which are mostly written by practitioners, were not included in the final set of papers.

Figure 4.5 shows the primary publication targets for agile DSD with ICGSE (16 publications), XP (9) and AGILE (5) conferences taking the lead. Both author affiliations (Figure 4.4) and publication targets (Figure 4.5) have not been analyzed for years 1999-2009 in (Jalali and Wohlin, 2010, 2012a), so no

Databases	2010	2011	2012	2013	2014	Sum
ACM	1					**1**
ACM, IEEE			1			**1**
AIS	1					**1**
Compendex	1		1			**2**
Compendex, IEEE		1	1	1		**3**
Compendex, INSPEC, ACM				2		**2**
Compendex, INSPEC, ACM, IEEE	2					**2**
Compendex, INSPEC, IEEE			1			**1**
Manual Search	4	1		2	4	**11**
INSPEC	1				1	**2**
Scopus	1	4	1	2	3	**11**
Scopus, Compendex	3	3	3	2	1	**12**
Scopus, Compendex, ACM	1	4	1		3	**9**
Scopus, Compendex, ACM, IEEE	1	1	1			**3**
Scopus, Compendex, INSPEC		3	1	1		**5**
Scopus, Compendex, INSPEC, ACM	1	1	1			**3**
Scopus, Compendex, INSPEC, ACM, IEEE	3	5	4	4		**16**
Scopus, Compendex, INSPEC, IEEE	3		4	2		**9**
Scopus, INSPEC	1					**1**
Sum	**24**	**23**	**20**	**16**	**12**	**95**

Table 4.2 – Coverage of included studies by the databases over the studied years 2010-2014. "Compendex, IEEE" e.g. means that the same paper has been found in both Compendex and IEEE databases, while e.g. "Compendex" denotes an exclusive hit in only this respective database.

comparison is offered.

Figure 4.6 shows the mapping of research types over the whole fifteen-year period including data from (Jalali and Wohlin, 2012a). In years 1999 to 2001 there was no publication regarding agile practices in DSD. Compared to 1999-2009 (Jalali and Wohlin, 2010, 2012a) the research field is starting to mature with a shift from experience reports, which was the most frequent paper type by far from 1999 to 2009, to evaluation studies in 2010 to 2014. It should be noted that experience papers were not included in this thesis' systematic mapping study for further analysis in the final set of 95 papers but they were added to Figure 4.6 to achieve comparison. Also, the amount of studies published on the subject in the five-year period of 2010-2014 (95 included papers without experience reports, cf. Appendix A.2) is higher than for the ten-year period of 1999 to 2009 before (81 papers including experience reports (Jalali and Wohlin, 2012a)). This shows that research interest has significantly increased in applying agile practices to DSD.

As Figure 4.7 shows, out of 95 included studies, about half used a qualitative approach and the rest is almost equally split between quantitative, mixed method or a not properly specified methodology. This shows a notable shift to

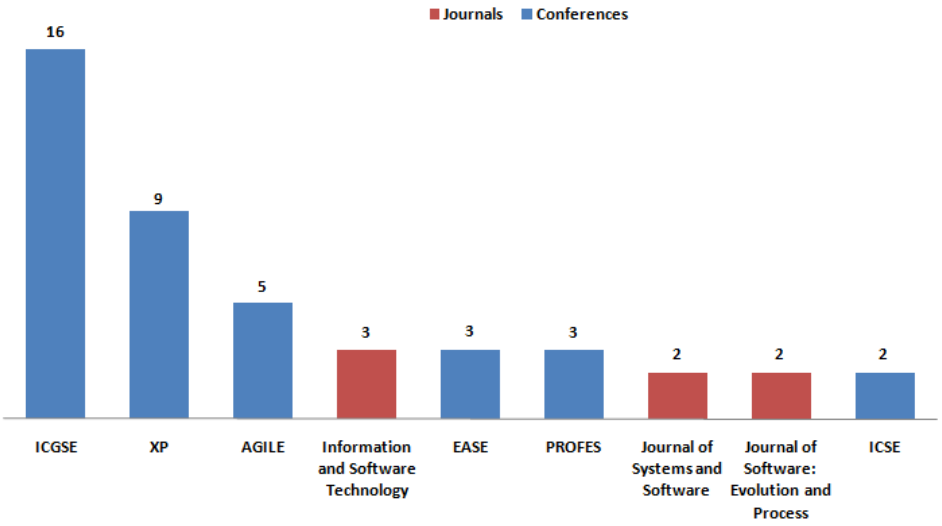


Figure 4.5 – Conferences and journals: Conferences show a clear lead of ICGSE, XP and AGILE, while journal publications are more widespread with many journals featuring just one included study. Only targets with more than one publication are included in this overview. Only years 2010 to 2014 are covered because the information was not available in (Jalali and Wohlin, 2010, 2012a).

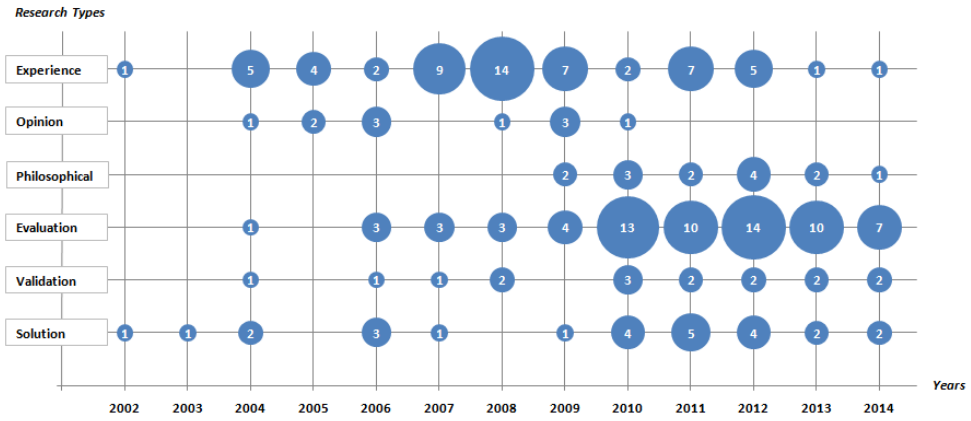


Figure 4.6 – Distribution of research types over the studied years 2010-2014 and data added from (Jalali and Wohlin, 2012a) for 2002-2009. There is a notable shift from experience papers towards evaluation papers. The total sum of studied papers for 2010-2014 is 111 papers here, because experience reports are not part of the included studies (N=95) for further analysis.

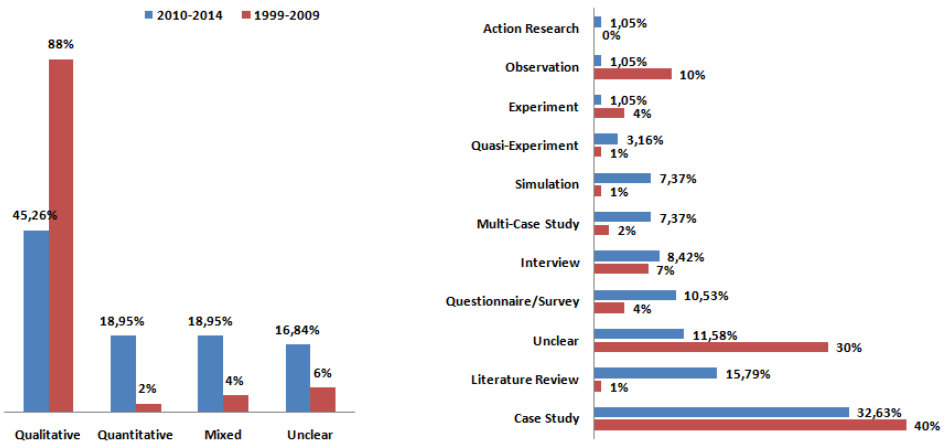


Figure 4.7 – Research methods and sub-methods for 2010-2014 and data added from (Jalali and Wohlin, 2012a) for 1999-2009.

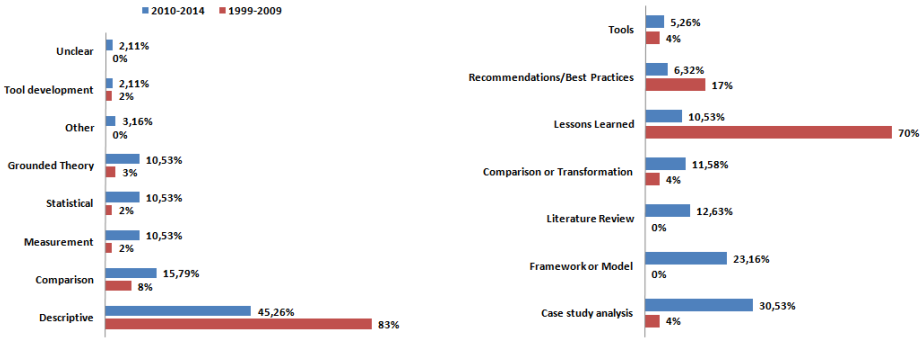


Figure 4.8 – Overview of the means of analysis and contributions of the studies for 2010-2014 and data added from (Jalali and Wohlin, 2012a) for 1999-2009.

more diverse research approaches compared to 1999-2009 with 88% qualitative studies. The most used research approach in 1999-2014 was the case study. Otherwise there is a rise in literature reviews, which can be explained by a maturing research field, and fewer studies with an unclear approach possibly due to experience reports being excluded in 2010 to 2014. Figure 4.8 shows that descriptive means of analysis are still widely used (45%) but not as much as before 2010 (83%). Contributions in the form of lessons learned (70%) are now lesser used in favor of case study analysis (31%) and model or framework development (32%) (cf. Figure 4.8). Only one paper (Vallon et al., 2014) in the fifteen years of 1999-2014 employed an action research approach.

4.3.2 Empirical Background

The majority (66 publications) of the 95 included studies (cf. Appendix A.2) in the years of 2010-2014 had an empirical background. If a study analyzed several cases, i.e. a multi-case study, each case was treated individually for extracting empirical information. However, a study featuring a survey has an empirical background but cannot be used to extract single-case characteristics. The deeper analysis thus continues for 62 reported cases (either from single-case or multiple-case studies and not taking into account e.g. surveys that could not be used for single-case extraction) from the 66 empirically based publications.

Table 4.3 shows case characteristics for all of the 62 reported cases between 2010-2014. Numbers in parentheses also show data from Jalali and Wohlin (2010) for the years of 1999-2009. Most cases (47) focus on the SE process as a whole rather than on a specific part of the workflow which indicates that research on agile practices in DSD is still quite holistic rather than in-depth. Certain context details were missing more frequently than others such as project duration (29 times), application domain (17), project size (15) or even whether distributed development was global or onshoring in the study (7). 45 cases explicitly reported success and only 3 reported failure, which shows that empirical publications in the field are drastically more solution-centric than problem-centric. Consistent with results from (Jalali and Wohlin, 2010), project size, project duration and application domain is still frequently not reported. In contrast to (Jalali and Wohlin, 2010), most studies targeted the whole SE process rather than a specific knowledge area.

Project Size		**Domain**		**Knowledge Area**	
Large	16 (6)	Unclear	17 (34)	SE Process	47 (5)
Small	16 (10)	Web	12 (8)	SE Management	6 (7)
Medium	15 (7)	Telecommunications	6 (3)	Tools & methods	5 (4)
Unclear	15 (18)	Service Provider	6 (1)	Requirements	2 (2)
Project Duration		Enterprise Software	6	Testing	1 (3)
Unclear	29 (35)	Finance	4 (1)	Design	1 (2)
Long	21 (11)	Open Source	3	**Successful**	
Medium	11 (6)	Automation	2	Yes	45 (49.5)
Short	1 (0)	Mobile	2	Unclear	14 (8)
Global Development		Industrial Products	1	No	3 (2.5)
Offshore	47 (37)	Energy	1	**Participants**	
Onshore	8 (1)	Supply Chain	1	Industry	54
Unclear	7 (14)	Risk Management	1	Students	8

Table 4.3 – Overview of the characteristics from 62 reported empirical cases for years 2010-2014. Numbers in parentheses is data added from (Jalali and Wohlin, 2010) for years 1999-2009.

4.3.3 DSD and Agile

Figure 4.9 provides an overview of distribution settings and the agile processes in use in a bubble chart diagram. The numbers in the bubbles of each block *Agility*, *Sites*, *Teams*, *Time*, *Distance*, *Sourcing* and *Shoring* sum up to 62, i.e. the total number of empirical cases analyzed. Since Jalali and Wohlin (2012a) use a more coarse-grained taxonomy, results cannot be compared to years 1999-2009.

Scrum is used in the majority of studies (37 out of the 62 cases) followed by mixed approaches (11 cases) and unspecified ones (10 cases). Compared to the years 1999-2009 studied in (Jalali and Wohlin, 2012a), two observations are apparent. First, the category *agile* was not used in this thesis' systematic mapping study. If it was obvious that scrum practices are mainly applied then the category scrum was assigned. If it was not obvious then the process was *unclear* rather than *agile*. The second observation is that Figure 4.9 does not show any Extreme Programming (XP) studies. The explanation is that in no case XP was used exclusively, but rather a combination with scrum, which falls under category *mixed*. Lean software development was only reported as process of choice in 3 cases, so it does either not get applied in DSD settings or is not reported by researchers, which indicates a research gap that requires further attention.

The majority of reported studies had the following characteristics: offshore (43 cases), far distance (29), large time gap (23), all-agile teams (23), two site environment (22) and insourcing (16) but closely followed by outsourcing (14), i.e. complex global cases. It also has to be noted that context is very seldom reported to a full extent, thus limiting the generalizability of results of a study. The most often skipped context details are team structure (36 cases), sourcing type (32), level of agility within the project/organizational environment (27), geographic distance (25) and temporal distance (25), which are in the 40% to almost 60% range of the total of 62 empirical cases.

Table 4.4 shows the supplier and customer countries involved in agile DSD. To achieve comparison to (Jalali and Wohlin, 2012a), cases with multiple countries were counted in a supplier-customer relationship as $\frac{1}{N}$ where N is the number of suppliers. In the studied years of 2010-2014, even more apparent than in the findings of Jalali and Wohlin (2012a), in many customer-supplier relationships (63.42% 2010-2014 vs. 32.86% 1999-2009) the customer's country was not reported in the analyzed cases. In alignment to (Jalali and Wohlin, 2012a), the reported main customers were the United States, followed by the UK and Denmark. Suppliers were reported more often than customers but still 39.61% remained in the dark. Jalali and Wohlin (2012a) found a similar behavior for the years of 1999-2009 with 41.55% undisclosed suppliers. The

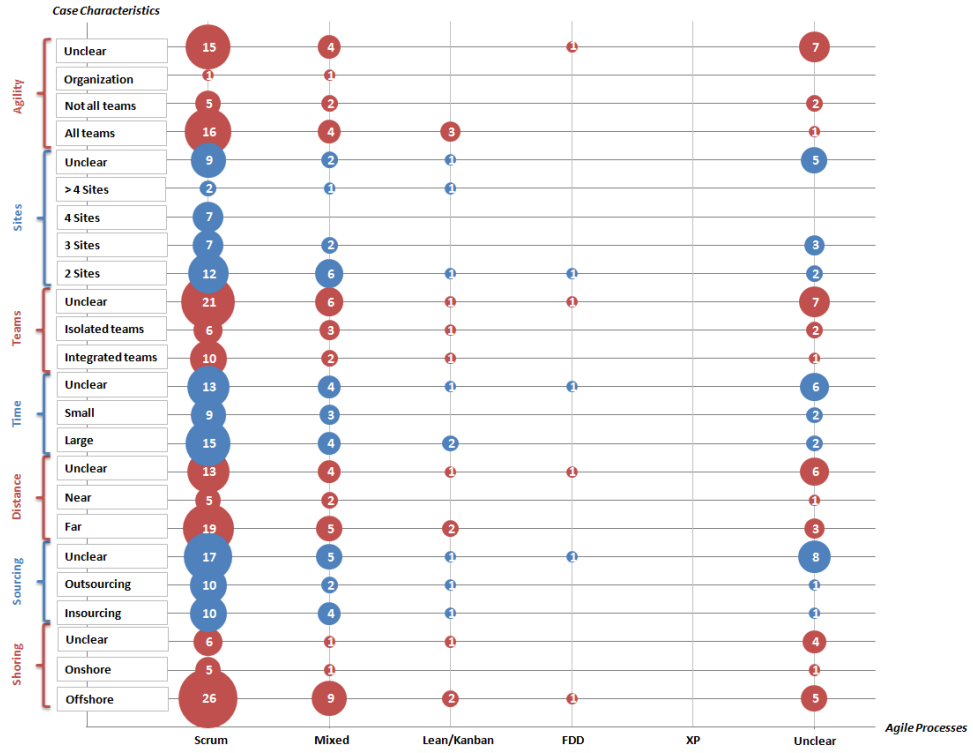


Figure 4.9 – Mapping the usage of agile processes against the reported DSD characteristics in the 62 reported cases in years 2010-2014.

most reported suppliers were Finland, India and the United States. For years 1999-2009 (Jalali and Wohlin, 2012a), India was the leading reported supplier country. Compared to the years of 1999-2009 (Jalali and Wohlin, 2012a) there is a stable amount of customers being reported (10 now compared to 11 then), but a significant increase in reported suppliers (29 now compared to 20 then) indicating the coverage of a greater variety of cases.

Customers / Suppliers	Unclear	US	UK	DK	DE	FI	BR	NL	RU	IN	NZ	Sum
Unclear	18.33 (14)	2 (3.2)	2							1	1 (0.5)	**24.33 (17.7)**
Finland	5.83				(0.2)	0.83 (1)						**6.66 (1.2)**
India	1.33	1.5 (10.5)	2 (1)	1	(1)			0.5 (1)				**6.33 (13.5)**
USA	1.86	1.16 (3)	(0.5)			0.33						**3.35 (3.5)**
UK	0.78	0.33	0.5				0.5					**2.11**
Norway	2	(0.5)										**2 (0.5)**
Denmark				1					0.5			**1.5**
Russia	0.33	(0.5)		0.5 (1)	(0.2)				0.5			**1.33 (1.7)**
Argentina	1.2											**1.2**
Canada	0.2	0.33 (0.5)				0.5						**1.03 (0.5)**
Republic of Korea	1											**1**
Germany	0.5				0.5							**1**
Austria					1							**1**
The Netherlands	0.45							0.5				**0.95**
Hungary	0.5					0.33						**0.83**
Brazil	0.25						0.5					**0.75**
Ireland	0.33	0.33 (2)										**0.66 (2)**
Senegal	0.58											**0.58**
New Zealand	0.5											**0.5**
Malaysia	0.5											**0.5**
Sudan	0.5											**0.5**
Pakistan			0.5									**0.5**
Poland		(1)			0.5							**0.5 (1)**
Australia	0.5											**0.5**
Israel		0.33 (0.5)										**0.33 (0.5)**
Romania	0.33		(0.5)									**0.33 (0.5)**
Belgium	0.33											**0.33**
Bangladesh	0.33											**0.33**
Greece	0.25											**0.25**
Cambodia	0.25											**0.25**
Sum	**38.96 (14)**	**5.98 (21.7)**	**5 (2)**	**2.5 (1)**	**2 (1.4)**	**1.99 (1)**	**1**	**1 (1)**	**1**	**1**	**1 (0.5)**	**61.43 (42.6)**

Table 4.4 – Supplier to customer relationships between the countries involved in agile DSD in the studied papers. Numbers in parentheses is data added from (Jalali and Wohlin, 2012a).

Figure 4.10 shows the extraction of all agile practices that have been reported in more than one case. Out of all studied papers successful practices have been extracted 309 times in the five-year period of 2010-2014 and 444 in total for the fifteen years of 1999-2014. For years 2010-2014, there is a strong support for the successful implementation of the most basic scrum practices such as standup meeting (32 cases), product owner (32, including variations of proxy product owner and product owner teams), backlog (31), sprint planning (25), retrospective (23), scrum master (21), user stories and sprint reviews (18 cases each). Neglected scrum practices (or ones that did not receive explicit attention in reports) were estimation meetings (2 cases), self-organizing teams (2) and burndown charts (2). Also, the scrum of scrums (6) has been seldom reported in DSD environments, although it is a practice to support scaling in agile processes.

Compared to years 1999-2009 (cf. Figure 4.10) the main scrum practices are still in the center of agile DSD implementations. XP practices seem to have been lesser used (or lesser reported), which can be explained by the fact that many cases use a mixed approach such as *XP@Scrum* (Vriens, 2003) with scrum for the general SE process and XP for development practices also in distributed development settings, which is also one of the results of this thesis' systematic mapping study (cf. mixed approach in Figure 4.9). Figure 4.11 shows which means have been used to overcome distance in agile processes, which are of general nature and not related to agile methods as such: video/audio conference meetings (27 cases), contact visits (11), instant messaging (10), wiki (9), screen sharing (3), ambassador (2) and chat (2). While these means do not qualify as *agile* practices, they are still important for the ADAPT framework as at least some of them (or related tools not covered in this systematic mapping) are needed to be part of any distributed process implementation to overcome distance.

4.3.4 Summary

This section answers the research questions RQ3a, RQ3b and RQ3c, which haven been addressed in this chapter's systematic mapping study.

RQ3a. What does the research landscape in the field look like in the 15 years of 1999 to 2014?

As Table 4.2 illustrates, the systematic search showed that 94.74% of the included studies could have been found using only the Scopus and Compendex databases (or 84.21% using only Scopus). The numbers also incorporate an additional manual screening of the proceedings of ICGSE, XP and AGILE conferences because these are the most prominent venues to publish papers on agile DSD. Journals received lesser attention as targets for publication with

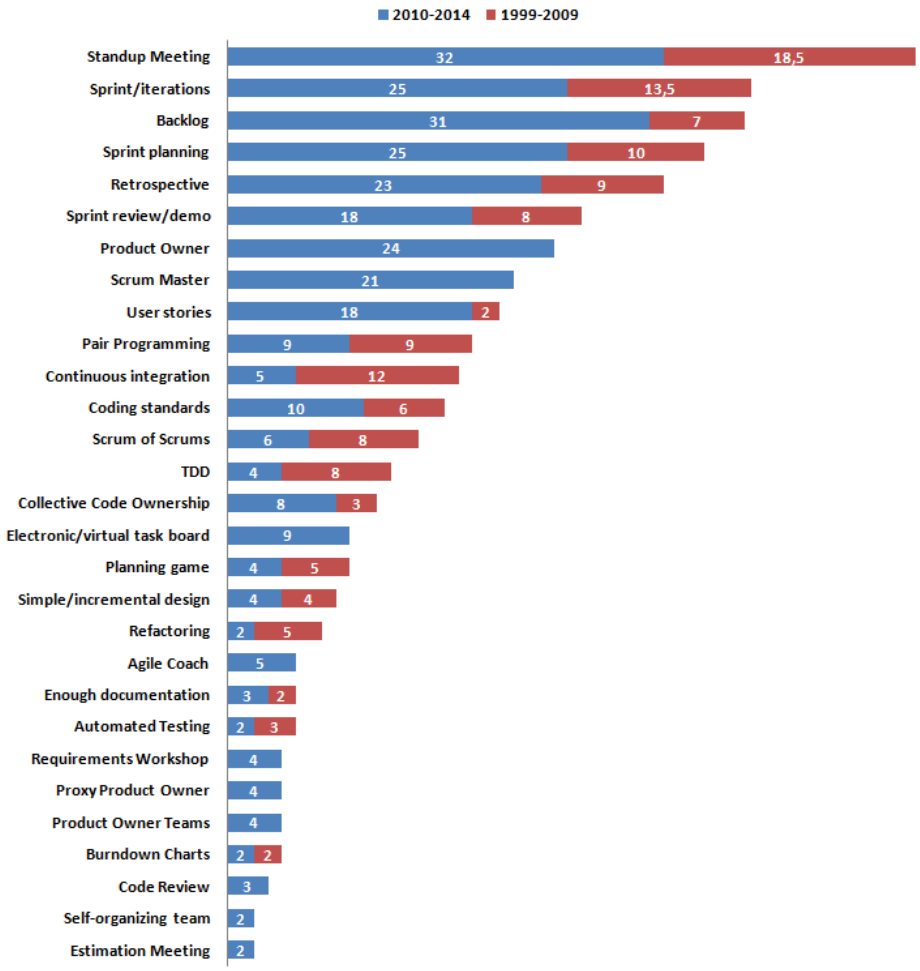


Figure 4.10 – Frequencies (>2) of successful application of agile practices for the studied years 2010-2014 and also years 1999-2009 by Jalali and Wohlin (2012a).

Information and Software Technology, Journal of Systems and Software and Journal of Software: Evolution and Process being the top three (cf. Figure 4.5). As Figure 4.4 shows, the most active countries interested in researching agile DSD were the United States, Finland and Germany, and the most active universities in the field were Aalto University, Universiti Teknologi PETRONAS and Blekinge Institute of Technology. The most involved countries in agile DSD were Finland and India as suppliers, UK and Denmark as customers and the United States in both categories (cf. Table 4.4).

Figure 4.6 draws a map of research types, showing that agile practices in DSD is an active research field with a variety of research types, most prominently evaluation studies with an empirical background. Figure 4.7 shows that out

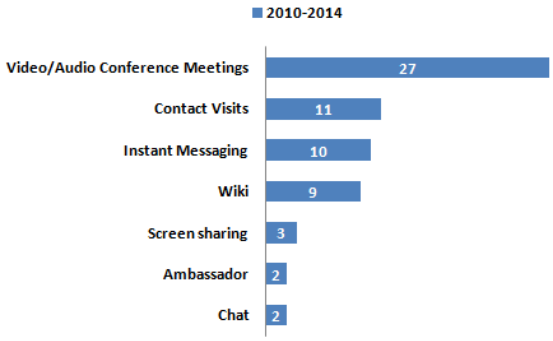


Figure 4.11 – Frequencies (>2) of successful application of means to overcome distance in agile processes for the studied years 2010-2014, which was not covered by Jalali and Wohlin (2012a) for years 1999-2009.

of 95 included studies about half used a qualitative approach and the rest is almost equally split between quantitative, mixed method or a not properly specified methodology. The most used research approach in 1999-2014 was the case study. Otherwise there is a rise in literature reviews, which indicates a maturing research field.

As Table 4.3 showed, most reported cases in years 2010-2014 (47 out of 62) focus on the SE process as a whole rather than on a specific part of the workflow which indicates that research on agile practices in DSD is still quite holistic rather than in-depth. Context details were often not stated clearly such as project duration (29 cases), application domain (17) and project size (15). 45 studies explicitly stated success and only 3 reported failure, which leads to the assumption that publications are more solution-centric than problem-centric.

RQ3b. What has changed in the later five years 2010 to 2014 in comparison to the former ten-year period of 1999-2009?

Figure 4.1 summarizes the whole study inclusion process, which led to the final set of 95 included studies (cf. Appendix A.2). As Figure 4.6 shows, there is a major increase in frequency of evaluation papers in the studied five-year period as a natural evolvement to the experience papers that were most frequent in the years up to 2009 (Jalali and Wohlin, 2010, 2012a). This finding indicates increasing research attention and interest to mature the field, which is also supported by a shift to more mixed and quantitative approaches, from close to 90% qualitative research 1999-2009 down to close to 50% 2010-2014 (cf. Figure 4.7). There is notably greater effort towards the evolvement of frameworks and models (cf. Figure 4.8) rather than mostly lessons learned before (Jalali and Wohlin, 2012a).

The greatest change of directions compared to 1999-2009 is the predominance of scrum, more frequent application of mixed methods and the neglecting of XP as a standalone process in agile DSD (Figure 4.9). Figure 4.10 supports that observation regarding a lack of XP processes by showing that XP practices have been applied fewer times successfully in 2010-2014 compared to the years of 1999 to 2009.

In general underspecified context had been an issue in previous research (Jalali and Wohlin, 2010, 2012a) already. However while the issue is definitely not resolved, as e.g. Figure 4.9 or Table 4.3 point out, most studies in the 2010-2014 studied time period at least painted a better picture for case characteristics than just "distributed teams" or "agile", which was frequent in years 1999-2009.

RQ3c. What are common agile practices and distribution scenarios?

Table 4.3 gives an extensive overview of empirical background data and Figure 4.9 maps global distribution scenarios against agile processes. The majority of reported studies had the following characteristics: offshore (43 cases), far distance (29), large time gap (23), all-agile teams (23), two site environment (22) and insourcing (16) but closely followed by outsourcing (14), i.e. complex global cases.

Scrum is by far the most used agile process across all distribution scenarios (37 cases), followed by mixed methods (11), most notably the combination of scrum methodology with XP development practices. Figure 4.10 also supports that observation since scrum practices are the most frequent ones.

As Figure 4.10 shows, out of all studied papers successful practices have been extracted 309 times in the five-year period of 2010-2014 and 444 in total for the fifteen years of 1999-2014. For years 2010-2014, there is a strong support for the successful implementation of the most basic scrum practices such as standup meeting (32 cases), product owner (32, including variations of proxy product owner and product owner teams), backlog (31), sprint planning (25), retrospective (23), scrum master (21), user stories and sprint reviews (18 cases each). Neglected scrum practices (or ones that did not receive explicit attention in reports) were estimation meetings (2 cases), self-organizing teams (2) and burndown charts (2). Also, the scrum of scrums (6) has been seldom reported in DSD environments, although it is a practice to support scaling in agile processes.

Agile practices are often supported by means to overcome distance as shown in Figure 4.11. These means are of general nature and not related to agile methods as such: video/audio conference meetings (27 cases), contact visits

(11), instant messaging (10), wiki (9), screen sharing (3), ambassador (2) and chat (2), but they are still important for the ADAPT framework as these or similar means are needed in any distributed process implementation, including agile ones.

4.4 Implications for Research and Practice

Replicating a systematic mapping study. Due to the detailed documentation of the process used by Jalali and Wohlin (2010, 2012a), it was possible to set up a procedure similar to the original one. Some adaptations have been made such as reporting the distribution context in alignment to the taxonomy of (Šmite et al., 2014), a natural improvement given the fact that one of the authors of the original study (Jalali and Wohlin, 2012a), Claes Wohlin, has been involved in the creation of that taxonomy in the meantime. By expanding on the former results, it was possible to cover a full fifteen-year time span of 1999-2014.

Insufficient abstract quality. As has been noted in (Jalali and Wohlin, 2012a) and other systematic reviews such as (Petersen et al., 2008), it is often not enough to judge studies based on abstracts due to low quality and missing structure, not even for simple systematic maps. It this thesis' systematic mapping many studies had to be excluded after full-text analysis although having been originally included based on abstract, title and keywords. This proved especially true for judging whether agile or DSD was a focus in the full text of the study despite of what has been stated in the abstract. If the systematic mapping had been based only on abstract, title and keyword, as many as one third of false positives would have been included, substantially distorting results. Hence this thesis' systematic mapping study's results show that a systematic mapping is hardly feasible without full-text analysis or at least not without studying the full text adaptively as suggested in (Petersen et al., 2008).

Maturing research field. In the five years of 2010-2014 there was a drastic shift from experience papers to evaluation ones indicating increasing researchers' attention to investigate the application of agile practices in DSD. Given the amount of empirical studies, the analysis of experience papers was skipped in this thesis' systematic mapping, because they usually lack a rigorous approach and are thus subject to bias. Nevertheless this thesis' mapping study still comprised a larger final set over five years (2010 to 2014) than the previous studies over a ten-year period (1999 to 2009).

Widely used scrum and "XP@Scrum". Scrum and XP@Scrum (scrum combined with XP development techniques) have been by far the most used processes across different kinds of distribution scenarios, which makes the application of agile practices more specific compared to years 1999 to 2009, where many studies just reported being "agile".

Neglected research approaches. The trend up to 2010 (Šmite et al., 2010c) is continued that the most empirical evidence is based on case studies with interviews being the primary data source. A possible variation could be to employ more action research as has been pointed out by Sjøberg et al. (2007). There was only a single study (Vallon et al., 2014) in the fifteen years of 1999-2014 that has adopted action research in agile DSD, although it is arguably the most realistic research setting and enables the researcher to gain an in-depth and first-hand understanding (Sjøberg et al., 2007).

No lean processes in agile DSD. Lean software development is getting used in many different environments in software development (Wang et al., 2012), but as this thesis' systematic mapping study shows its application in DSD is close to non-existent, although *lean* was a designated keyword in the search. Thus little is known about the application of lean tools to DSD, hence requiring further research attention.

High-level studies covering the whole process. Although description of study context has improved over (Jalali and Wohlin, 2012a), the majority of studies focuses on the whole process. As such a study often cannot cover the implementation of agile practices in great detail leaving out important information for use by practitioners. Hence it would be interesting to see more in-depth studies covering specific parts of the agile value chain in DSD. It often seemed that agile practices have been described rather vaguely, along the notion of "scrum practices like backlog and sprints have been used" rather than naming all used practices and also elaborating more on *how* the practice has been applied. For example if sprint planning is used in DSD, it will make a huge difference which sites were involved, whether it was held on site or virtually and which specific techniques and procedure was used. It does not really add any value to just name an agile practice in a DSD environment that has been designed for use in collocated environments without further elaboration.

Solution-centric publications. While the results show many successful agile practices, the report of challenges with agile practices is rather scarce and requires further attention. Still, this thesis' systematic mapping study identified several practices that were not used frequently in a DSD context or have not received much attention: estimation meeting, self-organizing team, code review, burndown charts or requirements workshop. If these practices do not get used, it would be interesting to investigate why and which practices are used instead.

No comprehensive agile DSD framework. Although there is an increasing amount of studies contributing models and frameworks, there is still no process framework to present a comprehensive approach to applying agile practices in DSD. This has been pointed out by Jalali and Wohlin (2012a), but the gap still exists after having analyzed years 2010 to 2014. This thesis' ADAPT framework strives to fill this research gap (cf. following Chapters 5 and 6).

Checklist for reporting context in agile DSD. The data extraction plan

(cf. Appendix A.3) has been improved from (Jalali and Wohlin, 2010) and there were almost no studies which managed to satisfy all inquired context details. Missing context details in general make it difficult to make use of the findings (Šmite et al., 2008). Serious gaps were found in the presentation of context in years 2010 to 2014 such as: the distribution type was not properly specified in 40% to 60% (depending on the criterion) of the cases (cf. Figure 4.9). Crucial information such as project duration was not reported in 50% of the cases. Project size and application domain were not reported in 25% of the cases, respectively. Furthermore, the customer country (60%) and the supplier country (40%) were frequently not reported. When context information was reported it also often had to be carefully extracted from the full paper text rather than being stated at one point or a table. The author encourages other researchers to use the updated checklist of Jalali and Wohlin (2010), which also incorporates the DSD taxonomy of Šmite et al. (2014) and is fully presented in Appendix A.3.

4.5 Conclusion

This chapter presented a systematic mapping of the application of agile practices in distributed software development for the five-year period of 2010 to 2014, and also in comparison to the results of studies (Jalali and Wohlin, 2010, 2012a) for years 1999 to 2009, effectively covering a fifteen-year period and as such enriches the body of knowledge for applying agile practices in distributed software development. On top of the previous work 95 papers for the years of 2010-2014 (cf. Appendix A.2) were analyzed and it was found that agile practices have been used 309 times successfully (total for 1999-2014: 444 times). The top three successful practices have been standup meeting, sprint iterations and backlog and in the fifteen years of 1999-2014. A serious shortcoming was identified in the presentation of context details in about 50% of the empirical cases studied, depending on the criterion, which limits the generalizability of results of the individual cases. The author proposes a data extraction checklist (cf. Appendix A.3) based on this thesis' systematic mapping study that can be used to describe context details in future studies to counteract this recurring problem.

Key findings of the systematic mapping include the following:

Best database: Scopus, followed by Compendex
Top three countries (counting all universities): USA, Finland and Germany
Top three universities: Aalto University (FI), Petronas University of Technology (MY) and Blekinge Institute of Technology (SE)
Top three conferences: International Conference on Global Software En-

gineering (ICGSE), International Conference on Agile Software Development (XP), The Agile Conference (AGILE)
Top three journals: Information and Software Technology (IST), Journal of Systems and Software (JSS), Journal of Software: Evolution and Process (JoS: EP)
Research types: there is a clear shift from experience papers to evaluation studies, indicating a maturing research field
Research methods: shift from almost exclusive qualitative studies to quantitative and mixed methods as well
Empirical agile DSD context: is often not fully reported, the most neglected context information is team structure, sourcing type and level of agility
Empirical project context: also often missing important information such as project duration, project size and application domain
Most common agile process: scrum, followed by mixed methods (mostly involving scrum and XP combined)
Most common distribution scenario: offshore, either insourcing or outsourcing, far distance, large time gap, involving two sites
Most reported customer countries in DSD: USA, United Kingdom and Denmark
Most reported supplier countries in DSD: Finland, India and USA

The remaining related work until the finalization of the thesis (09/2014 to 12/2015), not covered in this chapter's mapping study, is added to the discussion (cf. Chapter 7) in Section 7.3.

CHAPTER 5

Single-Case Analysis

Single-case reports have been published and presented at the 14th International Conference on Agile Software Development (XP 2013) in Vienna (Vallon et al., 2013b), the 8th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2013) in Ville d'Angers, France (Vallon et al., 2013a) and the 2014 Agile Conference (AGILE 2014) in Orlando, FL (Vallon et al., 2014). The multiple-case report has been published in the journal Computer and Information Science (Vallon and Grechenig, 2016).

Contents

This chapter presents the individual case reports on the multiple-case study that has been conducted in this thesis. The study provides the following contributions to the evolving research field of applying agile process in distributed software development (DSD) environments:

1. Empirical evidence from a long-term multiple-case study covering three different distribution scenarios: cross town, no timeshift and continental

2. Cross-case analysis of agile practices among these three different distribution scenarios

3. Addressing the need for more robust primary empirical studies as identified in DSD in general (Marques et al., 2012) and in particular for agile practices in DSD (Hanssen et al., 2011)

Section 5.1 presents the overarching multiple-case research design. Individual cases *CrossTown* (Section 5.2), *NoTimeshift* (Section 5.3) and *Continental* (Section 5.4) are then reported, discussing each background, challenges, agile practices and the resulting input of guidelines of practices for the ADAPT framework. An aggregation of single-case results and the cross-case (multi-case) analysis follows in Chapter 6.

5.1 Research Design

This multi-case study is a major step in this doctoral research project aimed at empirically investigating the application of agile practices in DSD. As such it adds to the empirical basis of the research field and represents an important link between the initial systematic mapping in Chapter 4 and the development of a first iteration of the ADAPT framework in Chapter 6. The research design follows the guidelines of Yin (2003) for general case study design and Verner et al. (2009) for conducting multiple-case studies in software engineering in particular. The case study protocol presented in this chapter is based on the template by Brereton et al. (2008). The research design decisions made are explained and embedded in the following subsections.

Both Jalali and Wohlin (2012a) and this thesis' extended systematic mapping (cf. Section 4) showed that context is not richly described in empirical studies in the area of agile DSD. The author of this thesis further investigated how context has been described in past studies and built a conceptual framework (cf. Table 5.1) to use for this multiple-case study and address the identified shortcoming. Moreover, in previous multiple-case studies methodological triangulation was found to be scarce. This finding supports the claim (Marques et al., 2012) that there is a need for robust primary empirical studies researching DSD and agile practices in DSD in particular (Hanssen et al., 2011). Hence, the following research gap was identified:

As context information in past empirical studies is often not richly provided, it is hard to generalize from past studies in the field. This study aims to investigate three cases implementing agile practices in DSD in significantly different distribution scenarios and presents rich contextual information for each case respectively. The nature of this multiple-case study is exploratory in the way that is has no clear, single set of outcomes (Yin, 2003) and that it is to the best of the author's knowledge unprecedented in the approach of analyzing the emergence of common heuristics (Heeager and Rose, 2014) in several distribution scenarios (cross town, no timeshift and continental).

Based on that objective, the research question (RQ4) guides the research:

RQ4a. *What process design guidelines and best practices can be formulated to increase the chances of a successful agile process implementation in distributed environment?*

RQ4b. *Do the different distribution scenarios affect the implementation of agile practices?*

Since this research is exploratory there are no a-priori propositions (Yin, 2003) and the research is guided by the research question RQ4.

5.1.1 Related Multiple-Case Studies

Jalali and Wohlin conducted a systematic mapping (Jalali and Wohlin, 2010) and literature review study (Jalali and Wohlin, 2012a) on agile practices in DSD reported in the years of 1999-2009. This section presents the multiple-case studies of their included set of primary studies. Ramesh et al. (2006) looked into applying agile principles to DSD as early as 2006 and concluded that careful integration may help in addressing challenges with communication, control and trust across distributed teams. Sison and Yang (2007) analyzed two cases in the Philippine IT industry and found agile principles to result in greater learning and greater teamwork in both cases. Paasivaara et al. (2009) offer insights to three cases of applying scrum in distributed environments. The results show how scrum practices have been applied in the cases and what the identified challenges and benefits were. So in the years of 1999-2009 (Jalali and Wohlin, 2012a) there were only three multiple-case studies in this specific area.

This thesis also looked for multiple-case studies in the more recent years 2010-2014 (cf. Chapter 4), which have not been covered in (Jalali and Wohlin, 2012a). Srinivasan and Lundqvist (2010) focus on agile DSD tied to Indian software organizations. The authors conclude that the following guidelines play an important role: appropriate selection of personnel, providing necessary training and mentoring and creating a set of work practices that promotes process excellence. Hossain et al. (2011b) analyze in four cases how scrum was tailored to fit individual DSD contextual requirements. Bass (2012) also investigated process tailoring, concluding that XP practices were much less widely used compared to scrum practices. Paasivaara et al. (2012) focused on ways of scaling the product owner role in their two-case study. Ramesh et al. (2012) investigate the ambidexterity of balancing agile and distributed development, which the authors see conflicting in nature as DSD forces more

plan-based approaches. The proposed solution is to simultaneously pursue the two rather than creating dual structures. Badampudi et al. (2013) identified 17 challenges and 28 mechanisms how these challenges affected certain roles in global large-scale agile project environments, involving enabling, planning, and coordinating the scrum teams and integrating their results. Daneva et al. (2013) focus on requirements (re)prioritization by so-called delivery stories which complement user stories with architectural design implications, test scenarios, effort estimation and associated risk. Paasivaara and Lassenius (2014a) focuses on agile coaching of global software development projects, pointing out their importance and proven benefit in the cases under study.

So in the research of related work eight multiple-case studies were found for the years of 2010-2014 as compared to three multiple-case studies for 1999-2009, which indicates an increasing research focus and interest on agile practices in DSD. Contextual details have often been not fully described, which corresponds with observations by Jalali and Wohlin (2012a) and the systematic mapping of Chapter 4.

5.1.2 Multiple-Case Study

The case study design was chosen as a natural fit to the research problem as it "investigates a contemporary phenomenon within its real-life context, especially when the boundaries between phenomenon and context are not clearly evident" (Yin, 2003, p. 13). Moreover, the author wants to focus on covering contextual background for which the case study is especially well-suited (Yin, 2003). In multiple-case studies it is especially important to follow a rigid protocol (Yin, 2003), which is described in Section 5.1. Evidence from multiple cases is considered more compelling (Herriott and Firestone, 1983).

The unit of analysis is a project within an organization for which agile processes are applied in a distributed development environment, i.e. the development itself has to take place on at least two sites. Hence, this multiple-case study is an embedded case study focusing on the output of individual projects as compared to a holistic one (Yin, 2003). For selecting information-rich cases purposeful maximum variation (heterogeneity) sampling (Patton, 2002) was used, i.e. heterogeneous cases were chosen based on the characteristics developed in the conceptual framework (cf. Table 5.1). Since scrum is the most widely used agile process also in DSD (Jalali and Wohlin, 2012a) the case selection was limited to scrum and the distribution scenario of the project was chosen as the primary dimension for selecting heterogeneous cases. This also adheres to the definition of Strauss et al. (1998, p. 120) for theoretic sampling, where "we want to know what happens [...] when the conditions under which it occurs vary". The final selection involved three cases applying scrum practices in greatly different distribution scenarios: cross town (sites in the same

city), no timeshift (sites in the same country) and continental (sites spanning multiple countries).

The assumed problem that individual cases are too different is intentional and argued to produce stronger results: "Any common patterns that emerge from great variation are of particular interest and value in capturing the core experiences and central, shared dimensions of a setting or phenomenon" (Patton, 2002, p. 235). In a multiple-case study, each case should be seen as a unique "experiment" and not just another sample (Yin, 2003). First, each case is treated individually with an individual case report before the cross-case analysis (Yin, 2003). This also aligns with Patton's guidance to variation sampling, i.e. to describe the uniqueness of each case and also look for common themes (Patton, 2002).

5.1.3 Conceptual Framework

DSD is a contemporary phenomenon which is studied within its real-life context. As such the boundaries between phenomenon and context are not clearly evident (Yin, 2003). There has been joint effort by Šmite et al. (2014) on how to properly provide context details and a suggested taxonomy in DSD research. However in past empirical studies context has often been poorly described as has been pointed out in the systematic literature review by Jalali and Wohlin (2012a) (years 1999-2009) and this thesis' own systematic mapping (cf. Chapter 4, years 2010-2014).

Based on the comprehensive review of previous work in Chapter 4, a conceptual framework with the key factors was defined that is the focus of the data collection (Verner et al., 2009). It is a central point in the process of "sorting out of the wheat of what is central to your findings from the chaff of specific irrelevancies" (Robson, 1993, p. 72). The conceptual framework is based on three main factors relevant to the context of the empirical study: DSD inspired by (Šmite et al., 2014; Jalali and Wohlin, 2012a), scrum practices extraction inspired by (Paasivaara et al., 2009; Hossain et al., 2011a) and general information with regard to the unit of analysis, i.e. the project, inspired by (Jalali and Wohlin, 2012a; Hossain et al., 2011a). Table 5.1 drafts the conceptual framework with all identified relevant empirical factors and their respective sources.

5.1.4 Case Organizations

Three heterogeneous cases were purposefully selected with regard to their varying distribution scenarios. Identities are withheld to preserve privacy and the three projects receive pseudonyms which will be used throughout the thesis: CrossTown, NoTimeshift and Continental. The case CrossTown involves

Key contextual factors	Sub-factors	Extraction Details	Inspired by
DSD	Location	Onshore, Offshore	(Šmite et al., 2014)
	Legal Entity	Insourcing, Outsourcing	(Šmite et al., 2014)
	Geographic Distance	Near, Far (used for both Onshore and Offshore)	(Šmite et al., 2014)
	Temporal Distance	Similar, Different (Onshore) Small, Large (Offshore)	(Šmite et al., 2014)
	Socio-cultural Distance	Low, Significant	(Hossain et al., 2011b)
	Supplier Country	Country name	Jalali and Wohlin (2012a), cf. Ch. 4
	Customer Country	Country name	Jalali and Wohlin (2012a), cf. Ch. 4
	Number of sites	≥ 2	Jalali and Wohlin (2012a), cf. Ch. 4
	Team Distribution Type	Integrated Teams, Isolated Teams, Mixed	cf. Ch. 4
AGILE	Process	Scrum, XP, Lean, Mixed, ...	Jalali and Wohlin (2012a), cf. Ch. 4
	Agility Level	Not all teams, All Teams, Organization-wide	Jalali and Wohlin (2012a), cf. Ch. 4
	Sprint	Implementation Details	(Hossain et al., 2011b; Paasivaara et al., 2009)
	Sprint Planning	Implementation Details	(Hossain et al., 2011b; Paasivaara et al., 2009)
	Daily scrum	Implementation Details	(Hossain et al., 2011b; Paasivaara et al., 2009)
	Scrum of scrums	Implementation Details	(Hossain et al., 2011b; Paasivaara et al., 2009)
	Sprint Review	Implementation Details	(Hossain et al., 2011b; Paasivaara et al., 2009)
	Retrospective	Implementation Details	(Hossain et al., 2011b; Paasivaara et al., 2009)
	Backlog	Implementation Details	(Hossain et al., 2011b; Paasivaara et al., 2009)
PROJECT (Unit of Analysis)	Application Domain	The project's domain	Jalali and Wohlin (2012a), cf. Ch. 4
	Experience with Agile	Company's experience in years	(Hossain et al., 2011b)
	Experience with DSD	Company's experience in years	(Hossain et al., 2011b)
	Project Size	Sum of project personnel	Jalali and Wohlin (2012a), cf. Ch. 4
	Team Size	Site A (> 0), ..., Site N (> 0)	(Hossain et al., 2011b)
	Project Duration	In Months	Jalali and Wohlin (2012a), cf. Ch. 4
	Project Type	Industry, Student	Jalali and Wohlin (2012a), cf. Ch. 4
	Successful	Yes, No	Jalali and Wohlin (2012a), cf. Ch. 4

Table 5.1 – Conceptual framework for the key factors to be extracted. Extraction details show the possible (exclusive) selection choices when extracting data, separated by a comma, or a further description (without concrete selection choices, i.e. free text).

teams in offices across the same city, case NoTimeshift has teams distributed across the same country and case Continental features teams distributed across several countries of the same continent. Table 5.2 shows an overview of the contextual factors DSD and unit of analysis (project). The remaining contextual factor agile (scrum) is discussed during case analysis.

	Sub-Factors	CrossTown	NoTimeshift	Continental
DSD	**Location**	Onshore	Onshore	Offshore
	Legal Entity	Insourcing	Outsourcing	Insourcing
	Geographic Distance	Near	Near	Near
	Temporal Distance	None	None	Small
	Socio-cultural Distance	None	None	Low
	Suppliers	Austria	Austria	3 European Countries
	Customers	Austria	Germany	European Country
	No. of sites	2	2	3
	Team Distribution	Integrated Teams	Integrated Teams	Isolated Teams
PROJECT	**Domain**	Web & Hardware	Enterprise Software	Web
	Agile Exp.	7+ years	3 years	3+ years
	DSD Exp.	10 years	2 years	15+ years
	Team Size	Overall: 19 "Dev Site": 13 (11 Dev, 1 SM, 1 PO) "Test Site": 6 (5 Test, 1 SM)	Overall: 30 "Main Site": 20 (11 Dev, 3 Test, 3 SM, 3 PO) "Add. Site": 10 (8 Dev, 2 Test)	Overall: 39 "EUC1": 14 (6 Dev, 1 Test, 4 PO, 3 PMO) "EUC2": 19 (12 Dev, 5 Test, 1 PO, 1 PMO) "EUC3": 6 (4 Dev, 1 Test, 1 PMO)
	Duration	15 months	6 months	9 months
	Type	Industry	Industry	Industry
	Successful	Yes	Yes	Yes

Table 5.2 – Contextual information on the selected cases.

5.1.5 Data Collection

The data collection strategy is aimed at finding out which scrum practices organizations have applied to distributed projects in what way and which DSD practices have been used to complement the agile process (RQ4a and RQ4b). This multiple-case study is qualitative, with the exception of case NoTimeshift also allowing an additional tracking of quantitative data.

In previous multiple-case studies methodological triangulation was not a major concern, either not used or not reported. The most often used approaches are (from most to least frequent): semi-structured/open-ended interviews (Ramesh et al., 2006; Sison and Yang, 2007; Paasivaara et al., 2009; Srinivasan and Lundqvist, 2010; Paasivaara, 2011; Hossain et al., 2011a; Bass, 2012; Paasivaara et al., 2012; Ramesh et al., 2012; Badampudi et al., 2013),

document analysis (Sison and Yang, 2007; Hossain et al., 2011b; Ramesh et al., 2012) and in one case also observation (Hossain et al., 2011b). Triangulation is important because each method reveals different aspects of empirical reality (Denzin, 1978). Denzin (1978) has identified four types of triangulation: data triangulation (variety of data sources), investigator triangulation (use of different researchers), theory triangulation (multiple perspectives to interpret a single set of data achieved by using multiple investigators (Stake, 1995)) and methodological triangulation (multiple methods to study a single problem). Table 5.3 shows how triangulation was achieved for each of the respective cases. Since triangulation is expensive, it was employed reasonably and practically (Patton, 2002) within the possibilities and limitations of each case (Patton, 2002). The triangulation sources are used as described by Yin (2003). The research design followed Stake's advice in using several investigators for each case to achieve investigator triangulation and also theory triangulation to include different viewpoints and perspectives by means such as discussions and reviews (Stake, 1995). All supporting investigators were only involved in one case of the multiple-case study (except for one exception, where a supporting investigator was involved in both cases NoTimeshift and Continental) in an effort to minimize bias. The author was the principal investigator (PI) in all three studies.

Cases	**Method and Data Triangulation**	**Investigator and Theory Triangulation**
CrossTown	• Participant-Observation (1 action researcher) • Documentation (160 documents) • Archival Records (3863 tickets in issue tracking system, 274 wiki pages) • Physical Artifacts (thousands of sticky notes and dozens of paper boards)	Vallon (PI) +1 senior researcher +2 supporting investigators
NoTimeshift	• Interviews (N=7) • Direct Observation (5 meetings across several sprints) • Documents (15 documents) • Archival Records (579 tickets, 37 wiki pages) • Physical Artifacts (thousands of sticky notes and dozens of paper boards)	Vallon (PI) +1 senior researcher +3 supporting investigators
Continental	• Interviews (N=11) • Documents (273 documents) • Archival Records (only limited view)	Vallon (PI) +1 senior researcher +4 supporting investigators

Table 5.3 – Different types of triangulation in the three cases.

All interviews were semi-structured (Patton, 2002), recorded and later transcribed by an investigator. Interviews usually lasted one to two hours.

Data collection for case CrossTown involves the following activities due to the action research setting: explore possibilities for accessing project data in the issue tracking tool (beginning of study), update project diary with field notes and photos (daily), extract data from issue tracking tool (each sprint), discuss problems and solutions with practitioners and record actions (sprint retrospective as well as informal discussions during sprint), track results of sprint planning/review to analyze teams' performance as well as results of sprint retrospective to collect problems and solutions (each sprint), discuss and analyze the data collected with the off-site supporting investigators and senior researcher (each sprint).

5.1.6 Data Analysis

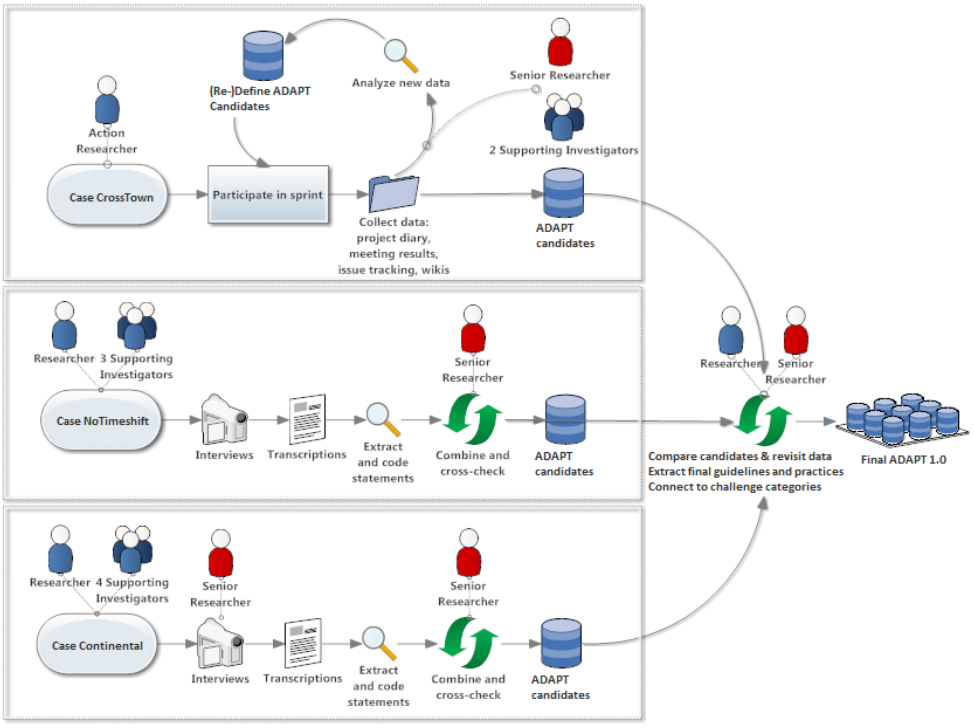


Figure 5.1 – Research methodology of the three individual cases and the cross-case analysis. Case CrossTown follows an action research approach and cases NoTimeshift and Continental use semi-structured interviews as primary source of investigation. Investigators attached in the illustration to a case's starting point have been participating in all steps of the individual case. All others (e.g. senior researcher) are explicitly attached only to the stages in which they have participated in.

Figure 5.1 illustrates the general data analysis process. All case study data was stored on a shared online drive. A description of all the applied techniques

follows in this section.

Action Research (AR): In case CrossTown action research was used as the primary research approach. AR uses "a spiral of steps, each of which is composed of a circle of planning, action, and fact-finding about the result of the action" (Lewin, 1946, p. 38). The term action research has been coined by Kurt Lewin in 1946 (Lewin, 1946). AR is, however, close to non-existent in SE research (Glass et al., 2002; Sjøberg et al., 2007; Santos and Travassos, 2009) although it provides the most realistic research setting and thus enables the researcher to gain an in-depth and first-hand understanding (Sjøberg et al., 2007). Usually the researcher will incorporate one or several feedback cycles (Davison et al., 2004). In CrossTown the process in use was scrum, which provides a variety of feedback cycles which can be utilized for AR. Hence, AR aligns very well with scrum, as the problem-solving cycle is already part of the process. A parallel second research cycle can be established without altering the original process.
Based on (Checkland and Holwell, 1998), McKay and Marshall (2001) propose a two-cycle feedback loop with one problem cycle to address the problematic situation and one research cycle to achieve scientific goals. This supports the separation of the dual imperatives of AR and enables scientific rigor. The stakeholders and participants own the problem-solving cycle (sprint iteration), while the researchers own the (in this case parallel) research cycle (McKay and Marshall, 2001). Researchers and participants collaborate to meet respective goals.
An additional research cycle can be added as pictured in Figure 5.2. The cycle starts with the research question RQ, a theoretical framework F based on propositions, a research method M_R and a problem-solving method M_{PS}. The research problem A was defined as the possibilities, challenges and solutions of transforming single-site scrum to a DSD. In this case study P is analyzed, which is a real-world instance of the research problem A. By evaluating actions on P, results for A shall be derived.
The problem-solving method M_{PS} is the regular scrum process, where decisions on the process are taken and evaluated in the retrospective meeting at the end of each sprint in the real-world problem situation P. The research method M_R reflects on the problems and decisions (actions) in a separate cycle with regard to research problem A. This parallel two-cycle process enables the separation of problem-solving and research interests. The steps of the dual AR cycle of Figure 5.2 are further explained in Table 5.4. A well defined process is very important to establish recoverability (Checkland and Holwell, 1998; Santos and Travassos, 2009). In this case, the exit criterion is time-boxed, i.e. the end of the product development phase.
One researcher (the author) participated in the action research assuming the role of a scrum master. The supporting investigators and senior researcher (who were not participating in the action research on site) analyzed new data,

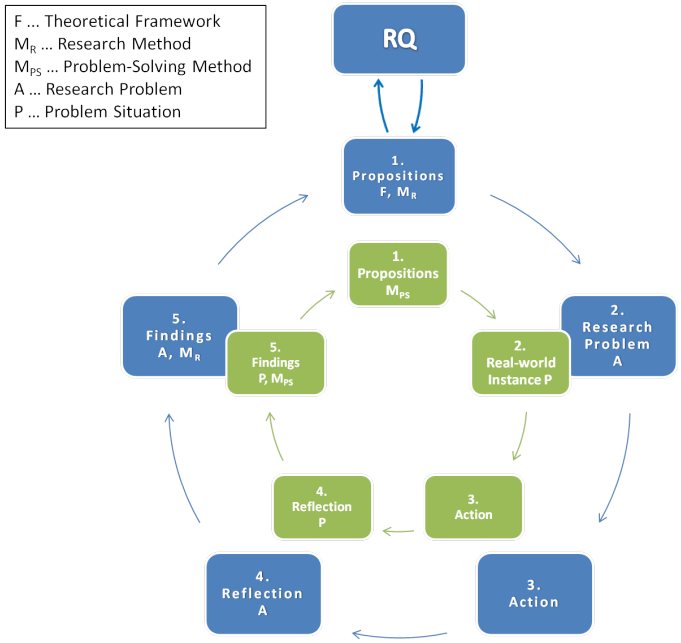


Figure 5.2 – A dual imperative AR cycle (adapted from (McKay and Marshall, 2001)): The inbound practitioners' cycle, designed to solve practical problems, provides input to the parallel outbound researchers' cycle, designed to gather knowledge on the research problem.

discussed findings in regular research meetings each sprint with the action researcher and corroborated findings with documents and archival records.

Interviews: In both cases NoTimeshift and Continental semi-structured interviews were used which allow a conversational manner but still follow an interview protocol (Yin, 2003). The interviews were recorded and later transcribed and lasted from 0.5-3 hours (on average 100.5 minutes). The author interviewed all different roles at least once for each case and also involved stakeholders. For analyzing the semi-structured interviews grounded theory was applied to start with empirical specifics and move towards general statements (Denzin and Lincoln, 2011). To extract findings *open coding* (Strauss et al., 1998) was followed, where the researcher generates categories fitting the data in relation to a general issue of concern (Bryman et al., 2002). "Codes are tags or labels for assigning units of meaning to the descriptive or inferential information compiled during a study" (Miles and Huberman, 1994, p. 56). "During open coding, data are broken down into discrete parts, closely examined, and compared for similarities and differences" (Strauss et al., 1998, p. 102). These discrete parts identified are called *concepts*. The goal is to derive *categories* from the concepts by comparing data from each case (Strauss et al.,

Cycle Steps	**Problem-solving Activity**	**Research Activity**
0. Definition of RQ	-	Define RQ
1. Definition of F, M_R, M_{PS}	(Re-)Define M_{PS} (Sprint)	(Re-)Define F, M_R
2. Update problems	Update problems of P (Daily, Retro)	Update observations of A
3. Take action	Act on problems of P (Sprint)	Collect and analyze data
4. Reflect on success	Reflect on success of actions (Retro)	Draw conclusions for A
5. Update findings	Update findings to P, M_{PS} (Retro)	Update findings to A, M_R
6. *Proceed with step 1 until exit criterion is met*		

Table 5.4 – The dual action research cycle and its implementation in scrum as used in case CrossTown.

1998). The "concepts that reach the status of a category are abstractions; they represent not one individual's or group's story but rather the stories of many persons or groups" (Strauss et al., 1998, p. 145).
Following this rationale the data is coded to find concepts using open coding. From these concepts both categories (ADAPT guidelines) and subcategories (ADAPT practices) are derived. Through *axial coding* subcategories (practices) are related to their categories (guidelines), termed "axial" because the category is the axis and the subcategories are linked to that axis (Strauss et al., 1998). While the guidelines (categories) are less concrete and overspanning, the practices (subcategories) give it greater explanatory power "reassembling data that were fractured during open coding" (Strauss et al., 1998, p. 124).
For single-case analysis in case Continental the ATLAS.ti qualitative data analysis software has been used for coding, but the other cases as well as the cross-case analyses have been conducted using spreadsheets. In case Continental the interview transcripts were approved by the interview partner before they were used for analysis.

Observation: In case NoTimeshift the author participated in all meetings at least once as a silent observer and took field notes, pictures and audio recordings.

Documentation: In all cases archival records, documents and physical artifacts (paper board and sticky notes) were gathered to corroborate and triangulate evidence found from the interviews (cases NoTimeshift and Continental) and action research (case CrossTown). For case Continental documents were collected but the author was granted only limited access to archival records.

Feedback sessions: In all three cases feedback sessions followed the case

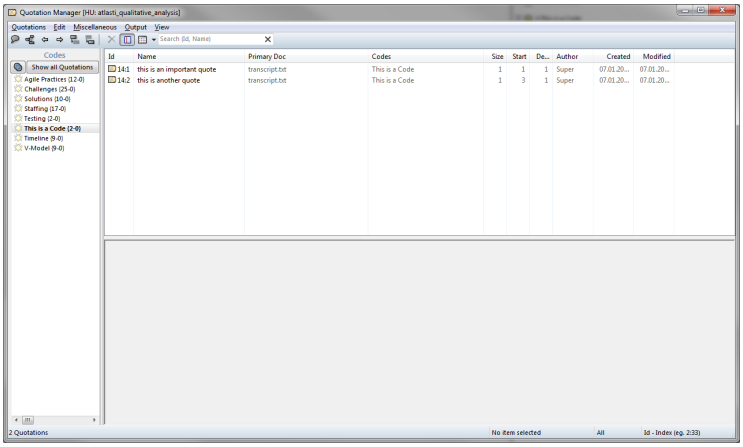


Figure 5.3 – Atlas.ti qualitative data analysis software enables powerful yet simple management of quotations/codes across different input source files. This shows a sample of the quotation manager with codes and its respective quotations.

study analysis where results were presented to gain feedback, corrections and possibly further input.

So in short, the data collection methods varied for the different cases. But what is shared by all cases during data analysis is that open coding was used to code the data from each case, resulting in the identification of concepts. These concepts can be regarded as practice (subcategory) or guideline (category) candidates. From an individual case point of view these candidates are just concepts, it takes a cross-case analysis to derive categories and subcategories, i.e. the ADAPT guidelines and practices, after the single-case analyses. Concepts are of higher abstraction and more general and are thus more likely to be suitable for guidelines/categories, while more concrete concepts are likely to be suitable practices/subcategories. The following steps are followed to arrive at a final set of practices and guidelines in the cross-case analysis (detailed execution cf. Chapter 6):

1. Divide all identified concepts by their level of abstraction into two groups: possible future categories (”guidelines”) and possible future subcategories (”practices”).

2. For each group arrange concepts with similar properties.

3. Add all concepts (in their groups from step 1) to a spreadsheet, assign each concept an ID for better future reference.

    a) Group ”guidelines”: Merge the identified duplicate concepts and concepts with similar properties to a category with empirical support in

at least two cases. Concepts without empirical support in more than one case are discarded.

b) Group "practices": Merge the identified duplicate concepts and concepts with similar properties to a subcategory with empirical support in at least two cases. Concepts without empirical support in more than one case retain their concept status.

4. Link all guidelines to one or more of the three DSD challenge types coordination, control and communication such that linked guidelines mitigate the challenge.

5. Link all practices (subcategories) to their guidelines (categories) such that practices help implement a guideline and give it further clarification and specification.

6. Also link remaining practice concepts (which have empirical support in only one case and were thus not merged into subcategories) to the guidelines but clearly mark them as conceptual practices, separated from the other *full practices* within the ADAPT framework.

There is a small but important difference in creating the guidelines in contrast to practices: practice concepts are included into the ADAPT framework as a separate group of practices (*conceptual practices*) while guideline concepts are not and are thus discarded. The rationale behind is that while it could be worthwhile to try out a practice concept in a project because it may fit a given project environment well, it is not advisable to implement one's process according to a guideline concept which has empirical support in only one case because the objective of ADAPT is to implement all ADAPT guidelines to any given process implementation, hence there is no place for guideline concepts. By linking the guidelines to the three DSD challenge types it is achieved that "categories are interrelated into a larger theoretical scheme" (Strauss et al., 1998, p. 146) that is the three-tiered ADAPT framework and thus the emerging theory of this dissertation.

5.2 Case CrossTown

5.2.1 Background

This case covers software development within an organization spread across two sites in Vienna, Austria. Before the beginning of the case study, the development of the product had started as an R&D (research and development) project on a single site for two years already with the goal of evaluating several technologies for the interplay of hardware devices and a web administration (cf. Figure 5.4). In the 15-month case study period, the goal was to turn the R&D prototype into a deliverable product. With the beginning of full-scale

product development, the team size doubled and project personnel were distributed across two sites due to space constraints. The team members were split into a development and a test site forming several fully distributed cross-functional teams (Sutherland et al., 2007), i.e. teams integrated across both sites. 27 two-week sprints were analyzed over the course of 15 months.

Table 5.5 lists all project members with regard to role and site. The following scrum roles were present: one product owner (PO) and two scrum masters (SM).

Co-Developers	Developer	Tester	Scrum Master	Product Owner	Sum
Development Site	11	0	1	1	13
Testing Site	0	5	1	0	6
Overall	11	5	2	1	19

Table 5.5 – Team sizes distributed across two sites in case CrossTown.

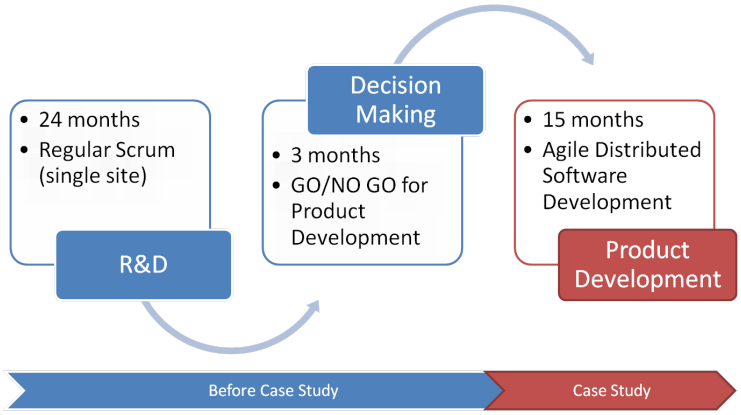


Figure 5.4 – Case background timeline.

5.2.2 Challenges

The initial process implementation suffered from a strong focus on the former R&D site, the development site, since the testing site was a new addition to the process after the initial two-year R&D phase. The development site worked in sprints without the direct involvement of the test site, which resulted in test-ready stories at the end of the sprint instead of deployment-ready ones. Furthermore, the teams had to deal with technically complex implementations regarding the interplay of hardware devices and software and decided to split in closely collaborating on-site micro teams with 3-4 team members. The process implementation exhibited various serious problems: A story was regarded

as accepted once it was developed. After that it was released for test, but the focus was on implementing new stories instead of fixing "old" ones. In consequence a customer deployment failed badly due to the low quality of software in sprint 7 (cf. Figure 5.5), which was the turning point to include the test site in the process more properly. The micro teams were extended to consist of 2-3 developers and one (remote) tester. Contact visits also increased, especially in the second week of each sprint the testers joined the developers for an intense story testing and bug fixing session. To make room for the testers, some developers moved either to the test site or worked at home these days. Greater effort was put into a more realistic sprint planning and commitment with the customer shipment always in mind. During the 15-months of agile DSD, the retrospective turned out to be an invaluable tool, especially at high-stress times, to keep process improvement going and thus keep frustration levels low as all team members could speak their mind and propose solutions.

5.2.3 Agile Practices

This section presents the established working process towards the end of the case study with regard to known scrum practices.

Sprint: Two-week iterations were used which were on few occasions prolonged to cope with holidays and customer's deadlines.
Sprint planning: A joint sprint planning was held in person at the development site with all developers and one or two testers (ambassadors) to represent the testing site. The ambassador(s) then traveled back to the test site to discuss the planning results.
Daily scrum: This project environment worked with very closely collaborating micro teams (cf. Figure 5.6) and the practice of daily scrums, although practiced in the beginning, was eventually dropped. The testers contacted the developers directly for information and updates when needed. Vice versa the developers tried to give a heads-up to the testers when possible. Both formal (ticket management system and emails) and informal (instant messaging, chats and phone calls) were extensively used between the two sites to compensate the lack of face-to-face communication.
Scrum of scrums: Although there were several micro teams, no scrum of scrums has been used as it was regarded as an overhead and the communication mechanisms in-place sufficed.
Sprint review and **retrospective**: These meetings were held in a similar fashion as the sprint planning, at the developer's site with one or more ambassadors from the testing site present. For the retrospective, the ambassador(s) made sure to collect positive and negative comments from all members of the testing site in advance and presented them at the retrospective.
Backlog: As the customer preferred to work with milestones, there was a rough set of user stories planned for each milestone, a generally two-month

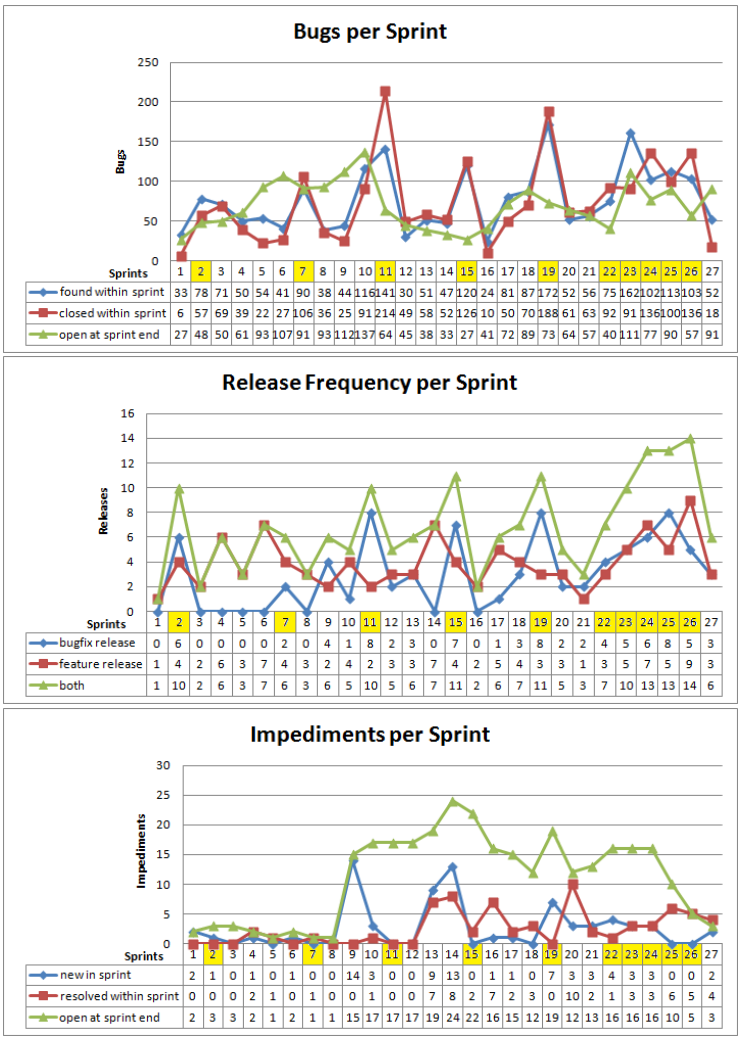


Figure 5.5 – Bug, release and impediment count measured per sprint: sprints 2, 7, 11, 15, 19 and 22-26 denote shipments to the customer. In these sprints one can see a rise in release frequency and closed bugs (positive) but also a rise in new impediments (negative), which shows that these sprints put the process to a test. The regular shipments starting with sprint 22 allowed for a more continuous flow.

time frame. The backlog planning relates to the process described by Hong et al. (2010): The roadmap planning was done for milestones and the detailed planning was done for sprints. Naturally with agile development, the roadmap was subject to change as stories were implemented by priority sprint after sprint.

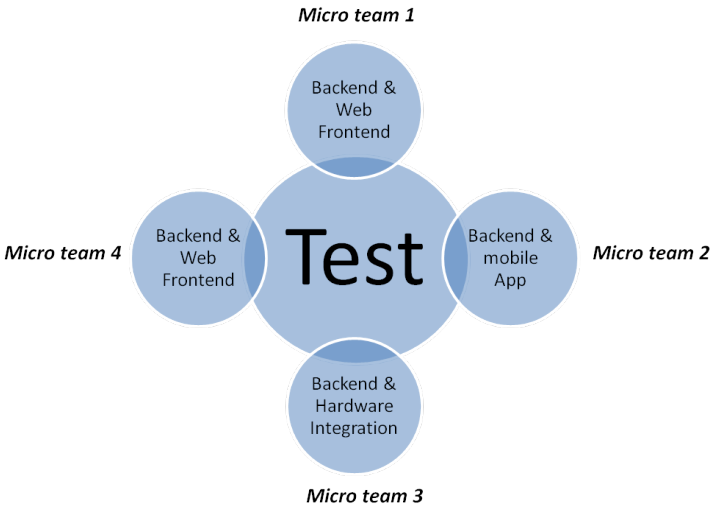


Figure 5.6 – The microteams in place in case CrossTown.

5.2.4 ADAPT Framework Input

This section shows deducted guidelines and practices based on the empirical evidence found in case CrossTown. Table 5.6 shows the problem root cause analysis and the decisions taken, which was an intermediary step before the extraction of the conceptual guidelines (cf. Figure 5.7) and practices (cf. Figure 5.8).

Problem Categories	Root Causes	Decisions
Test-ready features at the end of the sprint instead of deployment-ready ones	• Isolation of test site • Focus on dev site	• Give test site the right to reject stories • Increase contact visits (for face-to-face specification enquiries) • Review of test cases for critical user stories by a developer • Constant availability of team members in instant messaging
Transparency for both sites not accomplished	• Dev site members did not keep issue tracking system up-to-date • Commitment not met (waste in commitment) • Overhead due to rising bug count	• All relevant informal communication needs to be appended to the feature ticket in the issue tracking tool • Contact visits to improve trust and team spirit
Low software quality Lack of focus in sprint	• Feature rush (many 80% ready features valued higher by product owner than fewer 95% ready ones) • Huge technical debt in the interplay of hardware devices and software • Definition of done not well defined • No pressure to deliver potentially shippable code	• Incremental inclusion of test site • Bugfix iteration (should be avoided) • Sunshine cases should work when a story is passed to the tester, so the tester can focus on corner cases • Implement all aspects of a story including non-functional requirements (e.g. stability, performance) • Continuous deployment to customer every sprint
Volatile specification	• Legacy stories from R&D project not defined • Stories not detailed enough before sprint • Stories written at dev site only • Stories too big	• No informal story updates • Meetings with customer before sprint • Involve customer more in the prioritization • Small manageable stories • Increased up-front planning and specification to identify problems, corner cases and impact on existing software early and improve estimation of team • Staffing: new Business Analyst
No up-to-date test cases	• Test cases not ready before stories get pulled into sprint • Specification wrongly interpreted • Informal specification adaptations without the other site's knowledge • Wrong effort estimation due to open questions in specification	• No story updates during sprint • On-demand specification meetings with members from both sites
Process adaptation in DSD is slower and more difficult than in regular collocated scrum	• Harder to propagate changes over multiple sites • More variables and complexities to take into account	Use retro as a driver for continuous process improvement

Table 5.6 – Case CrossTown: identified problems, root causes and the decisions taken.

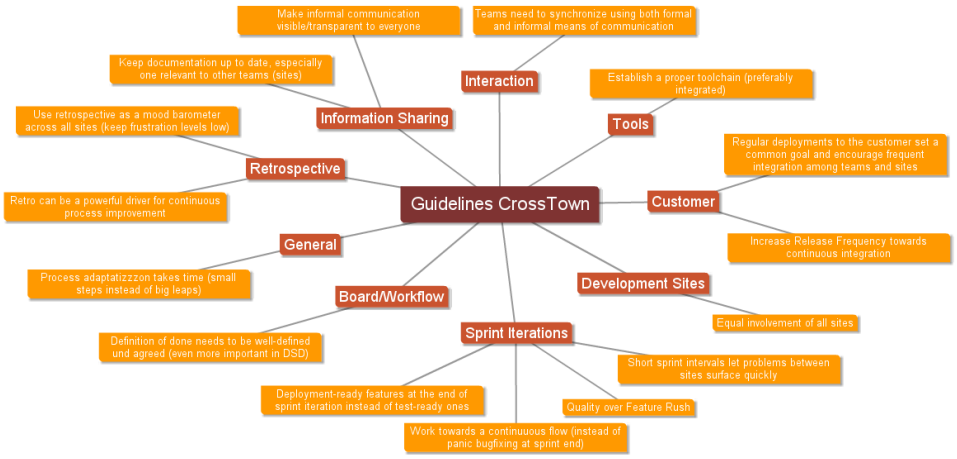


Figure 5.7 – Draft of the deducted guidelines of case CrossTown.

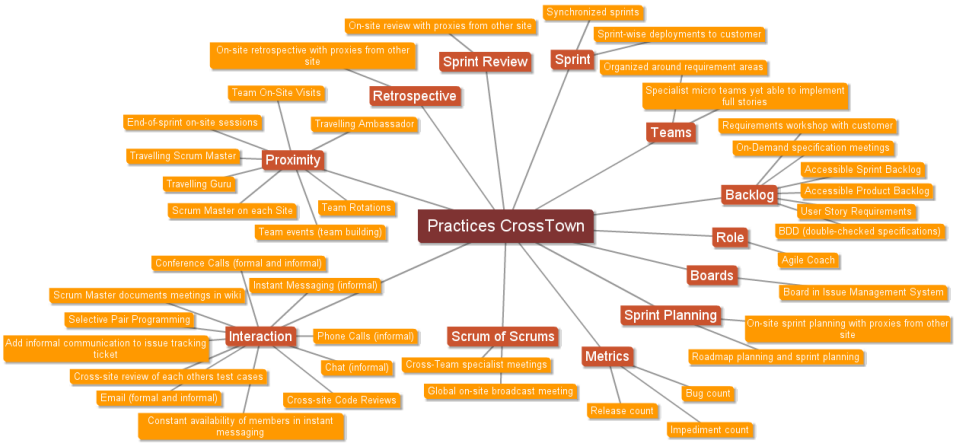


Figure 5.8 – Draft of the deducted practices of case CrossTown.

The case showed that adaptations to the process take longer to take effect because solutions need to be propagated across several sites. Synchronization of team members needs mechanisms such as instant messaging and contact visits to substitute on-site availability. Documentation and knowledge management is more important in agile DSD since even informal enquiries (between a developer and the product owner e.g.) need to be communicated to the other site. The retrospective served as a mood barometer for team members of both sites and was a great driver for process improvement. In the last months of the case study, software has been successfully shipped to the customer each sprint, which underlines the working process adaptations for the given distributed project environment.

5.3 Case NoTimeshift

5.3.1 Background

This case covers two unaffiliated organizations, a main supplier (MainSupp) and an additional supplier (AddSupp), co-developing three product variations of the same code base, resulting in three product owners. Both suppliers have successfully applied regular scrum before and chose to implement an adapted version of scrum to better suit the needs of a DSD environment. The two organizations develop at their own sites in two different cities in Austria, separated by about 300km. MainSupp is a large organization whose IT department is involved in the development of the three software products. It acts as point of contact to customers and provides the bigger part of the development staff. AddSupp is a medium-sized core software development company and a subcontractor to MainSupp for the software development. It complements the MainSupp's development with additional staff and know-how. The teams are all integrated (cross-site) distributed ones (Sutherland et al., 2007).

Table 5.7 shows the distribution of team members over the two suppliers. The MainSupp has one product owner (PO) for each software product and three scrum masters (SM) serving three teams. The AddSupp does neither have a PO nor an SM on site.

Co-Developers	**Developer**	**Tester**	**Scrum Master**	**Product Owner**	**Sum**
Main Supplier (MainSupp)	11	3	3	3	20
Additional Supplier (AddSupp)	8	2	0	0	10
Overall	19	5	3	3	30

Table 5.7 – Team sizes distributed across two sites in case NoTimeshift.

5.3.2 Challenges

Although both companies had previous experience with applying regular scrum successfully, both lacked experience in DSD. Transparency was a big issue between the two suppliers and low quality video conferences and little available documentation for AddSup handicapped communication and coordination in the first months. There was no high level overview of the progress of all three teams available to everyone since paper scrum boards and burndown charts were used. All three scrum teams were staffed by members of both suppliers, yet all product owners and scrum masters were based on the MainSupp's site.

"They [MainSupp] are not used to work with other suppliers collaboratively and hence naturally the process is focused on their staff and site. They need to learn that there needs to be a planning that involves both suppliers because we are not within earshot. There is a lot to learn in both directions." (Scrum Master, AddSupp)

The resulting coordination issues are best described in the words of one of the AddSupp's developers:

"I would love to break down tasks to a decent level, but if we do not know what should be developed exactly, that is hard to achieve." (Developer, AddSupp)

The situation eventually improved with heavy use of video conferencing to complement the scrum process meetings.

5.3.3 Agile Practices

This section presents the established working process towards the end of the case study with regard to known scrum practices.

Sprint: Two-week sprint iterations were used with reviews every sprint and planning and retrospective meetings only every other sprint.
Sprint planning was a two-tiered process and covers two sprints. The first tier involved planning at the MainSupp's site with one ambassador from AddSupp present. The second-tier planning continued at the AddSupp's site. The ambassador returned with pre-estimated user stories which were then broken down into tasks by AddSupp's developers. When a developer accepted a task, he adjusted the original estimation of the MainSupp to his own. An updated planning spreadsheet was then returned to MainSupp.
Daily scrum: Each scrum team held a daily video conference meeting, where respective team members of the MainSupp and AddSupp participated.
Scrum of scrums: One of the AddSupp's developers traveled to the MainSupp's site once a week for face-to-face updates and discussions. Both sites also engaged in their own intra-site coordination scrum of scrum right after the daily scrum. The testers of each team also felt the need to coordinate across all teams in their own scrum of scrums.
The **sprint review** was held jointly for all the distributed teams. It was primarily held at the MainSupp's site with one or two *proxies* from AddSupp on site. In contrast to the sprint planning, for the review and retrospective, AddSupp joined directly via video conference. The review consisted of story demonstrations and discussions about different areas of the current product increment.
The **retrospective** followed the review in the same setup, but only every other sprint.

Backlog: Each product owner maintained a product backlog on the MainSupp's site for his product. AddSupp worked only with the sprint backlog, which was a planning spreadsheet created during the two-tiered planning process.

5.3.4 ADAPT Framework Input

This section shows deducted guidelines and practices based on the empirical evidence found in case NoTimeshift. Table 5.8 shows the problem root cause analysis and the decisions taken, which was an intermediary step before the extraction of conceptual guidelines (cf. Figure 5.9) and practices (cf. Figure 5.10).

Problem Categories	Root Causes	Decisions
Transparency	• Suppliers not Collocated • Communication Issues • Little Documentation • No Overview over All Teams	• Installment of Video Conference Daily Scrum, On-site Scrum of Scrums, Phone Calls and Screen-sharing • Paper boards for each site
Commitment	• Commitment Fails with Insufficient Planning • Commitment Fails with Late Planning • Commitment Fails with Frequent Changes • Little Respect for Iterations	Invite AddSupp more into the process with contact visits and two-tiered planning
Planning	• Late Actual Beginning of Sprint • Little Participation of AddSupp • Little Information for AddSupp	Two-tiered planning with ambassador from other site first, then planning on the other site with returning ambassador.
Estimation & Predictability	• User Story Estimation in Hours • Pre-estimations by MainSupp • No Proper Sprint Velocity • Further Impediments for Better Predictability	*Unresolved within the case study period*
Self-Organizing Teams	• No Official Scrum Roles at the AddSupp • No Joint Estimation and Planning • Inter-Company Distribution of Team Members • Tasks Assigned to Team Members • Estimations Based on Individuals • Cross-Team Working Agreements	• Scrum masters evolved at AddSupp's site • Cross-Team QA Scrum
Tools	• Tools Lack Scrum Compatibility • Limited Remote Access for AddSupp • Paper Scrum Board and Burndown Chart	Paper boards for each site

Table 5.8 – Case NoTimeshift: identified problems, root causes and the decisions taken.

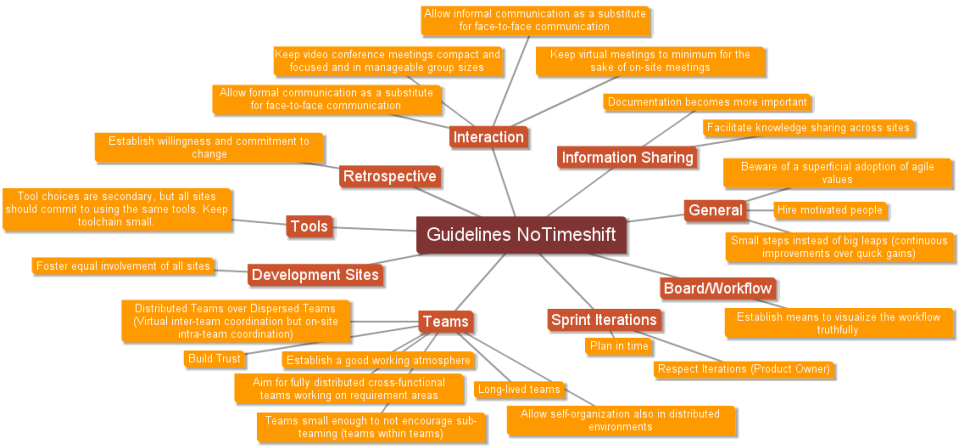


Figure 5.9 – Draft of the deducted guidelines of case NoTimeshift.

Figure 5.10 – Draft of the deducted practices of case NoTimeshift.

The case showed that inter-organizational co-development added another layer of complexity. Even though both suppliers have experience with collocated scrum, the transition to distributed scrum was not an easy one. The larger the organization, the harder it is to introduce changes and the more time it takes. This could especially be observed with the MainSupp, where changes took long to be realized compared to the AddSupp, which is smaller in size. Compromises had to be made to deal with organizational impediments such as the switch to paper boards.

5.4 Case Continental

5.4.1 Background

This case covers a customer based in EU country 1 (EUC1) and a supplier based in both EU country 1 and two further European countries (EUC2, EUC3), which are withhold for privacy reasons. The project had a rushed start because of a tight schedule and deadline. Moreover it was one of the first projects to be done in this setup and there were organizational restrictions on the supplier's side to use the V-model (Boehm, 1979) also in this distributed development. Still, agile practices were eventually used in an effort to improve collaboration among the three development sites. The project spanned 9 months. There were no time zone issues and cultural issues can be regarded as minor between the three European countries (EUC) due to their proximity. Table 5.9 shows the detailed project's staffing setup across the three sites. Since the overall process was the V-model, there was no scrum master in place, yet several PMO roles such as project manager, test manager, solution architect and change manager.

Co-Developers	**Developer**	**Tester**	**Scrum Master**	**Product Owner**	**PMO**	**Sum**
EUC1	6	1	0	4	3	14
EUC2	12	5	0	1	1	19
EUC3	4	1	0	0	1	6
Overall	22	7	0	5	5	39

Table 5.9 – Team sizes distributed across three sites, in three different European countries, in case Continental.

5.4.2 Challenges

Three problem categories were identified in this case, the inflexibility of the V-model, weak feedback loops and collaboration with the customer and further intra-supplier issues. The V-model was implemented because the supplier had many years experience with it, but the project had been under a very tight schedule from the very beginning and the V-model did not allow enough flexibility to cope with unforeseen problems (both technical and organizational in nature). The supplier's internal problems, being distributed across three sites, were in the end successfully mitigated by employing several agile meetings such as daily scrum and daily scrum of scrums that helped bring the project back on track. The customer collaboration could only be improved up to a certain point, because it is not an integral part of the V-model. The customer felt left out of feedback loops and was not fully content with the final product. A mitigation strategy was to make a dashboard available to the customer for live test reports, but it only came late in the project's timeline.

This project also served as a ramp-up for future collaborations and as such was a great learning experience for all parties. The customer and the supplier decided to alter the process for future projects in favor of the implementation of more agile practices.

5.4.3 Agile Practices

There was no sprint, sprint planning or backlogs in use as the V-model worked with milestones and formal reviews and a fixed set of requirements with little flexibility other than change requests. Still, a few agile practices were implemented as a crucial improvement to the development collaboration: there was a 15 minutes **daily scrum** within teams and another daily **scrum of scrums** including development lead, PMO, solution architect and test manager. Furthermore there were also weekly meetings between development lead of EUC2 and specialists from EUC1 and EUC3 and two-weekly meetings of all teams and stakeholders to spread knowledge on project's status. In short, communication was very important for the supplier internally but not towards the customer. There was no retrospective meeting, only a one-time lessons-learned workshop after the completion of the project. The implemented scrum-style meetings were an effort to improve collaboration and interaction in the distributed setting.

5.4.4 ADAPT Framework Input

This section shows deducted guidelines and practices based on the empirical evidence found in case Continental. Table 5.10 shows the problem root cause analysis and the decisions taken, which was an intermediary step before the extraction of conceptual guidelines (cf. Figure 5.11) and practices (cf. Figure 5.12).

Problem Categories	Root Causes	Decisions
Inflexible V-model	• The supplier had many years experience with the V-model thus making it a constraint to be used also for this distributed project and hindering the application of an agile process, which would have been preferred by the customer • Initial use cases do not get refined as project involves and thus little collaboration between product owner and testers due to strict phases, especially during the "lower" development phases of the V-model • Dealing with formalities seems to slow the project down	• Find a compromise to restrictions laid upon the project by the V-model by allowing communication and interaction, especially during the development phase, to be more agile, resulting in a less formal agile development phase, yet the usual formal phases for integration and system testing in the V-model • Future projects between the customer and the supplier should incorporate more agile elements and feedback loops to the customer
Customer collaboration	• Ramp-up problems being first project in collaboration of customer and supplier • High role fluctuation and weakly enforced contact interfaces • Customer feels shut out and does not know real project status, i.e. no customer contact during development, only in the beginning and end of project and also no test reports	• Invite customer into the tool chain and provide a dash board with live status reports • Keep well-defined communication points of contact steady between customer and supplier during project execution • Lessons Learned Workshop after project
Internal supplier collaboration	• Concerns regarding efficiency of collaboration: expensive on-shore developers vs less expensive off-shore ones that may need considerably more time • All technical decisions have to be approved by EUC1 site	• Implement different formal ways of communication (daily scrums, daily scrum of scrums, Weekly specialist meetings, two-weekly general meetings of all teams and stakeholders (without customer) to discuss project's status • Implement different informal ways of communication (face-to-face on site, conference calls and instant messaging between sites) • Regular contact visits from European EUC2 and EUC3 sites to main EUC1 site, also during system, integration and acceptance testing in the end • Language courses for EUC2 and EUC3 sites allowed written conversation in the foreign language, all new documentation in English

Table 5.10 – Case Continental: identified problems, root causes and the decisions taken.

Although no full scrum process has been employed in this case, it is an interesting finding that agile practices such as daily scrum and scrum of scrums were used to get the development process to work and establish frequent communication. Even within the strict frame of a V-model, the agile practices managed to allow a little bit of flexibility from the inside and foster collaboration. The retrospective workshop after the finished project showed that the involved parties, especially the customer, would highly prefer to move to a more agile process to increase transparency and shorten feedback cycles as the customer felt largely uninvolved and out of the picture as he was kept out of the intra-agile feedback loops.

Figure 5.11 – Draft of the deducted guidelines of case Continental.

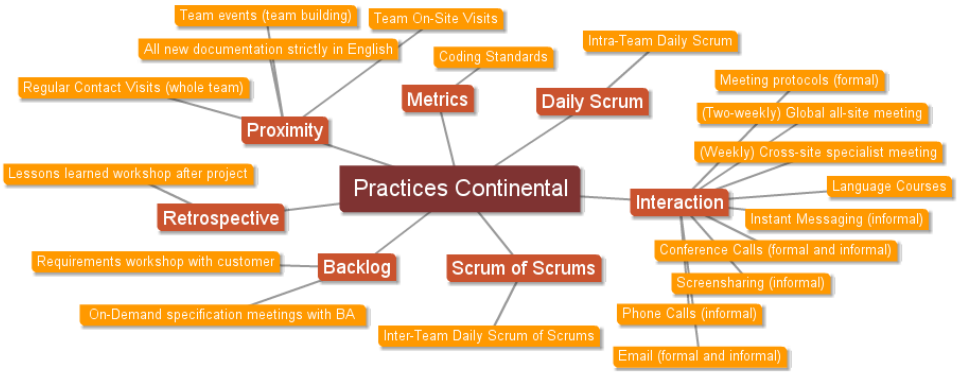


Figure 5.12 – Draft of the deducted practices of case Continental.

5.5 Conclusion

The multiple-case study let the following research question guide data collection and analysis:

RQ4a. *What process design guidelines and best practices can be formulated to increase the chances of a successful agile process implementation in distributed environment?*

RQ4b. *Do the different distribution scenarios affect the implementation of agile practices?*

This chapter presented single-case results of this multiple-case study and can thus only answer RQ4 partly. The analysis continues with a cross-case analysis in Chapter 6 and the final answer will be given in Chapter 7. The multiple-case study spanned an overall timeframe of several years and involved the following distribution scenarios:

- Case CrossTown: sites distributed within one city, spanning two districts
- Case NoTimeshift: sites distributed within one country, spanning two cities
- Case Continental: sites distributed within one continent, spanning three countries

Through careful analysis multiple *conceptual* guidelines and practices (i.e. in a first draft) were extracted from the empirical evidence of each case which are subject to further analysis and a cross-case comparison in the next chapter:

- Case CrossTown: 15 guidelines, 40 practices
- Case NoTimeshift: 22 guidelines, 35 practices
- Case Continental: 12 guidelines, 19 practices

So the individual single-case analysis resulted in total in 49 guideline and 94 practice *concepts*. The concepts are analyzed (including splits, merges and redefinitions) during the cross-case analysis in Chapter 6.

CHAPTER 6

Cross-Case Analysis: Building the ADAPT Framework v1.0

At the time of finalizing this dissertation, this chapter's work was in preparation for publication (to appear, cf. Section 1.5).

Contents

With the multiple-case study research approach, the author looks for two kinds of findings (Patton, 2002, p. 235):

1. High quality, detailed descriptions of each case, useful for documenting uniqueness (cf. Chapter 5)

2. Important shared patterns that cut across cases and derive their significance from having emerged out of heterogeneity (this chapter)

This chapter describes the execution of the cross-case analysis, commencing in Section 6.1 with a short overview of the agile practices in use in a cross-case manner but arranged similar to the presentation of the individual cases. Section 6.2 then proceeds with the actual extraction of practices that have emerged from the three cases CrossTown, NoTimeshift and Continental for

the first iteration of the ADAPT framework, which will result in a set of *full practices* that have empirical support in at least two out of three cases and a set of *conceptual practices* that only have empirical support in one case. Section 6.3 continues with the next level of the ADAPT framework, the guidelines. The pieces are then put together in final Section 6.4 of this chapter resulting in the first full version of the framework, the ADAPT framework v1.0.

The actual order in which the cross-case analysis has been conducted is defined in Section 5.1.6, where guidelines have been created before the practices. However, for allowing a compact presentation of the ADAPT framework within this thesis and better accessibility to the reader, the report in this chapter follows a bottom-up presentation, from practices (cf. Section 6.2) to guidelines (cf. Section 6.3) to the complete picture, the full ADAPT framework also including challenge types (cf. Section 6.4).

6.1 Cross-Case Summary

Sprint: Cases CrossTown and NoTimeshift used two-week sprints, while Continental worked with a V-model and thus did not use iterations. No DSD methods were applied in any of the cases to alter the sprint practice.
Sprint Planning: Cases CrossTown and NoTimeshift applied a similar approach using an ambassador and focused the planning physically on one site only. In case NoTimeshift the other site also held another (second-level) planning following the return of the ambassador. Case Continental worked with a V-model and up-front heavy-weight requirements and planning. The DSD enhancement of adding a travelling ambassador worked well in both cases and was a substantial improvement for a working scrum process.
Daily Scrum: CrossTown worked in micro teams and dropped the practice in favor of using several other means of formal and informal communication (ticket management system, phone calls, emails, chat, instant messaging and a wiki). NoTimeshift and Continental both implemented the practice of a daily scrum, for case NoTimeshift with the help of video conferences (integrated distributed teams) and on-site for case Continental (isolated distributed teams).
Scrum of Scrums: CrossTown also did not use scrum of scrums due to the same rationale as not using daily scrums. NoTimeshift used scrum of scrums for on-site inter-team coordination. Continental applied several scrum of scrums for cross-team and cross-site coordination by means of video conferencing and screen sharing sessions.
Sprint Review and **Retrospective:** These two practices have been applied in the same setup within the respective case of CrossTown and NoTimeshift with the introduction of a travelling proxy/ambassador similar to the sprint planning acting on behalf of the colleagues not present in case CrossTown and serving as a proxy (with the team joining in video conference) in case

NoTimeshift. Continental used a V-model with its respective phases and reviews.
Backlog: Case CrossTown used a product backlog with coarse-grained low-priority stories and fine-grained high-priority ones, planned for the next "milestone", usually a time span of about 4-5 sprints, which would then each have a regular sprint backlog. Case NoTimeshift had the product backlog handled by the main site (as consequence of all the product owners residing there) and handed only the sprint backlog to the additional site for co-development by both sites. No DSD practices have been used to facilitate this practice other than ticket management systems. Continental used a V-model with a pre-defined release plan and no backlog practices.
Summary of DSD Enhancements: In this multiple-case study the following DSD practices supported the application of scrum practices in a distributed environment: contact visits by a travelling proxy/ambassador (sprint planning, review and retrospective), different types of formal and informal means of communication such as video conferences, phone calls, chat, emails, screen sharing sessions, ticket management systems and wikis (sprint planning, review, retrospective, daily scrum, scrum of scrums, backlog) in order to mitigate the lack of face-to-face communication in DSD environments.

6.2 Practices

Figure 5.1 presented the analysis process including the cross-case analysis. Each individual case resulted in an extracted set of working practices (cf. Figures 5.8, 5.10 and 5.12). From an ADAPT framework's point of view these reported working practices are only regarded as conceptual practices. This section now compares the three sets of practices to find common (full) practices.

The following steps were employed to arrive at the final set of practices (cf. Section 5.1.6).

1. Arrange practice concepts in groups with similar properties (visualized in Figures 5.8, 5.10 and 5.12).

2. Add all practices with the identified similar concepts to a spreadsheet, assign each practice an ID for better future reference (cf. Table 6.1).

3. Merge the identified duplicate concepts and concepts with similar properties to a full practice (in terms of open coding: a subcategory) with empirical support in at least two cases. Concepts without empirical support in more than one case retain their conceptual status (in terms of open coding: a concept).

4. Link all practices to the guidelines[4] such that practices help implement a guideline and give it further clarification and specification.

5. Also link remaining practice concepts (which have empirical support in only one case and were thus not merged into subcategories) to the guidelines but clearly mark them as *conceptual practices*, separated from the regular *full practices* within the ADAPT framework.

Table 6.1 shows step 2, where all practices are added to a spreadsheet. The groups to intermediary classify concepts were: proximity, interaction, sprint, backlog, teams, boards, sprint planning, retrospective, sprint review, metrics, daily scrum, scrum of scrums and role. They are only used to identify possible concepts for merging as a temporary help and have no further semantic meaning to the ADAPT framework.

ID	Merge Groups	Single-Case Practices	Source
CT.P.1	Proximity	Travelling Scrum Master	CrossTown
CT.P.2	Proximity	End-of-sprint on-site sessions	CrossTown
CT.P.3	Proximity	Travelling Ambassador	CrossTown
CT.P.4	Proximity	Travelling Guru	CrossTown
CT.P.5	Proximity	Team On-Site Visits	CrossTown
CT.P.6	Proximity	Team Rotations	CrossTown
CT.P.7	Proximity	Team events (team building)	CrossTown
CT.P.8	Proximity	Scrum Master on each Site	CrossTown
NT.P.16	Proximity	Scrum Master on each Site	NoTimeshift
NT.P.17	Proximity	Team On-Site Visits	NoTimeshift
NT.P.18	Proximity	Team Rotations	NoTimeshift
NT.P.19	Proximity	Travelling Ambassador	NoTimeshift
NT.P.20	Proximity	Travelling Guru	NoTimeshift
NT.P.21	Proximity	Travelling Scrum Master	NoTimeshift
NT.P.22	Proximity	Team events (team building)	NoTimeshift
C.P.13	Proximity	Regular Contact Visits (whole team)	Continental
C.P.14	Proximity	All new documentation strictly in English	Continental
C.P.15	Proximity	Team events (team building)	Continental
C.P.16	Proximity	Team On-Site Visits	Continental
CT.P.9	Interaction	Cross-site Code Reviews	CrossTown
CT.P.10	Interaction	Cross-site review of each other's test case	CrossTown
CT.P.11	Interaction	Add informal communication to issue tracking ticket	CrossTown
CT.P.12	Interaction	Email (formal and informal)	CrossTown
CT.P.13	Interaction	Instant Messaging (informal)	CrossTown
CT.P.14	Interaction	Chat (informal)	CrossTown
CT.P.15	Interaction	Phone Calls (informal)	CrossTown
CT.P.16	Interaction	Constant availability of members in instant messaging	CrossTown
CT.P.17	Interaction	Scrum Master documents meetings in wiki	CrossTown
CT.P.18	Interaction	Conference Calls (formal and informal)	CrossTown
CT.P.40	Interaction	Selective Pair Programming	CrossTown
NT.P.5	Interaction	Screensharing (informal)	NoTimeshift
NT.P.6	Interaction	Phone Calls (informal)	NoTimeshift
NT.P.7	Interaction	Email (formal and informal)	NoTimeshift
NT.P.8	Interaction	Instant Messaging (informal)	NoTimeshift
NT.P.9	Interaction	Meeting protocols (formal)	NoTimeshift
NT.P.10	Interaction	Scrum Master documents meetings in email	NoTimeshift
NT.P.11	Interaction	Cross-site review of each other's test case	NoTimeshift
NT.P.12	Interaction	Conference Calls (formal and informal)	NoTimeshift
C.P.1	Interaction	(Weekly) Cross-site specialist meeting	Continental
C.P.2	Interaction	(Two-weekly) Global all-site meeting	Continental
C.P.3	Interaction	Conference Calls (formal and informal)	Continental
C.P.4	Interaction	Instant Messaging (informal)	Continental
C.P.5	Interaction	Language Courses	Continental

[4]Guidelines are presented in the next Section 6.2.

C.P.6	Interaction	Email (formal and informal)	Continental
C.P.7	Interaction	Phone Calls (informal)	Continental
C.P.8	Interaction	Meeting protocols (formal)	Continental
C.P.9	Interaction	Screensharing (informal)	Continental
CT.P.19	Sprint	Sprint-wise deployments to customer	CrossTown
CT.P.20	Sprint	Synchronized sprints	CrossTown
NT.P.31	Sprint	Synchronized sprints	NoTimeshift
NT.P.35	Sprint	Frequent deployments to customer	NoTimeshift
CT.P.21	Backlog	On-Demand specification meetings with PO	CrossTown
CT.P.22	Backlog	BDD (double-checked specifications)	CrossTown
CT.P.23	Backlog	Accessible Product Backlog	CrossTown
CT.P.24	Backlog	Accessible Sprint Backlog	CrossTown
CT.P.25	Backlog	Requirements workshop with customer	CrossTown
CT.P.37	Backlog	User Story Requirements	CrossTown
NT.P.27	Backlog	Accessible Product Backlog	NoTimeshift
NT.P.28	Backlog	Accessible Sprint Backlog	NoTimeshift
NT.P.29	Backlog	BDD	NoTimeshift
NT.P.33	Backlog	User Story Requirements	NoTimeshift
C.P.17	Backlog	Requirements workshop with customer	Continental
C.P.19	Backlog	On-Demand specification meetings with BA	Continental
CT.P.26	Teams	Specialist micro teams yet able to implement full stories	CrossTown
CT.P.27	Teams	Organized around requirement areas	CrossTown
NT.P.30	Teams	Organized around requirement areas	NoTimeshift
CT.P.28	Boards	Board in Issue Management System	CrossTown
NT.P.13	Boards	Paper boards on each site	NoTimeshift
NT.P.14	Boards	One board for all teams on a site	NoTimeshift
CT.P.29	Sprint Planning	On-site sprint planning with proxies from other site	CrossTown
CT.P.30	Sprint Planning	Roadmap planning and sprint planning	CrossTown
NT.P.25	Sprint Planning	Multi-level planning (two on-site plannings)	NoTimeshift
NT.P.26	Sprint Planning	Allow time for research and learning	NoTimeshift
CT.P.31	Retrospective	On-site retrospective with proxies from other site	CrossTown
NT.P.24	Retrospective	Video-conference Retro with proxies	NoTimeshift
C.P.10	Retrospective	Lessons learned workshop after project	Continental
CT.P.32	Sprint Review	On-site review with proxies from other site	CrossTown
NT.P.23	Sprint Review	Video-conference Review with proxies	NoTimeshift
CT.P.33	Metrics	Bug count	CrossTown
CT.P.34	Metrics	Release count	CrossTown
CT.P.35	Metrics	Impediment count	CrossTown
NT.P.15	Metrics	Accessible Burndown chart	NoTimeshift
NT.P.34	Metrics	Code Quality	NoTimeshift
C.P.18	Metrics	Coding Standards	Continental
NT.P.1	Daily Scrum	Video Conference Daily	NoTimeshift
NT.P.2	Daily Scrum	Audio Conference Daily	NoTimeshift
C.P.11	Daily Scrum	Intra-Team Daily Scrum	Continental
CT.P.38	Scrum of Scrums	Cross-Team specialist meetings	CrossTown
CT.P.39	Scrum of Scrums	Global on-site broadcast meeting	CrossTown
NT.P.3	Scrum of Scrums	Cross-Team QA Scrum	NoTimeshift
NT.P.4	Scrum of Scrums	On-site Scrum of Scrums with Ambassador	NoTimeshift
C.P.12	Scrum of Scrums	Inter-Team Daily Scrum of Scrums	Continental
CT.P.36	Role	Permanent agile coach	CrossTown
NT.P.32	Role	Temporary agile coach	NoTimeshift

Table 6.1 – The table gives an overview of the cross-case practice extraction status after step 2 and shows 94 single-case practices.

After running through all the steps, the final set included 22 practices (cf. Section 6.2.1) that had empirical evidence in at least two out of the three cases and 10 additional conceptual practices (cf. Section 6.2.2) which had only support in one case. The following sections present each of the practices in greater detail and also explain which of the initial conceptual practices have been merged to arrive at the final set. Although the process for arriving at the final set of guidelines (cf. Section 6.3) and the final version of the ADAPT framework (cf. Section 6.4) has not been discussed yet, the following practice

description already includes the linked guidelines and challenge categories to present the complete practices in one spot only (for further reference).

The practices are described in the following way: The description is opened with a short overview of the practice, covering its essence. The section *further elaboration from empirical evidence* explains how the practice was constructed using single-case practices (flavors) of the three cases CrossTown, NoTimeshift and Continental from Table 6.1 including the practice IDs in parentheses. The description concludes with *linked guidelines*, *mitigated challenges* and the *rationale* behind.

6.2.1 Full Practices

Practice P1: Travelling Ambassador

The travelling ambassador is a person which travels between development sites to exchange information and also serves as an on-site proxy for the other distant sites while he is away.

Further elaboration from empirical evidence: The travelling ambassador practice addresses the lack of proximity in DSD. The person travelling can be a specifically nominated person for that purpose (CT.P.3, NT.P.19), the scrum master (CT.P.1, NT.P.21), a guru, i.e. a technical, process or otherwise specialist (CT.P.4, NT.P.20), regular team members (CT.P.5, NT.P.17, C.P.16) or a combination thereof.
Linked Guidelines: G1, G3
Mitigates Challenges: Coordination
Rationale: The practice improves coordination as a working knowledge transfer between sites (G3) is a prerequisite to establish inter-site coordination. Furthermore the ambassador stands up and speaks for the other site, demanding inter-site coordination and a more equal involvement of all the sites (G1).

Practice P2: Full Team On-Site Sessions

The teams gather for regular face-to-face meetings or coding/integration sessions at one site.

Further elaboration from empirical evidence: This practice addresses the lack of proximity in DSD. The teams can gather for special occasions like kickoff or end-of-milestone sessions (CT.P.2) or establish regular contact visits for face-to-face meetings (C.P.13).The concentration on one site allows the combination of all forces for a limited time in a collocated manner, which also helps to establish trust among sites. If one site has not enough space to host all team members, pair programming is a viable solution (two people per

desk) and some team members from the hosting site can take turns in working from home instead (CT.P.2).
Linked Guidelines: G3
Mitigates Challenges: Coordination
Rationale: The practice improves coordination as on-site sessions allow an efficient in-depth knowledge transfer (G3) as well as building trust and getting to know each other on a personal level as a pre-requisite for inter-team coordination.

Practice P3: Team Rotations

Team members rotate such that people can get to know each other in person.

Further elaboration from empirical evidence: This practice addresses the lack of proximity in DSD. Team members can rotate for a single or multiple sprints (NT.P.18) or just several days (CT.P.6). The rotated team members can also serve as an on-site proxy to their now-distant fellow team members (NT.P.18).
Linked Guidelines: G1, G3
Mitigates Challenges: Coordination
Rationale: The practice improves coordination as rotating team members naturally foster knowledge transfer between sites (G3), making it easier to coordinate on with a common and up-to-date knowledge base, and the practice also supports an equal involvement of all sites (G1) when team members get to know other sites better through rotation.

Practice P4: Team Events

All teams gather for a social event to establish personal relationships.

Further elaboration from empirical evidence: This practice addresses the lack of proximity in DSD. Team events can be used to build trust and establish personal relationships and to get to know each other also on a personal level (NT.P.22, C.P.15). This lowers the barrier for frequent communication to exchange information across sites. Team events can also be used as a motivational factor, e.g. to celebrate a milestone or a release (CT.P.7).
Linked Guidelines: G2
Mitigates Challenges: Coordination
Rationale: The practice allows the establishment of trust and getting to know each other on a personal level, which makes inter-site coordination and self-organization of teams (G2) easier with distant team members.

Practice P5: Scrum Master on each Site

A dedicated scrum master needs to be present at all sites to look after the agile process.

Further elaboration from empirical evidence: This practice addresses the lack of proximity in DSD. Case NoTimeshift showed that it is very inefficient for a scrum master to serve distant team members, the situation substantially improved when every site had one (NT.P.16). It is thus necessary to have a scrum master on each site that is dedicated to track and overcome impediments and set impulses to make the agile process work both on-site and cross-site (NT.P.16, CT.P.8).
Linked Guidelines: G1, G5, G10
Mitigates Challenges: Coordination, Control, Communication
Rationale: By having a scrum master on each site, agile values are more likely to be appreciated leading to an equal involvement of all sites (G1) for better coordination at eye level. The scrum masters can furthermore help in visualizing the workflow truthfully (G5), serving as a means of control for all parties. The scrum masters can also assist in improving communication with incremental steps (G10) until a good quality is reached. Following these three guidelines successfully is much more likely if there is a dedicated scrum master on each site to look after the process implementation.

Practice P6: Cross-Site Reviews

Cross-site reviews foster inter-site information exchange and knowledge transfer.

Further elaboration from empirical evidence: This practice addresses the lack of interaction in DSD. As knowledge transfer and information exchange are essential between sites, establishing the practice of cross-site reviews, such as code reviews (CT.P.9) or test case reviews (CT.P.10, NT.P.11), fosters inter-site collaboration and improves overall quality.
Linked Guidelines: G5, G7, G9
Mitigates Challenges: Control, Communication
Rationale: Cross-site reviews achieve two goals. First they serve as a mechanism for controlling software quality (G7). Second they improve communication by adding a new feedback loop (G9) that can be integrated as a step into the workflow (G5).

Practice P7: Multi-Way Informal Communication

Implement different forms of synchronous and asynchronous informal communication to allow self-organization.

Further elaboration from empirical evidence: This practice addresses the lack of interaction in DSD. While formal communication needs to be established in every project setup, informal communication is often not fostered as much. The multiple-case study showed that informal communication is especially important in an agile setup that demands frequent interaction for self-organization. To that end, empirical evidence shows that it is best to allow a variety of different-purpose ways of communication, both synchronous such as phone calls (CT.P.15, NT.P.6, C.P.7), conference calls (CT.P.18, NT.P.12, C.P.3) or chat (CT.P.14) and asynchronous such as instant messaging (CT.P.13, NT.P.8, C.P.4), email (CT.P.12, NT.P.7, C.P.6), wikis (CT.P.17) or issue tracking systems/tickets (CT.P.11).
Linked Guidelines: G2, G3, G8, G10
Mitigates Challenges: Coordination, Communication
Rationale: Individuals and interactions are a central part of the agile manifesto (Fowler and Highsmith, 2001) and thus so is informal communication (G10). It should be possible in multiple ways supported by the tool chain (G8). The practice is very important to allow self-organizing teams (G2) to coordinate and exchange knowledge and information (G3).

Practice P8: Meeting Minutes

All meetings must be summarized in a compact way and made available to all members of all sites.

Further elaboration from empirical evidence: This practice addresses the lack of interaction in DSD. It is very important to track communication in DSD such that information important to all sites does not get lost. The scrum master (NT.P.10) proved especially capable for accomplishing that goal, posting meeting minutes of all scrum meetings (and others) to a wiki (CT.P.17) or mailing list (NT.P.9). In case Continental an overarching project manager took care of meeting minutes (C.P.8).
Linked Guidelines: G3, G4
Mitigates Challenges: Coordination, Control
Rationale: Meeting minutes help transfer knowledge from on-site meetings to other sites (G3) allowing coordination with all information freely available and also serving as a means of control for the other sites. While all meeting outcomes should be shared, it is especially important for the retrospective to document and track steps for continuous improvement and general satisfaction with the current process implementation (G4).

Practice P9: Ad-Hoc Screensharing

Screensharing enables faster problem solving and information sharing.

Further elaboration from empirical evidence: This practice addresses the lack of interaction in DSD. Screensharing allows faster problem-solving especially for programming, integration or configuration problems (NT.P.5, C.P.9) and should be set up in an easy-to-use way to allow ad-hoc usage.
Linked Guidelines: G8, G9
Mitigates Challenges: Communication
Rationale: Sceensharing facilitates problem-centered communication as part of an environment with multiple feedback loops (G9). To be of value, it needs to be easily accessible within the toolchain (G8) for an ad-hoc, hazzle-free usage.

Practice P10: Synchronized Sprints

Synchronize sprints across sites to work towards common goals and not encourage waterfall-thinking.

Further elaboration from empirical evidence: This practice deals with handling iterations in DSD. The multiple-case study shows that is highly preferable to work with the same iterations across sites, i.e. have sprint planning, review and retro all at the same time (CT.P.20, NT.P.31). Otherwise more waterfall-like thinking is encouraged and inter-team coordination gets more complex if teams cannot resolve dependencies within the same sprints but e.g. have to wait for other sprints to complete first.
Linked Guidelines: G1, G5
Mitigates Challenges: Coordination, Control
Rationale: Coordination is easier to achieve if sprints are held in synch across all sites, which makes it harder to drift into unequal involvement of the sites (G1). Synchronized sprints also take complexity out of the workflow visualization (G5) for better control.

Practice P11: Accessible Backlogs

The backlog, in its current state, as a living artifact, needs to be accessible to all members.

Further elaboration from empirical evidence: This practice deals with backlog handling in DSD. Both backlogs, the sprint and the product backlog, need to be accessible to all team members for transparent handling of requirements. The process of grooming the product backlog by the product owner, with the customer or with the team needs to be tracked, historized, transparent and available to everybody (CT.P.23, CT.P.24, NT.P.27, NT.P.28).
Linked Guidelines: G1, G8
Mitigates Challenges: Coordination, Communication
Rationale: Inaccessible backlogs to all parties severely complicate coordina-

tion and foster unequal site involvement (G1). Backlogs should be accessible within the tool chain (G8) for improved communication and flow of information.

Practice P12: Tangible Requirements (BDD)

Behavior Driven Development (BDD) creates human-readable yet executable acceptance criteria.

Further elaboration from empirical evidence: This practice deals with backlog handling in DSD. *Behavior Driven Development (BDD)* involves breaking down requirements to a "Given-When-Then" scheme: "*Given* some initial context (the givens), *When* an event occurs, *Then* ensure some outcomes" (North, 2006). Each step can be automated for running tests with the outcome that each requirement has executable human-readable acceptance criteria. This can be especially powerful in DSD, where all sites work with same test cases and conduct cross-site test case reviews (cf. P6) (CT.P.22, NT.P.29).
Linked Guidelines: G3, G5, G6
Mitigates Challenges: Coordination, Control
Rationale: Tangible requirements improve knowledge transfer (G3) both between sites and the customer (G6) by having testable acceptance criteria as a prerequisite for goal-oriented coordination. The criteria can serve as a valuable addition to the definition of done for controlling the truthful completion of a step in the workflow (G5).

Practice P13: Customer Requirements Workshop

Regular requirements workshop should be held in person with the customer for continuous grooming of the product backlog.

Further elaboration from empirical evidence: This practice deals with backlog handling in DSD. Similar to collocated development, it is important that an initial set of requirements (user stories) is created, discussed and explained in a kickoff workshop with the product owner and the customer (C.P.17). The workshop greatly benefits if members from all sites are able to participate in this workshop and if it is held in regular intervals in order to form a common vision and view on the requirements (CT.P.25).
Linked Guidelines: G3, G6
Mitigates Challenges: Coordination, Control
Rationale: Regular workshops with the customer make sure that the product is heading in the right direction (G6) and serve as a means of both control and coordination between supplier and customer by creating common knowledge (G3).

Practice P14: Feature-Team Organization

Have as small teams as possible to be able to develop features on their own.

Further elaboration from empirical evidence: This practice deals with team setup in DSD. Also in distributed environments it is important that teams remain self-organizing and are able to develop features on their own with as little dependency to other teams as possible. Micro teams have less intra-team coordination to overcome and are more flexible (CT.P.26). It also helps to organize teams around requirement areas so that teams get the chance to become familiar with the area and are thus more productive (CT.P.27, NT.P.30).
Linked Guidelines: G2
Mitigates Challenges: Coordination
Rationale: Having feature teams allows teams to work as autonomous units, being able to implement features on their own with no or little dependency to other teams. Hence intra-team coordination is facilitated and the need for inter-team coordination is minimized (G2).

Practice P15: Multi-Site Multi-Team Workflow Board

The multi-site and multi-team workflow needs to be illustrated in detail, kept up-to date and made available to all sites.

Further elaboration from empirical evidence: This practice deals with handling workflow boards in DSD. A board describing all steps of the workflow truthfully, either electronic (CT.P.28) or on paper (NT.P.13, NT.P.14), needs to be accessible to all team members. While the electronic version is preferable, if opposed by lacking tool support or other reasons, paper boards can also be used facilitated by web cams streaming them to the other sites and updated during the daily scrum. It is important that all work is visualized in the workflow to become the central instrument in the process implementation.
Linked Guidelines: G5, G8
Mitigates Challenges: Control, Communication
Rationale: An overarching multi-team workflow board needs to visualize all steps in the value stream truthfully (G5) and is best integrated tightly into the tool chain (G8). The workflow board serves both as a means of control and as another communication channel, i.e. a feedback loop showing bottlenecks and where other teams stand in the sprint.

Practice P16: Multi-Level On-site Proxy-Planning

Multi-level planning involves an initial planning on one site with proxies and a subsequent planning on the other sites when proxies return.

Further elaboration from empirical evidence: This practice deals with sprint planning in DSD. If it is not feasible for everybody to gather at one site for every sprint planning (cf. P2), sending proxies is a viable option that is preferable over full video conference plannings that tend to be chaotic and of little value to the individual as case NoTimeshift pointed out. Cases CrossTown (CT.P.29) and NoTimeshift (NT.P.25) both featured a planning on one site with proxies attending from the other site. The proxies then returned to their site to hold a second planning meeting, completing the planning cycle and reporting back to get an overall planning result for the sprint.
Linked Guidelines: G1
Mitigates Challenges: Coordination
Rationale: On-site proxies help to create an equal involvement of the sites (G1) during planning and thus allow easier coordination.

Practice P17: Separation of Roadmap and Sprint Planning

Both roadmap and sprint planning should be practiced to keep all sites moving in the right direction.

Further elaboration from empirical evidence: This practice deals with sprint planning in DSD. Every couple of sprints (depending on sprint length) it is important to do a backlog grooming session and conduct roadmap planning, i.e. the planning of what lies ahead, rather than only regular sprint planning for the next sprint, i.e. several weeks. This backlog grooming should include most of the team members (CT.P.30) with either proxies present (cf. P1) or in full on-site sessions (cf. P2). The results must be made available to everyone including the rationale behind (NT.P.27, NT.P.28). This allows all sites to get a heads-up on what lies ahead and move in the same direction.
Linked Guidelines: G1, G3
Mitigates Challenges: Coordination
Rationale: The roadmap and sprint planning helps coordination by creating a common understanding of what lies ahead. The practice improves knowledge transfer (G3) and an equal involvement of the sites (G1).

Practice P18: On-Site Retrospective with Proxies

This multi-site retro involves both proxies on-site and other team members joining via video conference.

Further elaboration from empirical evidence: This practice deals with the retrospective in DSD. In both cases CrossTown (CT.P.31) and NoTimeshift (NT.P.24) the retrospective was held on one site with proxies from the other site present (cf. P1). In case NoTimeshift the other distant team members also attended via video conference. In case CrossTown the relevant argu-

ments for the retrospective were collected by the travelling ambassador prior to the retrospective and presented then on site at the meeting. In any case, the retrospective produced a clear set of problems and the measures discussed to be implemented in the next sprint.
Linked Guidelines: G4, G9
Mitigates Challenges: Control, Communication
Rationale: The retrospective with proxies serves a feedback loop and communication channel (G9). As such it is also a control mechanism and mood barometer for the current process implementation (G4).

Practice P19: On-Site Sprint Review with Proxies

This multi-site sprint review involves both proxies on-site and other team members joining via video conference.

Further elaboration from empirical evidence: This practice deals with the sprint review in DSD. The setup was in both cases similar to the retrospective (cf. P18). In both cases CrossTown (CT.P.32) and NoTimeshift (NT.P.23) the sprint review was held on one site with proxies from the other site present (cf. P1). In case NoTimeshift the other distant team members also attended via video conference. In case CrossTown some sprint reviews were attended by all team members from both sites (cf. P2) and others only by proxies from the other site (cf. P1). In any case, the sprint review involved feature demonstrations of the features implemented within the sprint.
Linked Guidelines: G7, G9
Mitigates Challenges: Control, Communication
Rationale: The review with proxies serves as another feedback loop and communication channel (G9) and a quality control mechanism for the current product increment (G7).

Practice P20: Establish Metrics to Evaluate the Process

Track metrics each sprint and make them available to all sites as an additional means to evaluate the efficiency of adaptations to the process.

Further elaboration from empirical evidence: This practice deals with metrics in DSD. While visualizing the workflow is important (cf. P15), it is also necessary to use metrics to oversee process adaptations. Without metrics it is hard to tell if a measure taken indeed has a positive effect on the process output. The metrics can vary, in case CrossTown three types of metrics were tracked: bug count (CT.P.33), release frequency (CT.P.34) and impediment count (CT.P.35). In case NoTimeshift a common burndown chart (NT.P.15), burning down estimated hours against actual progress, was set in place to track sprint progress. In any case, the metrics need to be made available to

all sites, either an electronic dash board or a webcam streaming the paper equivalent.
Linked Guidelines: G4, G5, G8
Mitigates Challenges: Control, Communication
Rationale: Metrics help track the process implementation for continuous improvement and mitigate the control challenge (G4). They can also help to identify bottlenecks in the workflow (G5) and are best tracked with tool support (G8) for an additional feedback loop improving communication across the sites.

Practice P21: Daily Intra-Team Communication

A compact Daily Scrum is necessary for intra-team coordination.

Further elaboration from empirical evidence: This practice deals with the daily scrum in DSD. Like in collocated scrum, the daily scrum is used for intra-team coordination. For distributed teams (all team members of a team are on the same site) no adaptation is necessary, for dispersed teams (team members of the same team span multiple sites) the daily scrum can be held using an audio or video conference setup (NT.P.1, NT.P.2, C.P.11).
Linked Guidelines: G2, G9
Mitigates Challenges: Coordination, Communication
Rationale: The daily scrum is a central part of intra-team coordination for self-organization (G2) and communication in terms of adding another feedback loop to the process (G9).

Practice P22: Inter-Team Scrum of Scrums

Regular inter-team coordination (SoS) is necessary to resolve dependencies and remove impediments.

Further elaboration from empirical evidence: This practice deals with the scrum of scrums in DSD. Unlike the daily scrum (cf. P21), the scrum of scrums (SoS) in DSD will always involve multiple sites since the focus is inter-team coordination. Case NoTimeshift used weekly SoS meetings on one site with proxies present (NT.P.4) (cf. P2) but also special-purpose cross-team meetings such as a QA SoS with audio/video conference support focusing especially on improving quality assurance and the BDD workflow (NT.P.3) (cf. P12). Case Continental held SoS meetings even daily (C.P.12) for cross-team coordination. In any case, the SoS should help and be limited to dealing with inter-team dependencies (and how to resolve them) and be kept short and compact similar to the daily scrum.
Linked Guidelines: G2, G9
Mitigates Challenges: Coordination, Communication

Rationale: The scrum of scrums is a central part in inter-team coordination for self-organization (G2) and communication in terms of adding another feedback loop to the process (G9).

Practice P23: Frequent Deployments to Customer

Frequent (e.g. sprint-wise) deployments to the customer help establish a common finish line.

Further elaboration from empirical evidence: This practice deals with the sprint iteration in DSD. Achieving the goal of deploying frequently to the customer puts the whole process to a test and keeps everybody focused and goal-oriented. After initial problems, in case CrossTown the teams managed to deploy to the customer each sprint, which was a major improvement (CT.P.19). In case NoTimeshift the deployments spanned several sprints, but they were still an important factor (NT.P.35). Frequent deployments with working software are a sign of a process implementation in good health and shorter delivery times are preferable to longer ones.
Linked Guidelines: G6, G7, G8
Mitigates Challenges: Control, Communication
Rationale: Frequent deployments invite the customer into the process (G6), which serves as a quality control (G7), improves communication to the customer and adds another layer of feedback. The necessity of frequent deployments also demands good usage of an automated build and deployment tool chain (G8).

Practice P24: On-Demand Specification Meetings

The product owner needs to be available to on-demand requirement clarifications also during the sprint.

Further elaboration from empirical evidence: This practice deals with backlog handling in DSD. It frequently happens that new questions arise during the implementation of features within the sprint iteration. It is thus necessary that the product owner takes an active role and is available to further inquiries by team members from all sites during the whole sprint (CT.P.21). If the product owner role is too big for one person to fill, it is also possible to have business analysts (BA) available for on-demand inquiries (C.P.19).
Linked Guidelines: G1, G7
Mitigates Challenges: Coordination, Control
Rationale: The availability of the product owner (or assistants) to all sites (G1) is important to ensure timely responses to inquiries on the backlog during development, which allows the PO better to control the implementation

and also adds quality early in the development (G7), i.e. in the same sprint. This practice improves coordination between the team and the PO.

Practice P25: Cross-Team Specialist Meetings

A cross-team (cross-site) specialist meeting can be a fruitful addition to the regular Scrum of Scrums.

Further elaboration from empirical evidence: This practice deals with the lack of interaction in DSD. In case Continental on top of the regular scrum of scrums, a weekly cross-site specialist meeting was held in a video conference setup to discuss technical solutions and architectural decisions (C.P.1). In case CrossTown technical specialist also met cross-team and on site each sprint to discuss technical solutions and possible technical debt (CT.P.38). Case NoTimeshift worked with daily cross-site QA-dedicated meetings to improve the BDD workflow (NT.P.3).
Linked Guidelines: G2, G9
Mitigates Challenges: Coordination, Communication
Rationale: Cross-team specialist meetings improve the self-organization for inter-team coordination (G2) and can be utilized as another feedback loop for cross-team/cross-site communication (G9).

Practice P26: Global All-Site Broadcast Meetings

Global all-site broadcast meeting are held to share important information with everybody.

Further elaboration from empirical evidence: This practice deals with the lack of interaction in DSD. In case Continental a moderated bi-weekly all-site meeting was held in a video conference setup, where all team members from all sites were asked to participate and which was used to share important organizational information such as changes to the project plan. The objective of the meeting was to broadcast information rather than discuss them due to the big number of participants (C.P.2). A similar meeting was held in case CrossTown, where all team members from both sites would gather on one site to discuss new milestones that were set with the customer, i.e. to give everybody a common understanding of what lies ahead, spanning multiple sprints (CT.P.39).
Linked Guidelines: G1, G3
Mitigates Challenges: Coordination
Rationale: An all-site broadcasting meeting can be an efficient way to transfer knowledge (G3) to all sites (G1) because inter-site coordination is easier when all sites share the same level of information.

Practice P27: User Story Requirements

Feature requirements should be phrased as user stories to communicate the user's role, goal and benefit clearly across sites.

Further elaboration from empirical evidence: This practice deals with backlog handling in DSD. Both cases CrossTown and NoTimeshift worked with a backlog containing user stories, well-known from collocated scrum. The idea is that feature requirements are customer-centric and as such add value to the product by being phrased in the following way: "As a <user role>, I want <to achieve some goal> so that <some reason/benefit>". The phrasing may differ, but the important thing is to state the user's role, the goal/function and the reason or benefit in a one-liner (CT.P.37, NT.P.33).
Linked Guidelines: G3, G6
Mitigates Challenges: Coordination, Control
Rationale: Formulating requirements as user stories controls that the development is in fact user-oriented with requirements that are understood by both the customer (G6) and the developers. User stories with the right granularity can be effectively used to transfer domain-specific knowledge across sites (G3) and simplify coordination with a well-defined backlog.

Practice P28: Code Quality/Standards

Cross-site code standards are important to improve code overall quality.

Further elaboration from empirical evidence: This practice deals with metrics in DSD. Case NoTimeshift had massive problems with code quality in their two-site environment including broken builds, which becomes even more frustrating if the teams are not collocated. The BDD workflow helped to improve code quality and also the establishment of general cross-team working agreements, both on the code level (code formatting, comments, ...) and on a social level (not committing untested revisions, letting others know of major refactorings that may raise problems for others, ...) (NT.P.34). Case Continental worked with very strict coding standards that e.g. also involved allowed third-party libraries (C.P.18).
Linked Guidelines: G3, G7
Mitigates Challenges: Coordination, Control
Rationale: Code standards can be used as one (among others) of the controls for software quality (G7) by sharing common standards among all sites (G3). Coordination is facilitated (and discussions are limited) when coding adheres to an agreed set of standards.

Practice P29: Agile Coach

An experienced agile coach can accompany the agile DSD process implementation.

Further elaboration from empirical evidence: This practice addresses a new role in agile DSD. An agile coach is an experienced agile practitioner and consultant who can help in setting up the initial process implementation (NT.P.32) or accompany the whole project (CT.P.36). The agile coach may be able to avoid some common pitfalls and thus arrive at a stable process implementation and good agile culture faster.
Linked Guidelines: G10
Mitigates Challenges: Communication
Rationale: An agile coach can oversee the agile process implementation and its continuous improvement (G10) and should be in touch with all sites.

6.2.2 Conceptual Practices

Conceptual Practice C1: Documentation Strictly in Common Language

All documentation should be held in a language understood by all parties.

Further elaboration from empirical evidence: This conceptual practice addresses proximity in DSD and is considered conceptual because it was only observed in one case (Continental). To allow everybody access to documentation, all written communication should be conducted in a language understood by all sites (C.P.14).
Linked Guidelines: G1, G3
Mitigates Challenges: Coordination
Rationale: Documentation needs to be in a language understood by everybody for an equal participation of all sites (G1) and efficient knowledge transfer (G3), otherwise coordination is severely handicapped.

Conceptual Practice C2: Document Informal Communication

All relevant informal communication must be documented and made available to all sites.

Further elaboration from empirical evidence: This practice addresses the lack of interaction in DSD and is considered conceptual because it was only observed in one case (CrossTown). Relevant informal communication is regarded as everything that adds to clarify issues, be it technical, process or of other nature. If a ticket management system is in place, it is good practice to add informal communication (chat, instant messages, emails, ...) to the ticket

description or post it to a wiki (CT.P.11).
Linked Guidelines: G3
Mitigates Challenges: Coordination
Rationale: Informal communication needs to be documented, especially when it involves decisions relevant to other sites (G3), otherwise it is a lot harder for teams to self-organize and coordinate.

Conceptual Practice C3: Hours of Availability

Publish individual hours of availability to all sites.

Further elaboration from empirical evidence: This practice addresses the lack of interaction in DSD and is considered conceptual because it was only observed in one case (CrossTown). This practice is two-fold (CT.P.16): First of all, all team members should be available and involved in informal communication, i.e. be online for chat and instant messaging, read emails regularly and also make known, when somebody is absent (e.g. a common wiki page for long term absences and status message in the messenger for short term absences). Absence includes both being physically not available, i.e. not in the office, or currently focusing on an issue and not available to talk or chat.
Linked Guidelines: G10
Mitigates Challenges: Communication
Rationale: Hours of availability need to be published to allow informal communication, which is necessity for every agile process implementation (G10).

Conceptual Practice C4: Offer Language Courses

Offering language course can improve building social relationships.

Further elaboration from empirical evidence: This practice addresses the lack of interaction in DSD and is considered conceptual because it was only observed in one case (Continental). For long-term distributed development efforts, offering language courses to interested team members can increase motivation and facilitate the establishment of trust and social relationships across sites (C.P.5).
Linked Guidelines: G1, G3
Mitigates Challenges: Coordination
Rationale: Language courses should be offered to overcome language barriers and share knowledge and information (G3) for intra- and inter-team coordination. It also helps to create an equal involvement of all sites (G1).

Conceptual Practice C5: Plan Time for Research and Learning

Research and learning needs to be included in estimations.

Further elaboration from empirical evidence: This practice deals with the sprint planning in DSD and is considered conceptual because it was only observed in one case (NoTimeshift). Implementing features may require searching for new technical solutions (research). New team members need time to get to know the environment and code base (learning). Research and learning are both worthwhile and necessary and this time needs to be accounted for during sprint planning (NT.P.26).
Linked Guidelines: G3, G7
Mitigates Challenges: Coordination, Control
Rationale: Research and learning is part of the development process when working with new technologies or new code or requirement areas. Sharing the information and knowledge takes time (G3) but is important to produce quality output (G7). Planning these efforts allows better control and provides more solid and reliable estimations, which facilitates many aspects of coordination.

Conceptual Practice C6: Lessons Learned Workshop after Project

Conducting a lessons learned workshop allows reflection across the whole time span and encourages improvements for new projects.

Further elaboration from empirical evidence: This practice deals with the retrospective in DSD and is considered conceptual because it was only observed in one case (Continental). While the retrospective focuses on the last sprint iteration, a lessons learned workshop after the project ends (or near the end) can be used to reflect on the whole project's time span (C.P.10) and can be used to derive new practices for the next project.
Linked Guidelines: G4
Mitigates Challenges: Control
Rationale: A lessons learned workshop can be regarded as a retrospective covering the full project's lifecycle (G4), which adds to controlling the project end and provides feedback for future implementations.

Conceptual Practice C7: Selective Pair Programming

Selective pair programming lets developers decide when pair programming is beneficial.

Further elaboration from empirical evidence: This practice deals with interaction in DSD and is considered conceptual because it was only observed in one case (CrossTown). In case CrossTown pair programming was used very selectively, only when developers decided they needed an extra pair of eyes, e.g. for critical algorithms or delicate refactorings (CT.P.40).

Linked Guidelines: G7, G9
Mitigates Challenges: Control, Communication
Rationale: Pair Programming provides an extra feedback cycle for communication and control purposes (G9) and helps to improve code quality (G7).

6.3 Guidelines

The overspanning guidelines were extracted similarly to the practices (cf. Figure 5.1): Each individual case resulted in an extracted set of guidelines (cf. Figures 5.7, 5.9 and 5.11). From an ADAPT framework's point of view these guidelines are only regarded as conceptual guidelines. This section now compares the three sets of guidelines to find common (full) ones.

The following steps were employed to arrive at the final set of guidelines (cf. Section 5.1.6).

1. Arrange guideline concepts in groups with similar properties (visualized in Figures 5.7, 5.9 and 5.11).
2. Add all guidelines with the identified similar concepts to a spreadsheet, assign each guideline an ID for better future reference (cf. Table 6.2).
3. Merge the identified duplicate concepts and concepts with similar properties to one guideline (in terms of open coding: a category) with empirical support in at least two cases. Concepts without empirical support in more than one case are discarded.
4. Link all guidelines to one or more of the three DSD challenge types coordination, control and communication such that linked guidelines mitigate the challenge.

Table 6.2 shows step 2, where all guidelines are added to a spreadsheet. The groups to intermediary classify concepts were: retrospective, board/workflow, development sites, teams, customer, sprint iterations, tools, information sharing, interaction and general. They are only used to identify possible concepts for merging as a temporary help and have no further semantic meaning to the ADAPT framework.

ID	Merge Group	Guideline Concept Title	Source
CT.G.7	Retrospective	Use retrospective as a mood barometer across all sites (keep frustration levels low)	CrossTown
CT.G.2	Retrospective	Retro can be a powerful driver for continuous process improvement	CrossTown
CT.G.9	Board/Workflow	Definition of done needs to be well-defined and agreed (even more important in DSD)	CrossTown
NT.G.15	Board/Workflow	Visualize the workflow truthfully	NoTimeshift
NT.G.1	Development Sites	Foster equal involvement of all sites	NoTimeshift
C.G.8	Development Sites	Understand each other's processes and try to align them (multiple suppliers)	Continental

CT.G.14	Development Sites	Equal involvement of all sites	CrossTown
C.G.11	Development Sites	Equally distribute decision makers among the sites	Continental
C.G.10	Development Sites	Define roles clearly and enact them	Continental
C.G.2	Development Sites	Enforce contact interfaces and keep role fluctuation to a necessary minimum	Continental
NT.G.12	Teams	Distributed teams over dispersed teams (virtual inter-team coordination but on-site intra-team coordination)	NoTimeshift
NT.G.5	Teams	Build trust	NoTimeshift
C.G.12	Teams	Build team-spirit	Continental
NT.G.22	Teams	Aim for fully distributed cross-functional teams working on requirement areas	NoTimeshift
NT.G.8	Teams	Hire motivated people	NoTimeshift
NT.G.7	Teams	Establish willingness and commitment to change	NoTimeshift
NT.G.9	Teams	Establish a good working atmosphere	NoTimeshift
NT.G.17	Teams	Establish long-lived teams	NoTimeshift
NT.G.18	Teams	Teams small enough to not encourage sub-teaming (teams within teams)	NoTimeshift
NT.G.20	Teams	Allow self-organization also in distributed environments	NoTimeshift
C.G.4	Customer	Involve customer in development progress (meetings)	Continental
C.G.5	Customer	Invite customer into the tool chain (dashboard, access to ticket management system)	Continental
CT.G.10	Customer	Frequent releases (as frequent as feasible/sensible)	CrossTown
CT.G.6	Customer	Regular deployments to the customer set a common goal and encourage frequent integration among teams and sites	CrossTown
NT.G.10	Sprint Iterations	Plan in time	NoTimeshift
NT.G.16	Sprint Iterations	Respect Iterations (Product Owner)	NoTimeshift
CT.G.12	Sprint Iterations	Short sprint intervals let problems between sites surface quickly	CrossTown
CT.G.5	Sprint Iterations	Quality over Feature Rush	CrossTown
CT.G.15	Sprint Iterations	Work towards a continuous flow (instead of panic bugfixing at sprint end)	CrossTown
CT.G.11	Sprint Iterations	Deployment-ready features at the end of sprint iteration instead of test-ready ones	CrossTown
CT.G.13	Tools	Establish a proper tool chain (preferably integrated)	CrossTown
NT.G.6	Tools	Tool choices are secondary, but all sites should commit to using the same tools. Keep tool chain small.	NoTimeshift
NT.G.19	Information Sharing	Facilitate knowledge sharing across sites	NoTimeshift
CT.G.3	Information Sharing	Make informal communication visible/transparent to everyone	CrossTown
CT.G.1	Information Sharing	Keep documentation up to date, especially one relevant to other teams (sites)	CrossTown
NT.G.14	Information Sharing	Documentation becomes more important	NoTimeshift
C.G.9	Interaction	Establish multi-level (multi-concern) feedback loops	Continental
NT.G.11	Interaction	Keep video conference meetings compact and focused and in manageable group sizes	NoTimeshift
NT.G.13	Interaction	Keep virtual meetings to minimum for the sake of on-site meetings	NoTimeshift
NT.G.2	Interaction	Allow informal communication as a substitute for face-to-face communication	NoTimeshift
NT.G.3	Interaction	Allow formal communication as a substitute for face-to-face communication	NoTimeshift
C.G.6	Interaction	All communication in common language	Continental
CT.G.4	Interaction	Teams need to synchronize using both formal and informal means of communication	CrossTown
C.G.1	General	Dealing with formalities slows down the project	Continental
C.G.3	General	Don't let years of experience get in the way of trying something new	Continental
CT.G.8	General	Process adaptation takes time (small steps instead of big leaps)	CrossTown
NT.G.4	General	Beware of a superficial adoption of agile values	NoTimeshift
C.G.7	General	Agile practices can also be employed in traditional environments (to a limited extent)	Continental

NT.G.21	General	Small steps instead of big leaps (continuous improvements over quick gains)	NoTimeshift

Table 6.2 – The table gives an overview of the cross-case guidelines extraction status after step 2 and shows 49 conceptual guidelines.

The guidelines are described in the following way: The description explains how the guideline was constructed using single-case guidelines extracted from the three cases CrossTown, NoTimeshift and Continental from Table 6.2 including the guideline IDs in parentheses. It concludes with *linked practices*, *mitigated challenge* and the *rationale* behind.

Guideline G1: Strive for an equal involvement of all sites with clearly defined roles and allow people enough time to concentrate on and fully enact their role.

All three cases showed that the sites need to be able to work at an equal level, i.e. be as equal as organizational constraints allow. The situation improved in all three cases when the sites moved together and improved collaboration. It is therefore necessary to distribute decision makers equally between sites, i.e. have product owners (or their proxies) on all sites. Moreover, the roles (such as the product owner and scrum master) need to be allowed enough time to fully enact their roles and be available to their on-site colleagues as well as to the other sites, this roles always require full-time effort in DSD environments. It is also important to build an understanding of each other's organizational processes and constraints in case several suppliers are involved in the DSD environment. This process of fighting inequalities may take time but is worthwhile to pursue (CT.G.14, NT.G.1, C.G.8, C.G.10, C.G.11).

Linked Practices: P1, P3, P5, P10, P11, P16, P17, P24, P26, C1, C4
Mitigates Challenge: Coordination
Rationale: Following the guideline G1 allows the mitigation of coordination challenges with development partners on different sites collaborating at eye level. Equal involvement can be facilitated with scrum masters on each site (P5) that can look after the process implementation to prevent local process optimizations interfering with the global (all-site) process. Travelling ambassador (P1) and/or team rotations (P3) help to better include distant sites better into everyone's thinking. Synchronized sprints (P10) severely reduce planning overhead. The following practices try to achieve better flow of information in various aspects to facilitate coordination: separation of roadmap and sprint planning (P17), accessible backlogs (P11), multi-level on-site proxy planning (P16), on-demand specification meetings (P24) and global all-site broadcast meetings (P26). Last but not least, having documentation strictly in a common language (C1) and offering language courses (C4) is a prerequisite for effective inter-site (inter-team) coordination.

Guideline G2: Create an environment that allows compact long-lived cross-functional teams to self-organize. Work towards distributed teams in the long run.

Long-lived teams with little fluctuation are generally regarded as more productive (Larman and Vodde, 2009). In DSD this needs to be weighed against having cross-site team rotations for getting to know each other better. So a rule of thumb can be defined as having little on-site fluctuation but allowing cross-site team rotations when feasible. No matter if teams are dispersed (team members spanning multiple sites) or distributed (all team members collocated on one site) it is advised to keep team sizes small to ease inter-team coordination, but teams should always be cross-functional, i.e. able to implement a feature on their own (or with as little dependency as possible) in order to not encourage sub-teaming (teams within teams). The cases CrossTown and NoTimeshift also showed that it helps to organize teams around requirement areas such that teams can familiarize themselves with the area. This procedure is beneficial, but can be discarded if the project environment does not allow such area groupings. In general it is easier and more productive to work with distributed teams instead of dispersed ones because then only inter-team coordination is necessary, while intra-team coordination can be done on site. The scrum masters must critically review at all times that teams are allowed to be self-organizing and handle their communication and coordination by themselves otherwise the process is not agile and he must work towards building trust and establishing a good working atmosphere (NT.G.5, NT.G.9, NT.G.12, NT.G.17, NT.G.18, NT.G.20, NT.G.22, C.G.2, C.G.12).

Linked Practices: P4, P7, P14, P21, P22, P25
Mitigates Challenge: Coordination
Rationale: The guideline G2 strives for self-organizing inter and intra-team coordination. Team events (P4) allow the team members to get to know each other in person and establish personal relationships, which makes it easier to make use of multi-way informal communication (P7) to coordinate. Feature-team organization (P14), daily intra-team communication (P21), inter-team scrum of scrums (P22) and the cross-team specialist meeting (P25) are all practices that help teams to succeed in their self-organization.

Guideline G3: Transfer knowledge and share information between sites to establish the necessary working flow of information.

Knowledge transfer and information sharing build viable links between sites in agile DSD and it is something that everybody has to work for. One important part is to make informal communication visible to everyone e.g. by appending information to a ticket in a ticket management systems and thus share clarifications and decisions to others interested. Moreover documenta-

tion plays a bigger part with distributed sites and more effort needs to be put into writing documentation (meaning all kinds of documentation, from full-fledged documents to comments in source code). Other possibilities for sharing information are already part of the scrum process with all meetings from planning to retrospective (CT.G.1, CT.G.3, NT.G.14, NT.G.19).

Linked Practices: P1, P2, P3, P7, P8, P12, P13, P17, P26, P27, P28, C1, C2, C4, C5
Mitigates Challenge: Coordination
Rationale: Knowledge transfer and information sharing is very important to allow inter-site coordination. Knowledge sharing is important both with regular on-site visits using travelling ambassador (P1), full team on-site sessions (P2), team rotations (P3) and/or customer requirements workshop (P13) but also distributed using multi-way informal communication (P7), meeting minutes (P8), global all-site broadcast meeting (P26), code quality/standards (P28) and documenting informal communication (C2). Practices tangible requirements (BDD) (P12) and user story requirements (P27) support the creation of comprehensible requirements, which are easier to share. Documentation needs to be strictly written in a common language (C1) and language courses (C4) can substantially improve knowledge transfer. Planning time for research and learning (C5) is important in agile processes implementations and holding roadmap and sprint plannings (P17) helps to provide a common picture on the future steps.

Guideline G4: Use retrospective as mood barometer and driver for continuous process improvement.

Case CrossTown and NoTimeshift both showed that the retrospective was the most important driver for process improvement and also serves as mood barometer and a means to keep frustration levels low as everybody can voice his opinion and suggest improvements to the process. The scrum master needs to take an active part in making the proposed changes happen and in tracking their progress throughout the sprint to establish and manifest the willingness and commitment to change and improve the agile process implementation, which is very important in DSD (CT.G.2, CT.G.7, NT.G.7).

Linked Practices: P8, P18, P20, C6
Mitigates Challenge: Control
Rationale: The retrospective can be regarded as a tool of control, both for management and for the process participants. When possible, an on-site retrospective with proxies (P18) provided good results during the case studies. The agreed measures during retrospective need to be tracked (P8) and evaluated (P20) in the next sprint. The lessons learned workshop at the

end of the project (C6) is a retrospective addressing the full project's timespan for reflecting and creating improvements for future projects.

Guideline G5: Visualize the workflow including all sites truthfully. Leave no steps out and clearly define the end of a step.

The multiple-case study showed that visualizing the workflow becomes especially important in DSD as it always involves a complex setup spanning multiple teams and development sites. It is therefore absolutely necessary to represent the workflow as it is now (and not as it should be), even if it is a subpar process in the beginning that involves a lot of manual steps. Otherwise bottlenecks in the workflow cannot be identified and improvements cannot be made. An important part of the workflow is to agree on common definitions of done (DoD). This includes both the definition (or several) of when a feature is really done, because *done* could mean e.g. any of the following: "ready for test", "ready for user acceptance test", "ready for production", as well as the definition of when a step in the workflow is completed. The workflow represents the process as a whole and as such is a living artifact in an agile process and should be under review each retrospective to find improvements (CT.G.9, NT.G.15).

Linked Practices: P5, P6, P10, P12, P15, P20
Mitigates Challenge: Control
Rationale: A truthful representation of the workflow can be used to control the process implementation and set initiatives for improvement. The scrum master on each site (P5) can help control that each step is visualized and performed correctly. The workflow is easier to visualize when using synchronized sprints (P10) and all teams should follow the same workflow in one board (P15). The workflow can be accompanied by metrics (P20) for evaluation. BDD-style requirements with acceptance criteria (P12) and cross-site reviews (P6) can be effective steps in the workflow to improve quality and control.

Guideline G6: Invite the customer in and establish a customer-centric process with frequent releases to the customer.

Case Continental showed that it is very important to include the customer regularly into the process otherwise the end product may not be what the customer actually wanted. Integrating the customer into the process is as important in DSD as it is in collocated development. Since not all sites may be able to meet with the customer, other ways have to be established to invite the customer into the process such as providing access to the workflow board, the ticket management system or a report dashboard. Deploying regularly (best case: each sprint) to the customer ensures that the software is moving

in the right direction and set a common goal for all sites to integrate frequently and produce working software (CT.G.6, CT.G.10, C.G.4, C.G.5).

Linked Practices: P12, P13, P23, P27
Mitigates Challenge: Control
Rationale: The customer needs to be satisfied with the product increment. A customer-oriented workflow demands adequate quality for delivery and serves as a control to the process implementation. Good means to invite the customer into the process are using tangible requirements with acceptance criteria (P12) formulated as user stories (P27), customer requirements workshop (P13) and frequent deployments to customer (P23).

Guideline G7: Embrace quality over feature rush: Respect iterations across multiple sites and work towards a continuous flow.

Since several sites collaborate to develop software, frequent integration is necessary and testing/quality assurance becomes very important. Quality must be established within the sprint iteration. To that end iterations must be respected by the product owner and other stakeholders and planning must be held timely at the beginning of the sprint to not have a late start. Short sprint intervals can help in letting problems between sites surface quickly. The flow should be continuous with early integration testing (as compared to last minute bugfixing at the end of the sprint) to have deployment-ready features at the end of the sprint instead of test-ready ones ("feature rush") (CT.G.5, CT.G.11, CT.G.12, CT.G.15, NT.G.10, NT.G.16).

Linked Practices: P6, P19, P23, P24, P28, C5, C7
Mitigates Challenge: Control
Rationale: This guideline addresses the need for not over-managing an agile process, which is a control challenge. The teams need to be allowed to work peacefully within iterations to build quality features and accomplish their commitment. Quality controls are very important in distributed development and can be achieved with the following practices: cross-site reviews (P6), on-site sprint review with proxies (P19), frequent deployments to customer (P23) demanding good quality at the end of each sprint, on-demand specification meetings (P24) allowing to clear doubts timely within the sprint iteration, code quality/standards (P28), selective pair programming (C7) for critical parts and allowing time for research and learning (C5) to build it right the first time.

Guideline G8: Build your tool chain to support agile practices as the process shall never be a slave to the given tool chain.

While tools are important for any process implementation the process shall never be limited by the given tool chain. If not the case, the tool chain must be improved towards and integrated one for all sites which supports the agile process (e.g. provide a workflow board). If not feasible or the tool integration takes too much time, it is possible to start on paper and stream the content with webcams to the other site. This is a cost-effective approach to begin with and can later be replaced by an electronic tool support (but does not have to). It is furthermore important that all sites use the same tools (even if multiple suppliers with their own separate tool landscape are collaborating). Having as few separate tools as necessary to support the process is the preferred approach (CT.G.13, NT.G.6).

Linked Practices: P7, P9, P11, P15, P20, P23
Mitigates Challenge: Communication
Rationale: The tool chain should increase, not limit, available communication channels. The tool chain should support the following practices: multi-way informal communication (P7), ad-hoc screensharing (P9), accessible backlogs (P11), multi-team workflow board (P15), metrics to evaluate the process (P20) and frequent deployments to the customer (P23).

Guideline G9: Establish multi-level (multi-concern) feedback loops as substitutes for face-to-face communication, including both formal and informal ways of communication.

All three cases showed that multi-level feedback loops between sites need to be established to serve different purposes from formal to informal communication. However virtual meetings are harder to moderate and should be kept compact and limited to a manageable size of participants. On-site meetings are preferable to virtual ones but this is often not feasible in DSD. The formal and informal channels of communication can substitute the missing face-to-face communication in DSD. All sites should use the same language for communication (CT.G.4, NT.G.2, NT.G.3, NT.G.11, NT.G.13, C.G.6, C.G.9).

Linked Practices: P6, P9, P18, P19, P21, P22, P25, C7
Mitigates Challenge: Communication
Rationale: Multi-level feedback loops need to be established to allow a varying richness in communication and information using some or all of the following practices: cross-site reviews (P6), ad-hoc screensharing (P9), on-site retrospective (P18) and sprint review (P19) with proxies, daily intra-team communication (P21), inter-team scrum of scrums (P22), cross-team specialist meeting (P25) and selective pair programming (C7).

Guideline G10: Create an environment where agile can work. Small steps over big leaps.

This guideline is general and overarching. Adapting the process implementation with the ADAPT framework takes time and small steps each sprint in the right direction. In DSD it is even harder because there are always multiple sites and also often multiple parties involved, which need to align their individual processes to a working collective. Be willing to try something new and do net let formalities and organizational constraints slow down the process improvement. Although agile practices can be applied in traditional environments as case Continental showed, beware of a superficial adoption of agile values. Making a distributed process work takes extra effort and motivated people, that also holds true for agile DSD, but it adds very valuable tools and drives continuous process improvement (CT.G.8, NT.G.4, NT.G.8, NT.G.21, C.G.1, C.G.3, C.G.7).

Linked Practices: P5, P7, P29, C3
Mitigates Challenge: Communication
Rationale: The overarching guideline G10 addresses communication as communication is regarded as a mediating factor affecting both coordination and control in the CCC model (cf. Figure 3.2). Both an agile coach (P29) and scrum masters on each site (P5) are important to oversee that agile values are implemented truthfully and not have a superfluous agile implementation. A very central practice for communication (and thus also for coordination and control) is establishing multi-way informal communication (P7) to allow teams to self-organize, best used in combination with having clear hours of availability (C3) of all team members.

6.4 ADAPT Framework v1.0

After the presentation of practices in Section 6.2 and guidelines in Section 6.3 this section puts the pieces together to the first iteration of the ADAPT learning framework and is as such the emerging theory and output artifact of this thesis. Table 6.3 illustrates the whole framework including the hierarchical setting of challenges categories, guidelines and practices and has been created using the following step from Section 5.1.6:

Link all guidelines to one or more of the three DSD challenge types coordination, control and communication such that linked guidelines mitigate the challenge.

Table 6.3 is the main output artifact of this thesis, as it provides a compact yet overarching view of the full ADAPT framework in its first full iteration with the hierarchy of challenge categories (CCC), guidelines and practices. The

practitioner can use it as a one-page roadmap to steer the process adaptation and instantiate a subset of the practices provided.

G1: Strive for an equal involvement of all sites with clearly defined roles and allow people enough time to concentrate on and fully enact their role.
G2: Create an environment that allows compact long-lived cross-functional teams to self-organize. Work towards distributed teams in the long run.
G3: Transfer knowledge and share information between sites to establish the necessary working flow of information.
G4: Use retrospective as mood barometer and driver for continuous process improvement.
G5: Visualize the workflow including all sites truthfully. Leave no steps out and clearly define the end of a step.
G6: Invite the customer in and establish a customer-centric process with frequent releases to the customer.
G7: Embrace quality over feature rush: Respect iterations across multiple sites and work towards a continuous flow.
G8: Build your tool chain to support agile practices as the process shall never be a slave to the given tool chain.
G9: Establish multi-level (multi-concern) feedback loops as substitutes for face-to-face communication, including both formal and informal ways of communication.
G10: Create an environment where agile can work. Small steps over big leaps.

	Coordination			**Control**				**Communication**		
Guidelines / **Practices**	**G1**	**G2**	**G3**	**G4**	**G5**	**G6**	**G7**	**G8**	**G9**	**G10**
P1: Travelling Ambassador	✓		✓							
P2: Full Team On-Site Sessions			✓							
P3: Team Rotations	✓		✓							
P4: Team Events		✓								
P5: Scrum Master on each Site	✓				✓					✓
P6: Cross-Site Reviews					✓		✓		✓	
P7: Multi-Way Informal Communication		✓	✓					✓		✓
P8: Meeting minutes			✓	✓						
P9: Ad-Hoc Screensharing								✓	✓	
P10: Synchronized Sprints	✓				✓					
P11: Accessible Backlogs	✓							✓		
P12: Tangible Requirements (BDD)			✓		✓	✓				
P13: Customer Requirements Workshop			✓			✓				
P14: Feature-Team Organization		✓								
P15: Multi-Team Workflow Board					✓			✓		
P16: Multi-Level On-site Proxy-Planning	✓									
P17: Separation of Roadmap and Sprint Planning	✓		✓							
P18: On-Site Retrospective with Proxies				✓					✓	
P19: On-Site Sprint Review with Proxies							✓		✓	
P20: Establish Metrics to Evaluate the Process				✓	✓			✓		
P21: Daily Intra-Team Communication		✓							✓	
P22: Inter-Team Scrum of Scrums		✓							✓	
P23: Frequent Deployments to Customer						✓	✓	✓		
P24: On-Demand Specification Meetings	✓						✓			
P25: Cross-Team Specialist Meeting		✓							✓	
P26: Global All-Site Broadcast Meeting	✓		✓							
P27: User Story Requirements			✓			✓				
P28: Code Quality/Standards			✓				✓			
P29: Agile Coach										✓
C1: Documentation Strictly in Common Language	✓		✓							
C2: Document Informal Communication			✓							
C3: Hours of Availability										✓
C4: Offer Language Courses	✓		✓							
C5: Plan Time for Research and Learning			✓				✓			
C6: Lessons Learned Workshop after Project				✓						
C7: Selective Pair Programming							✓		✓	

Table 6.3 – Compact overview of the ADAPT framework v1.0 including the full hierarchy of challenge categories (CCC), guidelines and practices.

6.5 Conclusion

The cross-case analysis in this chapter aimed at answering the research questions that guided the multiple-case study RQ4a and RQ4b.

RQ4a. *Which process design guidelines and practices can increase the chances of a successful agile process implementation in distributed environments?*

The three cases CrossTown, NoTimeshift and Continental showed that agile practices can be successfully applied to all three distribution scenarios defined by Prikladnicki et al. (2003). All three cases have been reported to the full extent of the checklist (cf. Appendix A.3) designed as an outcome of this thesis' systematic mapping study. The multiple-case study featured the following cases:

- Case CrossTown: sites distributed within one city, spanning two districts, yielding 15 conceptual guidelines and 40 conceptual practices
- Case NoTimeshift: sites distributed within one country, spanning two cities, yielding 22 conceptual guidelines and 35 conceptual practices
- Case Continental: sites distributed within one continent, spanning three countries, yielding 12 conceptual guidelines and 19 conceptual practices

So the individual single-case analysis resulted in total in 49 guideline and 94 practice *concepts*. These concepts were then analyzed, merged and combined cross-case and led to a final set of 10 guidelines, 29 full practices (with evidence in at least two out of the three cases) and 7 conceptual practices (that lack evidence in more than one case). The practices were then linked to guidelines and the guidelines in turn linked to the three challenge categories coordination, control and communication, to build the first full iteration of the ADAPT learning framework as the outcome of this thesis (cf. Table 6.3). The guidelines provide high level information and can be implemented using the concrete linked practices.

RQ4b. *Do the different distribution scenarios affect the implementation of agile practices?*

The thesis showed that 29 practices emerged from varying distribution scenarios to generate common practices, hence there is no evidence that the scenario has effects on the implementation of the agile practices. However, the global scenario has been outscoped for the ADAPT framework as it was not featured in the multiple-case study, but it can be added in future iterations. With the given empirical base of three case studies results show that the practices are not covering each problem to the same extent, e.g. while there are P1-P4

dealing with missing proximity in DSD, there is only one practice to address the sprint iteration (P10). Future iterations should try to grow larger in aspects that the current three cases could not shed more light on as of now. Further discussion of this research question will be provided based on expert interviews in Chapter 7.

CHAPTER 7

Evaluation and Discussion of Results

Contents

This chapter discusses results and reports feedback on the ADAPT framework from several points of view. Section 7.1 presents feedback from two focus groups on the research methodology and framework design, one held at the XP 2014 conference in Rome and one at the Center for Design Research (CDR) at Stanford University. Further evaluation is conducted in Section 7.2, where 10 experts have been interviewed about the thesis, the first half on the framework design and the other half on the results (ADAPT v1.0). A summarizing look at related work in the field up to 12/2015 is presented in Section 7.3 to position the framework in the research field, now that it is completed. The propositions that have been created in the design theory (cf. Chapter 3) are now revisited in Section 7.4 to analyze whether the ADAPT v1.0 has followed the design truthfully and successfully fulfilled all five propositions TP1-TP5. Section 7.5 gives a compact discussion and final answer to the research questions RQ1-RQ4 that led through the different parts of the thesis. Section 7.6 describes the limitations and Section 7.7 presents various possibilities for

further enhancements in future work. In contrast to the preceding chapters, this chapter does not feature a conclusion section as the overall conclusion is covered in Chapter 8.

7.1 Focus Groups

Focus groups (Morgan, 1997) are group interviews, which a moderator guides and the empirical data is what the participants say during the focus group. The participants should typically come from a similar background. This thesis applied two focus groups as a means of evaluation. The first one was held during the XP 2014 conference and featured researchers and PhD students in the field of agile software development. The second focus group was held with the designX lab group at the Center for Design Research of Stanford University during the research leave of the author and thus featured a group of design research experts. Both focus groups were kicked off with a compact presentation by the author of about 20 minutes, which then evolved into a group discussion/interview, moderated by the author. The focus groups proved to be exceptionally useful to reflect and evaluate the design theory behind the framework (focus group at CDR, Stanford University, cf. Section 7.1.2) and the overall interest in the framework for the research field of agile software development (focus group at XP 2014, cf. Section 7.1.1).

7.1.1 XP 2014 Conference, Rome

The author attended the PhD symposium at the 15^{th} *International Conference on Agile Software Development (XP 2014)* in Rome, Italy and presented the status at the time with a special focus on the methodology, followed by a group discussion with 10 participants (not including the author and not counting sit-ins, who did not actively participate), which overall took about 40 minutes. The organizers were Dr. Davide Falessi (Fraunhofer Institute for Experimental Software Engineering, USA) and Dr. Xiaofeng Wang (Free University of Bozen-Bolzano). The invited mentor was Prof. Dr. Pekka Abrahamsson (Norwegian University of Science and Technology). Additionally attending and providing feedback was Prof. Dr. Giovanni Cantone (University of Rome "Tor Vergata").

Overall the feedback was very helpful and valuable to the creation of this thesis. Feedback received involved the sharpening of research questions to better reflect the planned outcome of the dissertation. Discussion showed that the process toolkit should be regarded as a first increment of such an agile distributed design, not a fully matured one, which aligns with the intention of the author and also the nascent nature of the research field. Furthermore, it was advised to explicitly state the given situational context as this approach may help to ensure a rigorous research output as well as facilitate future

research to build upon it. Table 7.1 summarizes the feedback and shows how it was addressed in the thesis.

Feedback	Implementation
Sharpen research questions as they cover too many different aspects	Cut down to the final 4 RQs (cf. Chapter 1)
ADAPT framework should be regarded as a first step in an ongoing process	Added more focus on it being a learning framework (cf. Chapter 1)
Empirical context is very important for future research	Designed the checklist for reporting context (cf. Chapter 4)
How to measure the effectiveness of the ADAPT framework	Test propositions TP1-TP5 (cf. Chapter 3)
Evolve into a pattern language	Needs more empirical data, future work (cf. Section 7.7)
Systematic literature review is essential to the nature of this thesis	Conducted a systematic mapping study of the last 15 years (cf. Chapter 4)

Table 7.1 – Summary of the feedback received at XP 2014 conference and details of its implementation.

7.1.2 Center for Design Research, Stanford University

The author held a focus group at the designX lab, Center for Design Research, Stanford University, which lasted 90 minutes. This section describes the moderated group interview. The 14 participants included Prof. Dr. Larry Leifer as well as several of his research staff and PhD students.

Overall the research endeavor of creating the ADAPT framework was regarded as very worthwhile, addressing a current and still unresolved issue, but it was also noted that it is a very big challenge and that it will require future iterations after the finalization of the thesis. The name ADAPT was perceived very positive and fitting well. It was stressed that design requirements need to feature a rationale and means of verification (testable proposition). It was clarified that the framework is not designed for a special kind of project but can and should be adapted to any context using the guidelines and practices at disposal. It was also discussed how implementing the ADAPT framework would be a topic of its own which would require different methods for validation. This is out of scope for the thesis, which focuses on creating a first full iteration of the framework. The need for a pattern language was also addressed, which is a possible evolutionary step for the framework after more empirical data is gathered. Furthermore, it was discussed whether the focus was research or practice and explained that this is a research-based framework which is built on empirical data and thus should also provide value to the practitioner. Table 7.2 summarizes the feedback and shows how it was addressed in the thesis.

Feedback	Implementation
Grand challenge requiring future iterations as well	Added to Design Theory (cf. Chapter 3)
Design requirements positively designed including rationale and metric	Added to Design Theory (cf. Chapter 3)
How to deal with different types of projects	The framework can deal with all types of projects (cf. Chapter 3)
Implementing the framework	Out-scoped for the thesis (cf. Section 7.7)
Pattern language	Needs more empirical data, future work (cf. Section 7.7)
Research or practice focus	Cf. Design Theory in Chapter 3: it is a research-based framework that also has value to the practitioner

Table 7.2 – Summary of the feedback received from the designX lab at Stanford University and details of its implementation.

7.2 Expert Interviews

Overall 10 semi-structured interviews have been conducted with experts in either research or practice to evaluate this thesis. The interviews lasted from about 40 to 90 minutes, were recorded, transcribed and coded. The interview guide can be found in Appendix A.4. Table 7.3 provides a short overview of all interviewed experts. As the table shows, the emphasis of the interviews was two-fold, interviews #1 to #5 concentrated on the framework design at the design stage, as the following exemplary quote shows, where a senior researcher advises to create a tangible framework, which eventually became part of the testable proposition TP4:

> "A sense of my thinking: How could we get this framework on the whiteboard, how could we start interacting with it, how can we get beyond words only?" (Sr. Researcher, Interview #1)

Interviews #6 to #10, on the other hand, dealt with the actual outcome of this thesis, the guidelines and practices of the ADAPT framework and their value to research and practice, including a presentation of the the compact ADAPT overview table (cf. Table 6.3):

> "[The overview table] is neat and easy to use. In one page, you have everything you need. [...] This is all of the information, only what's necessary. To me this is optimal, as an engineer. [...] What I find intriguing is that you could take this as is and start using it right now." (Software Architect & PhD, Interview #6)

The second set of interviews covers a varying background: a software architect and agile practitioner for many years, also holding a PhD (interview #6), a

long-time agile coach and certified scrum master (interview #7), an experienced program manager for distributed projects, holding a PhD in the field and taking a look from a non-agile perspective (interview #8), an industrial agile researcher (interview #9, only interview done remote and not in person) and interview #10, a practitioner and book author, for covering the product owner perspective as well. The experts remain anonymous for privacy reasons. Table 7.4 shows an aggregated summary of the interviews, including both approval and suggestions for future improvement.

#	**Current role**	**Exp.**	**Expertise**	**Origin**	**Lang.**	**Time**
1	Sr. Researcher	50 years	Design Research	US	ENG	2x40'
2	Sr. Researcher	10 years	Design Research	FI	ENG	92'
3	Sr. Industrial Researcher	15 years	Design Research	FI	ENG	60'
4	Sr. Researcher	5 years	Design Research	DE	GER	90'
5	Sr. Researcher	10 years	Design Research	DE	GER	40'
6	Software Architect & PhD	5 years	Agile DSD	AT	GER	63'
7	Agile Coach	7 years	Agile DSD	AT	GER	67'
8	Program Manager & PhD	11 years	DSD	AT	GER	60'
9	Industrial Researcher	8 years	Agile DSD	DE	GER	69'
10	Product Owner	5 years	Agile DSD	AT	GER	56'

Table 7.3 – Overview of the interview partners in the semi-structured expert interviews during evaluation of results.

Type	Category	Code	Summarized Feedback	Addressed how	Interview Source
Approval	Methods	Action research	Direct observation is the best research method, lucky to have such case for this framework	No action required	#1
		Case study research	Case study fits the problem well, used much in design research	No action required	#1
		Design propositions	Good messages, especially: adaptive, no silver bullet solution, easily extensible, tangible and detailed	No action required	#1
		Research problem	Grand challenge that could be very rewarding	No action required	#1, #2
		Overall methodology	Very good structure with design theory, systematic mapping, multiple-case study and evaluation	No action required	#6, #7, #8, #9
		Evaluation	Focus groups and interviews are good means to achieve first evaluation	No action required	#4, #6, #7
	ADAPT	Learning framework	Modularity is well-fit for future iterations	No action required	#5
		Golden nugget	Practices were regarded as the central component with the guidelines providing good meta-information, ready-to-go for implementation study	No action required	#2, #5, #6, #7, #8, #9, #10
		Guidelines	Provide good meta-information, well-phrased using imperative	No action required	#2, #7, #10
		ADAPT overview table	Well-designed to have all necessary information on one page	No action required	#6, #7, #8, #9, #10
		Hierarchy	The hierarchy of challenges - guidelines - practices serves the framework well and is also tangible for the practitioner	No action required	#2, #3, #5, #6, #7, #8, #9, #10
		Scoping	Outscoping global scenario makes sense to cut the scope to a manageable size	No action required	#1, #2, #6, #7
		Global scenario	Many current practices make sense in a global scenario as well, good extensibility for that scenario	No action required	#8
Improvements	Future Work	Implementation study	Would be very interesting, but requires different methods, could easily fill the scope of another dissertation. Interviewees 7, 9 and 10 also showed interest in supporting the implementation.	Added to future work (cf. Section 7.7)	#1, #2, #3, #4, #5, #6, #7, #8, #9, #10
		Database	Could provide a great means to improve the empirical basis, but needs to be moderated	Added to future work (cf. Section 7.7)	#5, #6, #7, #8, #9
		Practice cards	Great idea, good starting point next to the overview table, worth testing in practice	Added to future work (cf. Section 7.7)	#6, #7, #8, #9, #10
		Distribution scenarios	Once empirical basis is larger, analyze whether practices fit better to one or the other distribution scenario	Added to future work (cf. Section 7.7)	#3, #6, #10
		Scaling	Needs to be well-considered, depends also on the featured approach (database or more studies)	Added to future work (cf. Section 7.7)	#6, #7, #8, #9
		Practice equivalency	Future work could also analyze which practices are interchangeable, offering different perspectives on the practices such as simplicity, cost, impact, etc.	Added to future work (cf. Section 7.7)	#2, #5, #6, #7, #8, #9, #10
	Methods	Design criteria (past version)	Too narrative, lack detail, need to be quantifiable	Added rationale and metric in Chapter 3	#1, #3, #4, #5
		Adding a practice	State criteria for adding a practice and establish a feedback loop	Added to Chapter 5	#5
		RQs	Too many aspects covered in RQ	Sharpened RQs in Chapter 1	#2

Table 7.4 – Summary of the feedback received from 10 expert interviews for evaluating the framework design (interviews #1 to #5) and the resulting ADAPT framework (interviews #6 to #10).

7.3 Related Work

The systematic mapping study in Chapter 4 covered related work from 1999 to 2014 by extending the study of Jalali and Wohlin (2012a) covering years 1999-2009 by another five years, 2010-2014, effectively covering the 15-year period of 1999 to 2014. The results of the mapping have already been extensively presented and discussed in Chapter 4. This section will now add to the discussion by comparing the ADAPT framework v1.0 (cf. Chapter 6) to the results of the mapping study conducted. The ADAPT framework limited its scope to scrum and applicable XP practices because the extensive systematic mapping showed that the vast majority of reported cases used either scrum (59.7%) or mixed XP@Scrum (17.7%). By including both types, scrum and XP@Scrum, into the ADAPT framework, it includes 77.4% of the analyzed cases and thus indicates to have high relevance for current real-world problems. Figure 4.10 provided an overview of reported successful practices in the last 15 years. The comparison with the ADAPT framework's practices v1.0 shows that the majority of practices can be found in other reported cases as well, thus improving confidence in the results of this thesis. Table 7.5 offers the complete comparison and shows a rough mapping of the more general practices as part of the mapping and the more concrete practices of the ADAPT framework. The overview shows that most practices of the ADAPT framework were also reported in other case studies, which shows that the ADAPT framework v1.0 has both support and relevance to related work. Both the systematic mapping study and the first iteration of the ADAPT learning framework are a major step towards aggregating know-how and thus moving to the next evolutionary step away from isolated case study results to a common basis (ADAPT v1.0) that other researchers can also build upon. It is also notable that ADAPT practices seem to also have strong resemblance to the global scenario, which was included in the systematic mapping, but out-scoped for the current iteration of the ADAPT framework.

The following practices from the systematic mapping were not identified during the multiple-case study and should be investigated further for consideration to future iterations of the ADAPT framework: collective code ownership, continuous integration, simple/incremental design, planning game, refactoring and estimation meeting.

The discussion continues to address relevant related publications, also covering the remaining period of 09/2014-12/2015, which has not been part of the systematic mapping study. Dumitriu et al. (2011) have published several *strategies* to cope with agile global software development. They are a mix between ADAPT's guidelines and practices (in terms of abstraction), explaining strategy and possible practices, in a limited fashion, in a few sentences. Some practices of the ADAPT framework can also be found there: P2: Full-

ADAPT Practices	Practices from Systematic Mapping of 1999-2014 (case count in parentheses)
P1: Travelling Ambassador	Ambassador (2)
P2: Full Team On-Site Sessions	Contact Visits (11)
P3: Team Rotations	-
P4: Team Events	-
P5: Scrum Master on each Site	Scrum Master (21)
P6: Cross-Site Reviews	Code Reviews (3)
P7: Multi-Way Informal Communication	Video/Audio Conference Meetings (27), Instant Messaging (10), Wiki (9), Chat (2)
P8: Meeting Minutes	Enough Documentation (5)
P9: Ad-Hoc Screensharing	Screen sharing (3)
P10: Synchronized Sprints	Sprint/Iterations (38.5)
P11: Accessible Backlogs	Backlogs (38)
P12: Tangible Requirements (BDD)	TDD (12), Automated Testing (5)
P13: Customer Requirements Workshop	Requirements Workshop (4)
P14: Feature-Team Organization	Self-organizing team (2)
P15: Multi-Team Workflow Board	Electronic/virtual task board (9)
P16: Multi-Level On-site Proxy-Planning	Sprint planning (35)
P17: Separation of Roadmap and Sprint Planning	Sprint planning (35)
P18: On-Site Retrospective with Proxies	Retrospective (32)
P19: On-Site Sprint Review with Proxies	Sprint review/demo (26)
P20: Establish Metrics to Evaluate the Process	Burndown Chart (4)
P21: Daily Intra-Team Communication	Standup Meeting (50.5)
P22: Inter-Team Scrum of Scrums	Scrum of Scrums (14)
P23: Frequent Deployments to Customer	-
P24: On-Demand Specification Meetings	Product Owner (24), Proxy Product Owner (4), Product Owner Teams (4)
P25: Cross-Team Specialist Meeting	-
P26: Global All-Site Broadcast Meeting	-
P27: User Story Requirements	User stories (20)
P28: Code Quality/Standards	Coding Standards (16)
P29: Agile Coach	Agile Coach (5)
C1: Documentation Strictly in Common Language	-
C2: Document Informal Communication	-
C3: Hours of Availability	-
C4: Offer Language Courses	-
C5: Plan Time for Research and Learning	-
C6: Lessons Learned Workshop after Project	-
C7: Selective Pair Programming	Pair Programming (18)

Table 7.5 – A comparison of ADAPT practices and practices reported in the systematic mapping study of Chapter 4.

Team On-Site Sessions, P3: Team Rotations, P24: On-Demand Specification Meetings and P29: Agile Coach. The scope is much more limited covering ten strategies. van Hillegersberg et al. (2011) offer an overview of 33 practices as one-liners, but most of them extracted from single case reports, as an introduction to their own case report. Šmite et al. (2010a) offer a small list of *practice advice*, that also resembles the scope of the ADAPT guidelines in terms of the level of abstraction. These examples, among others (cf. Chapter 4), show that there are many studies who strive to improve agile DSD, but what separates the ADAPT framework is that it is fully based on empirical evidence, demands full description of contextual factors and allows further extension/iterations, also by other researchers.

Using the same search terms and databases as in Chapter 4 (except for database INSPEC as it was not available via TU Wien library services at

the time of finalizing the thesis), the period of 09/2014 of 12/2015 yields 34 potentially relevant studies, indicating a great interest in the research field. More recent work in the field includes several systematic literature reviews such as (Rizvi et al., 2015; da Silva Estácio and Prikladnicki, 2015; Razavi and Ahmad, 2014), case studies such as (Sundararajan et al., 2014; Lehtinen et al., 2015), also more interest towards lean approaches in DSD (Viswanath, 2014; Tripathi et al., 2015) and towards knowledge sharing (Razzak and Ahmed, 2014; Sungkur and Ramasawmy, 2014; Razzak and Mite, 2015). However, to date (12/2015) no framework similar to the scope and extent of the ADAPT framework can be found, neither in the systematic mapping study of 1999 to 08/2014 (cf. Chapter 4) nor in the additional literature search for the time period of 09/2014 to 12/2015 within this section.

7.4 Propositions revisited

In Chapter 3 five propositions have been described which are now revisited after the completion of the first full iteration of the ADAPT learning framework, the focus groups and evaluation expert interviews. The proposition descriptions, rationale and means of verification are taken from Chapter 3. The discussion is added in this section. It shall be noted that future iterations of the framework need to also adhere to these five initial propositions, i.e. the propositions have to be revisited each time a new iteration of the ADAPT framework is desired.

TP1. *Each practice of the ADAPT framework is grounded in empirical evidence.*

Rationale: The ADAPT framework is not a silver bullet solution but it is a set of tools based on empirical evidence showing what worked in which context. **Verification:** The empirical context of all practices and guidelines must be fully specified according to the checklist defined in Chapter 4. **Discussion:** The systematic mapping study of related work in the field resulted in a checklist for reporting contextual information in case studies on agile DSD. This checklist was used to describe the context for all three cases in this study CrossTown, NoTimeshift and Continental. All practices (and guidelines) were derived from these three cases, where the context has been fully described according to the checklist design. Hence this proposition has been successfully fulfilled for the ADAPT framework v1.0.

TP2. *The ADAPT framework allows a simple, pragmatic and iterative process tailoring (rather than planned and strictly managed).*

Rationale: Process tailoring should be part of any agile implementation.

Verification: Evaluation of the process design (cf. Section 3.3) is conducted through two focus groups and ten expert interviews. **Discussion:** The feedback from expert interviews and focus groups has been overwhelmingly positive that the ADAPT framework in its current state (v1.0) is ready to be implemented and experimented with in a real study to achieve further improvements. Such study is planned in future work (cf. Section 7.7). This proposition is regarded as successfully fulfilled as the expert interviews indicate that the framework is heading in the right direction, but also taking into account that it needs further practical evaluation in future work.

TP3. *The ADAPT framework supports project-based process tailoring (rather than organization-based).*

Rationale: Even within the same organization each project is unique. **Verification:** The multiple-case study needs to feature different distribution scenarios (outscoped: global scenario) and thus different project-based environments. **Discussion:** The multiple-case study featured three different distribution scenarios: CrossTown, NoTimeshift and Continental. 10 guidelines and 29 practices were found that emerged from at least two of these distribution scenarios, which shows that the adapt framework provides value for different distribution types and thus also different project settings. The global scenario is currently out of scope but can be added to the framework once more empirical evidence is gathered in future studies. Hence this proposition has been successfully fulfilled for the ADAPT framework v1.0.

TP4. *The ADAPT framework provides tangible and detailed advice to the practitioner.*

Rationale: In order to be of practical use the practices must provide enough detail. **Verification:** Evaluation is conducted through two focus groups and ten expert interviews. **Discussion:** The interviews (especially the more industry-focused ones: #3, #6, #7, #8 and #10) showed that the current iteration (v1.0) of the ADAPT framework provides a good starting point to implement agile practices to a DSD environment. While it has been noted that in some parts the variety of practices is rather limited (e.g. sprint variations, only P10 available), the overall common evaluation was that it is a worthwhile endeavor, also for the practitioner. The current overview table (cf. Table 6.3) was found to have a very compact design that is very useful once you know what all the practices are about, for which the in-depth practice description has to be read first. There were also several interesting future approaches to make the framework even easier to handle in practice (cf. future work in Section 7.7 such as e.g. practice cards, a website/database or a pattern language). Following the positive feedback by experts, this proposition is regarded as successfully fulfilled for the current iteration of the ADAPT

framework, keeping in mind that future implementation studies can make the testing of this proposition more robust.

TP5. *The ADAPT framework is easily extensible.*

Rationale: In order for the ADAPT framework to further evolve and improve, practices and guidelines need to be extensible. **Verification:** Evaluation is conducted through two focus groups and ten expert interviews. At least one possible way of expanding the framework must be scheduled for future work. **Discussion:** The current iteration (v1.0) has been built using three cases, where guidelines and practices were first extracted using a grounded theory approach and then merged to a final set of practices and guidelines, which emerged from at least two of the three cases. While the two out of three approach is a good means to start building a base, it is not an approach worth following for future iterations of the ADAPT framework, as experts seemed to agree during the evaluation interviews. The experts acknowledged that in general the flat hierarchical structure of the ADAPT framework has good potential to be expanded, but that further scaling mechanisms need to be designed or, possibly, experimented with as the empirical basis grows larger. Possible scaling techniques discussed during the interviews (to be evaluated in a future study) included a fixed percentage of when a practice may be included (e.g. 66% successfully employed), which is agreed by the author and the experts interviewed is not the way to proceed. Suggested ways by experts included having worked at least once in all distribution scenarios or have minimal empirical grounding of a certain number of cases (rather than a percentage of all cases). It is also agreed that expanding the ADAPT framework has to be moderated and that case studies by other authors can be considered if all propositions are fulfilled (especially *TP1* that the full empirical context of the case study is reported). Hence the proposition is regarded as successfully fulfilled with regard to positive feedback from experts that the current framework design allows a variety of future scaling mechanisms, which need to be analyzed in detail in future work.

7.5 Research Questions revisited

In Chapter 1 four RQs have been posed which are now revisited after the completion of the first full iteration of the ADAPT learning framework as an output to this thesis and also taking the evaluation in focus groups and expert interviews into account. The RQs will be answered here in a compact manner.

RQ1. *Why would distributed software development benefit from agile practices?*

This introductory research question to the thesis has been tackled by analyzing related work. Based on the investigation presented in Chapter 2, the typical challenges found in DSD are related to communication, coordination and control. Previous studies indicate that they can be successfully mitigated by implementing agile practices (cf. Section 2.2). The systematic mapping study in Chapter 4 shows that there is an increasing research interest in the fifteen years of 1999-2014 (and later) to transfer agile values to DSD environments, but that there is no framework yet to achieve that task.

RQ2. *What are suitable design components for building a distributed agile process framework?*

Chapter 3 presented the complete design theory behind the ADAPT framework. Table 3.3 discussed the design components *purpose and scope*, *constructs*, *principles of form and function*, *artifact mutability*, *testable propositions*, *justificatory knowledge*, *principles of implementation* and *expository instantiation*. The design components clearly outlined the envisioned framework and five testable propositions TP1-TP5 were described to test the ADAPT design after completion. The precedent Section 7.4 revisited all five propositions and determined their successful accomplishment. It shall be noted that a future implementation study is advised to further test the propositions and that the propositions must also be satisfied in future iterations of the ADAPT framework, beyond the scope of this thesis.

RQ3a. *What does the research landscape in the field look like in the 15 years of 1999 to 2014?*

Chapter 4 examined the research landscape of agile DSD in great detail. Interesting findings include that 94.74% of the 95 finally included studies in the mapping could have been found using only the *Scopus* and *Compendex* database. The most prominent venues for publishing papers on agile DSD are ICGSE, XP and AGILE conferences as well as (to a lesser extent) the journals IST, JSS and JoS: EP. The most active countries in researching agile DSD are the United States, Finland and Germany, and the most active universities in the field are Aalto University, Universiti Teknologi PETRONAS and Blekinge Institute of Technology. The most involved countries in agile DSD were Finland and India as suppliers, UK and Denmark as customers and the United States in both categories.

The map of research types (cf. Figure 4.6) shows that applying agile practices in DSD is an active research field with a variety of research types, most prominently evaluation studies with an empirical background. Out of 95 included studies in this thesis' systematic mapping of Chapter 4 about half used

a qualitative approach and the rest is almost equally split between quantitative, mixed method or a not properly specified methodology. The most used research approach in the whole fifteen-year timespan of 1999-2014 was the case study. Otherwise there is a rise in literature reviews, which indicates a maturing research field.

The most reported cases in years 2010-2014 (47 out of 62) focus on the SE process as a whole rather than on a specific part of the workflow which indicates that research on agile practices in DSD is still quite holistic rather than in-depth. Context details were often not stated clearly such as project duration (29 cases), application domain (17) and project size (15). 45 studies explicitly stated success and only 3 reported failure, which leads to the assumption that publications are more solution-centric than problem-centric.

RQ3b. *What has changed in the later five years 2010 to 2014 in comparison to the former ten-year period of 1999-2009?*

The systematic map of research types shows that, compared to 1999-2009, there is a major increase in frequency of evaluation papers in the studied newer five-year period of 2010 to 2014 as compared to the experience papers that were most frequent in the years before (Jalali and Wohlin, 2010, 2012a). This finding indicates increasing research attention and interest to mature the field, which is also supported by a shift to a greater variety of research approaches, from close to 90% qualitative research 1999-2009 down to close to 50% 2010-2014. There is notably greater effort towards the evolvement of frameworks and models rather than mostly lessons learned before (Jalali and Wohlin, 2012a). However, none of the said models or frameworks tackle a similar scope as the ADAPT framework.

The greatest change of directions compared to 1999-2009 is the predominance of scrum, more frequent application of mixed methods and the neglecting of XP as a standalone process in agile DSD. That observation is supported in Figure 4.10, showing that XP practices have been applied fewer times successfully in 2010-2014 compared to the years of 1999 to 2009. In general underspecified context had been an issue in previous research (Jalali and Wohlin, 2010, 2012a) already. However, while the issue is definitely not resolved, most studies in the 2010-2014 studied time period at least painted a better picture for case characteristics than just "distributed teams" or "agile", which was frequent in years 1999-2009.

RQ3c. *What are common agile practices and distribution scenarios?*

Out of the 95 included studies, the majority (66 publications) had an empirical background. To encourage a full description of context in future work

a checklist was designed as a further outcome of the systematic mapping study of Chapter 4 that can be found in Appendix A.3.

The majority of reported studies had the following characteristics: offshore (43 cases), far distance (29), large time gap (23), all-agile teams (23), two site environment (22) and insourcing (16) but closely followed by outsourcing (14), i.e. complex global cases. Compared to years 1999 to 2009 there is a greater variety of involved countries being reported in general. Scrum is by far the most used agile process across all distribution scenarios (37 cases), followed by mixed methods (11), most notably the combination of scrum methodology with XP development practices.

Out of all studied papers successful practices have been extracted 309 times in the five-year period of 2010-2014 and 444 in total for the fifteen years of 1999-2014. For years 2010-2014, there is a strong support for the successful implementation of the most basic scrum practices such as standup meeting (32 cases), product owner (32, including variations of proxy product owner and product owner teams), backlog (31), sprint planning (25), retrospective (23), scrum master (21), user stories and sprint reviews (18 cases each). Neglected scrum practices (or ones that did not receive explicit attention in reports) were estimation meetings (2 cases), self-organizing teams (2) and burndown charts (2). Also, the scrum of scrums (6) has been seldom reported in DSD environments, although it is a practice to support scaling in agile processes. Overall for 1999-2014 the top three agile practices were standup meeting (50.5 cases), sprint/iterations (38.5 cases) and backlog (38 cases).

Agile practices are often supported by means to overcome distance. These means are of general nature and not related to agile methods as such (only covering years 2010-2014): video/audio conference meetings (27 cases), contact visits (11), instant messaging (10), wiki (9), screen sharing (3), ambassador (2) and chat (2). However, they are still important for the ADAPT framework as these or similar means are needed in any distributed process implementation, including agile ones.

RQ4a. *Which process design guidelines and practices can increase the chances of a successful agile process implementation in distributed environments?*

The three cases CrossTown, NoTimeshift and Continental showed that agile practices can be successfully applied to all three distribution scenarios defined by Prikladnicki et al. (2003). All three cases have been reported to the full extent of the checklist (cf. Appendix A.3) designed as an outcome of the systematic mapping study. The multiple-case study featured the following cases:

- Case CrossTown: sites distributed within one city, spanning two districts, yielding 15 conceptual guidelines and 40 conceptual practices
- Case NoTimeshift: sites distributed within one country, spanning two cities, yielding 22 conceptual guidelines and 35 conceptual practices
- Case Continental: sites distributed within one continent, spanning three countries, yielding 12 conceptual guidelines and 19 conceptual practices

So the individual single-case analysis resulted in total in 49 guideline and 94 practice *concepts*. These concepts were then analyzed, merged and combined cross-case and led to a final set of 10 guidelines, 29 full practices (with evidence in at least two out of the three cases) and 7 conceptual practices (that lack evidence in more than one case). The practices were then linked to guidelines and the guidelines in turn linked to the three challenge categories coordination, control and communication, to build the first full iteration of the ADAPT learning framework as the outcome of this thesis (cf. Table 6.3). The guidelines provide high level information and can be implemented using the concrete linked practices.

RQ4b. *Do the different distribution scenarios affect the implementation of agile practices?*

This thesis found 10 common guidelines and 29 common practices emerge from varying distribution scenarios. The global scenario has been outscoped for the ADAPT framework as it was not featured in the case study, but is planned to be added in future iterations (cf. Section 7.7). Interview #8 with an expert on global software development also confirmed that in fact most of the current practices already have value in a global scenario, only missing practices to better cope with cultural and timezone issues. The systematic mapping study, which featured many studies in global scenarios, supports that claim (cf. Table 7.5). With the given empirical basis of three case studies results show that the practices are not covering each problem to the same extent, e.g. while there are P1-P4 dealing with missing proximity in DSD, there is only one practice to address the sprint iteration (P10). Future iterations of the ADAPT framework should try to grow larger in those aspects that the current three cases could not shed more light on as of now. Nevertheless experts during evaluation interviews agreed that the current set of guidelines and practices is a very good starting point, both for an implementation study as is, as well as for adding more cases to the empirical basis.

7.6 Limitations

Systematic Mapping. *Reliability.* Threats to the reliability and validity were a great concern which is one of the reasons why multiple researchers were

involved in this systematic mapping study, effectively minimizing individual bias. Threats to study design and procedure have been discussed early in the study and finalized before the beginning of the actual searches. The inclusion process has been executed by the author, but has been double-checked by a supporting researcher to the extent presented in Section 4.2. Reliability of the procedure is also increased since this is a successful replication for the most part (cf. Section 4.2 for adaptations) of (Jalali and Wohlin, 2012a) to update trends and directions for the more recent years of 2010-2014. *Validity.* Each step along the way has been documented and presented thoroughly and extensively. All of the study has been conducted in a continuous and compact timely flow led by the author, so that all information was fresh in the minds of the researchers during execution, documentation and analysis of the study. One of the most time-consuming parts during the inclusion process was to identify replicate cases, i.e. evaluation studies, which have been published more than once. The most promising approach to counter that was to group publications by authors and try to make sure that the same case was only reported once in our included study set. Still it is a difficult long-winding process since these repeatedly reported studies also tend to not reference each other. After the complete paper draft has been set up by the first author, it has also been thoroughly reviewed and improved by all authors for construct and conclusion validity. Individual decisions may differ, but the author feels confident that the general trends and directions identified in this study would be very similar if replicated by other researchers.

Multiple-case study and the ADAPT framework. Like any empirical study this study exhibits certain threats to validity (Yin, 2003) and the generalizability of the results is limited in light of its limitations. *Construct validity* was addressed by using multiple sources of evidence, a chain of evidence and informants validated the results for each of the three cases in separate feedback sessions. To achieve *internal validity* the author employed method, data, investigator and theory triangulation. For *external validity* the author deducted a checklist for reporting context in case studies based on the systematic mapping study in Chapter 4. *Reliability* was established by following a case study protocol, a case study data drive and the data analysis software ATLAS.ti for consistent handling. The great difference in the cases' context was purposefully introduced via maximum variation (heterogeneity) sampling with the objective of finding that "a theme song emerged from all the scattered noise" (Patton, 2002, p. 235). While multiple-case studies provide more value for generalization than single-case studies, results should be regarded as specific to the individual case's context and thus generalized with caution until more robust empirical evidence is found. Case Continental did not contain a full-fledged agile process, but worked with agile development methods inside a traditional setup. This resulted in a fewer amount of guidelines and practices (due to the agile nature of the framework) but certainly enriched the

empirical basis due to the greatly distributed setting across three countries in Europe.

The current setup of challenge categories, guidelines and practices has been constructed solely by the author and then reviewed by experts during the evaluation interviews. Once the empirical basis grows in future work, it would be beneficial to employ methods such as *card sorting* (Nawaz, 2012) to further refine the setup of guidelines and practices. The current iteration (v1.0) serves as the base of the learning framework but does not cover each aspect equally and is thus currently lacking alternatives to some of the practices. Until further empirical studies are added to the framework in future iterations, the conceptual practices have been provided to be tested if a current full practice does not yield the expected results. This dissertation does not feature an implementation study of the ADAPT framework as this was out-scoped and would require different methods for validation and is planned for future work.

The focus groups and expert interviews served to reflect with both research and industrial experts on the thesis' methodology and results to minimize bias and thus reduce limitations.

7.7 Future Work

There are several directions in which the ADAPT framework can be developed in the future, following multiple of the approaches below or one exclusively.

Case Study: Implementing the ADAPT framework

Conducting one or several case studies to put the ADAPT framework v1.0 to the test is arguably the most important step (next to further building the empirical basis) for future increments of the ADAPT learning framework. It would allow to analyze the application of the current ADAPT guidelines and practices and their effectiveness and also take a closer look into how people would use the ADAPT framework, when provided with the overview table (cf. Table 6.3) and a short description of the aim of the framework and its guidelines and practices. Interesting phenomena to analyze in this regard could be which practices practitioners regard as interchangeable, possibly on different dimensions such as e.g. simplicity (easy to implement), cost and impact.

Growing the Empirical Basis: ADAPT Database and Website

Equally important to improving the usability of the framework is growing the empirical basis which currently features the three cases of this thesis: CrossTown, NoTimeshift and Continental. There are several possibilities to

grow the empirical basis, one is simply for the author to conduct more case studies, e.g. for the global scenario, which was outscoped for this thesis. Another more intriguing possibility is to invite further researchers to participate in building the framework, which would be much faster for developing a stronger empirical basis and also minimize researcher's bias. The minimum requirements for case studies to be considered as an addition to the empirical basis of the ADAPT framework is that (at least) all types of contextual information in the developed checklist (cf. Appendix A.3) are reported. For past studies, this could be accomplished by contacting the researchers of the cases identified in the systematic mapping study of this thesis to fill out the missing pieces of contextual information in their case report. Yet another approach is to develop an ADAPT database (with practices and guidelines) and a website, where researchers and practitioners can search for practices and also e.g. comment on them or suggest further practices. The latter would again require a full presentation of all contextual information in the checklist developed. However, building the ADAPT framework needs to be a moderated procedure for systematic handling, otherwise the community could add/remove/modify practices as they see fit. Once the empirical basis of practices grows larger, it would also be interesting to investigate further which practices are reported to work the most in which context (e.g. distribution scenarios).

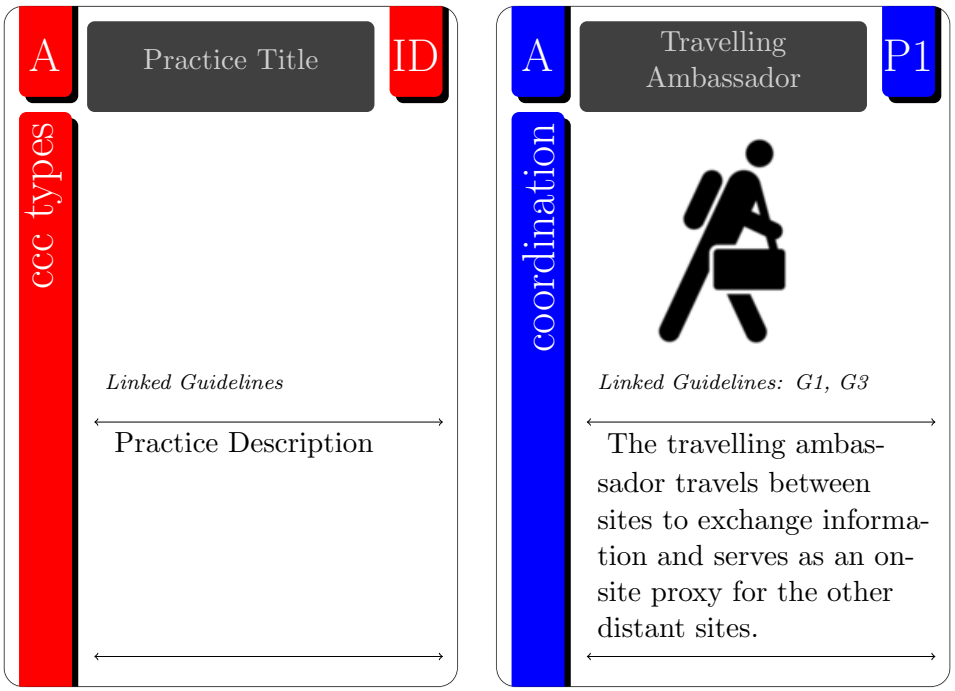
ADAPT Pattern Language

It has been noted in both focus groups that developing a pattern language (Alexander et al., 1977) is a possible future step. However, Alexander et al. describe their pattern language of 253 architectural patterns after eight years of research in the matured research field of architecture. The ADAPT framework addresses the current problem of distributed software development in the comparably new field of software engineering (in comparison to other disciplines). Hence the thesis uses *practices* instead of *patterns* to address the difference in maturity. Furthermore, the current iteration of the ADAPT framework is description-oriented (Aken, 2004). Once the empirical basis grows larger, evolving the ADAPT framework to a (possibly more prescription-oriented) pattern language in future iterations could be very interesting and rewarding to both research and practice, similar to Alexander et al.'s contribution to the field of architecture.

Gamification: ADAPT Practice Cards

An interesting approach could be to turn each practice into a playing card as Figure 7.1 showcases. The practice cards can be used for a kick-off workshop (when deciding on an initial set of practices) and/or in each retrospective (when deciding how to adapt the current process implementation). Agile processes often rely on physical artifacts such as the paper boards and sticky

notes and using cards is nothing new to the agile practitioner as many have used *planning poker* (Grenning, 2002) with success. The advantage of physical cards is that the practices become more tangible and the cards can be grouped or aligned on the table without further ado to create the new process implementation. No other tools would be required than a printer and a pair of scissors. In a distributed environment it depends of course on how the retrospective is set up, if it is held in a video conference setting, then physical cards are of lesser use, but the cards could also be made available electronically (which could be a further improvement after the initial implementation of the physical card game). Both physical and electronic cards would make an interesting subject for future studies.



(a) ADAPT schematic practice card. **(b)** ADAPT example card for P1.

Figure 7.1 – Gamification approach: card game for practices to be used during kick-off and retrospective (cards are in real-world size). Icon "traveller" by Alexander Wiefel from the Noun Project used under CC BY 3.0 US.

CHAPTER 8

Conclusion

This thesis created the first iteration of the ADAPT framework featuring 10 guidelines, 29 full practices (with evidence in at least two cases of the multiple-case study) and 7 conceptual practices (lacking evidence in more than one case). The guidelines and practices are regarded as more robust as they emerged from three different distribution scenarios. The ADAPT framework can be used both by researchers to further expand the empirical basis and by practitioners to drive their process implementation in agile DSD environments.
The ADAPT framework is based on a design theory that has been discussed and evaluated during a four-month research visit at Stanford University's Center for Design Research and features the following core testable propositions (TP):

TP1 Each practice and guideline of the ADAPT framework is grounded in empirical evidence.

TP2 The ADAPT framework allows a simple, pragmatic and iterative process tailoring (rather than planned and strictly managed).

TP3 The ADAPT framework supports project-based process tailoring (rather than organization-based).

TP4 The ADAPT framework provides tangible and detailed advice to the practitioner.

TP5 The ADAPT framework is easily extensible.

The surrounding research field has been extensively analyzed in a systematic mapping study, offering numerous insights, most prominently the frequency of various practices in agile DSD. The mapping study also resulted in a checklist

for reporting context in agile DSD (cf. Appendix A.3) and thus serves as an additional result of this thesis. The checklist has been used to report empirical context in the multiple-case study conducted which covered the following distribution scenarios:

- **Case CrossTown:** sites distributed within one city, spanning two districts, yielding 15 conceptual guidelines and 40 conceptual practices
- **Case NoTimeshift:** sites distributed within one country, spanning two cities, yielding 22 conceptual guidelines and 35 conceptual practices
- **Case Continental:** sites distributed within one continent, spanning three countries, yielding 12 conceptual guidelines and 19 conceptual practices

The conceptual guidelines and practices from each single-case analysis were then aggregated cross-case to arrive at the final set of 10 guidelines, 29 full practices and 7 conceptual practices. The guidelines were connected to the three challenge categories *Coordination, Control and Communication*. This three-tiered hierarchy of challenge categories, guidelines and practices together forms the first full iteration of the ADAPT framework as the outcome of this thesis.
The ADAPT framework is regarded as a learning framework, which means that it is designed with future iterations in mind to further improve both the empirical basis (more cases to deduct more guidelines and practices) and the utility of the framework for practitioners, by implementing the framework in a case study. To evaluate this first iteration of the ADAPT framework, two focus groups, one at Stanford University and one at the XP 2014 conference, have been held and ten expert interviews have been conducted. The feedback has been overwhelmingly positive, acknowledging this research endeavor as a grand challenge, well-designed and fit for future iterations. Three experts even offered to implement the framework in one of their projects, right after the interview has ended, which would be an important next step for future work. Other improvements, as discussed with experts during interviews, in future work will target growing the empirical basis by possibly inviting other researchers in, which could be facilitated by a database/website, and making the implementation more tangible by providing a set of physical practice cards for kick-off workshops and retrospectives.

List of Figures

List of Tables

Bibliography

Abran, A., Moore, J. W., Bourque, P., Dupuis, R. and Tripp, L. *Guide to the software engineering body of knowledge*. IEEE Computer Society, 2004.

Ågerfalk, J. and Fitzgerald, B. Flexible and distributed software processes: old petunias in new bowls. In *Communications of the ACM*, 2006.

Ågerfalk, P. J., Fitzgerald, B., Holmström, H., Lings, B., Lundell, B. and Conchúir, Ó. A framework for considering opportunities and threats in distributed software development. In *Proceedings of the International Workshop on Distributed Software Development (Paris, Aug. 29, 2005). Austrian Computer Society*, pp. 47–61, 2005.

Akbar, R., Hassan, M., Qureshi, M. and Safdar, S. Structured role based interaction model for agile based outsourced IT projects: Client's composite structure. *Information Technology Journal*, 10(5):1009 – 1016, 2011a. ISSN 18125638.

Akbar, R. and Hassan, M. F. A collaborative-interaction model of software project development: An extension to agile based methodologies. In *Information Technology (ITSim), 2010 International Symposium in*, volume 1, pp. 1–6. IEEE, 2010.

Akbar, R., Hassan, M. F. and Abdullah, A. A Review of Prominent Work on Agile Processes Software Process Improvement and Process Tailoring Practices. In *Software Engineering and Computer Systems*, pp. 571–585. Springer, 2011b.

Akbar, R., Hassan, M. F., Abdullah, A., Safdar, S. and Qureshi, M. A. Directions and advancements in global software development: A summarized review of GSD and agile methods. *Research Journal of Information Technology*, 3(2):69–89, 2011c.

Akbar, R., Hassan, M. F. and Abdullah, A. A framework of software process tailoring for small and medium size IT companies. In *Computer & Information Science (ICCIS), 2012 International Conference on*, volume 2, pp. 914–918. IEEE, 2012.

Aken, J. E. v. Management research based on the paradigm of the design sciences: the quest for field-tested and grounded technological rules. *Journal of management studies*, 41(2): 219–246, 2004.

Alexander, C. *The timeless way of building*, volume 1. Oxford University Press, 1979.

Alexander, C., Ishikawa, S. and Silverstein, M. *A pattern language: towns, buildings, construction*, volume 2. Oxford University Press, 1977.

Almeida, L. H., Albuquerque, A. B. and Pinheiro, P. R. A multi-criteria model for planning and fine-tuning distributed scrum projects. In *Global Software Engineering (ICGSE), 2011 6th IEEE International Conference on*, pp. 75–83. IEEE, 2011.

Alqahtani, A. S., Moore, J. D., Harrison, D. K. and Wood, B. M. The Challenges of Applaying Distributed Agile Software Development: A Systematic Review. *International Journal of Advances in Engineering & Technology*, 5(2):23–36, January 2013.

Alsmadi, I. and Saeed, S. A software development process for open source and open competition projects. *International Journal of Business Information Systems*, 12(1):110–122, 2013.

Alyahya, S., Ivins, W. K. and Gray, W. Raising the Awareness of Development Progress in Distributed Agile Projects. *Journal of Software*, 8(12):3066–3081, 2013.

Ambler, S. *Agile modeling: effective practices for extreme programming and the unified process*. John Wiley & Sons, 2002.

Ambler, S. W. Questioning "Best Practices" for Software Development: Practices are Contextual, Never Best, 2011. URL `http://www.ambysoft.com/essays/bestPractices.html`.

Anderson, D. J. *Agile management for software engineering: Applying the theory of constraints for business results*. Prentice Hall Professional, 2003.

Anderson, D. J. *Kanban: Successful Evolutionary Change For Your Technology Business*. Blue Hole Press, 2010.

Anderson, L., Alleman, G. B., Beck, K., Blotner, J., Cunningham, W., Poppendieck, M. and Wirfs-Brock, R. Agile management-an oxymoron?: who needs managers anyway? In *Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, pp. 275–277. ACM, 2003.

Ansari, A. A. and Ansari, A. Enabling to Apply XP Process in Distributed Development Environments with Tool Support. *International Journal of Computer Science Issues(IJCSI)*, 9(4):272–276, 2012.

Ansari, A. A., Sharafi, S. M. and Nematbakhsh, N. A method for requirements management in distributed extreme programming environment. *Journal of Theoretical & Applied Information Technology*, 20(1):52–58, 2010.

Ashraf, M. A., Shamail, S., Rana, Z. et al. Agile model adaptation for e-learning students' final-year project. In *Teaching, Assessment and Learning for Engineering (TALE), 2012 IEEE International Conference on*, pp. T1C–18. IEEE, 2012.

Badampudi, D., Fricker, S. A. and Moreno, A. M. Perspectives on Productivity and Delays in Large-Scale Agile Projects. In *Agile Processes in Software Engineering and Extreme Programming: 14th International Conference, XP 2013, Vienna, Austria, June 3-7, 2013, Proceedings*, volume 149, p. 180. Springer, 2013.

Bailey, J., Budgen, D., Turner, M., Kitchenham, B., Brereton, P. and Linkman, S. Evidence relating to Object-Oriented software design: A survey. In *Empirical Software Engineering and Measurement, 2007. ESEM 2007. First International Symposium on*, pp. 482–484. IEEE, 2007.

Bandukda, M. and Nasir, Z. Efficacy of distributed pair programming. In *Information and Emerging Technologies (ICIET), 2010 International Conference on*, pp. 1–6. IEEE, 2010.

Bannerman, P. L., Hossain, E. and Jeffery, R. Scrum practice mitigation of global software development coordination challenges: A distinctive advantage? In *System Science (HICSS), 2012 45th Hawaii International Conference on*, pp. 5309–5318. IEEE, 2012.

Baskerville, R. and Pries-Heje, J. Discursive Co-development of Agile Systems and Agile Methods. In *Grand Successes and Failures in IT. Public and Private Sectors*, pp. 279–294. Springer, 2013.

Bass, J. M. Influences on agile practice tailoring in enterprise software development. In *AGILE India (AGILE INDIA), 2012*, pp. 1–9. IEEE, 2012.

Bass, J. M. How product owner teams scale agile methods to large distributed enterprises. *Empirical Software Engineering*, In-Print 20(6):1525–1557, 2015, Online–First, 2014.

Batra, D. Modified agile practices for outsourced software projects. *Communications of the ACM*, 52(9):143–148, 2009.

Batra, D., Xia, W., VanderMeer, D. and Dutta, K. Balancing agile and structured development approaches to successfully manage large distributed software projects: A case study from the cruise line industry. *Communications of the Association for Information Systems*, 27(1):21, 2010.

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

Beck, K. Embracing change with extreme programming. *Computer*, 32(10):70–77, 1999.

Beck, K. *Extreme programming explained: embrace change*. Addison-Wesley Professional, 2000.

Beck, K. *Test-driven development: by example*. Addison-Wesley Professional, 2003.

Begel, A. and Nagappan, N. Usage and perceptions of agile software development in an industrial context: An exploratory study. In *Empirical Software Engineering and Measurement, 2007. ESEM 2007. First International Symposium on*, pp. 255–264. IEEE, 2007.

Belsis, P., Koutoumanos, A. and Sgouropoulou, C. PBURC: a patterns-based, unsupervised requirements clustering framework for distributed agile software development. *Requirements Engineering*, 19(2):213–225, 2014.

Bocock, L. and Martin, A. There's something about lean: A case study. In *Agile Conference (AGILE), 2011*, pp. 10–19. IEEE, 2011.

Boehm, B. Guidelines for Verifying and Validating Software Requirements and Design Specifications. Technical report, University of Southern California, 1979.

Boehm, B. *Software risk management*, pp. 1–19. ESEC '89: 2nd European Software Engineering Conference University of Warwick, Coventry, UK September 11–15, 1989 Proceedings. Springer Berlin Heidelberg, Berlin, Heidelberg, 1989. ISBN 978-3-540-46723-6.

Boehm, B. W. A spiral model of software development and enhancement. *Computer*, 21(5): 61–72, 1988.

Brereton, P., Kitchenham, B., Budgen, D. and Li, Z. Using a protocol template for case study planning. In *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering. University of Bari, Italy*, 2008.

Bryman, A., Burgess, B. et al. *Analyzing qualitative data*. Routledge, 2002.

Carmel, E. *Global software teams: collaborating across borders and time zones*. Prentice Hall PTR, 1999.

Carmel, E. and Agarwal, R. Tactical approaches for alleviating distance in global software development. *Software, IEEE*, 18(2):22–29, 2001.

Ceria, S. and Pallotti, C. Argentinas Offshore Software Industry–Opportunities and Challenges. In *Software Engineering Approaches for Offshore and Outsourced Development*, pp. 23–36. Springer, 2010.

Checkland, P. and Holwell, S. Action research: its nature and validity. *Systemic Practice and Action Research*, 11(1):9–21, 1998.

Chen, J. Q., Phan, D., Wang, B. and Vogel, D. R. Light-Weight Development Method: a Case Study. In *Service Systems and Service Management, 2007 International Conference on*, pp. 1–6. IEEE, 2007.

Chung, M.-W. and Drummond, B. Agile at yahoo! from the trenches. In *Agile Conference, 2009. AGILE'09.*, pp. 113–118. IEEE, 2009.

Cocco, L., Mannaro, K. and Concas, G. A Model for Global Software Development with Cloud Platforms. In *Software Engineering and Advanced Applications (SEAA), 2012 38th EUROMICRO Conference on*, pp. 446–452. IEEE, 2012.

Cockburn, A. and Williams, L. Agile software development: It's about feedback and change. *Computer*, 36(6):39–43, 2003.

Cohn, M. *User stories applied: For agile software development*. Addison-Wesley Professional, 2004.

Cohn, M. *Agile estimating and planning*. Pearson Education, 2005.

Coplien, J. O. *A generative development process pattern language*. Cambridge University Press, New York, 1998.

Coyne, R. *Designing information technology in the postmodern age: From method to metaphor*. Mit Press, 1995.

Cristal, M., Wildt, D. and Prikladnicki, R. Usage of Scrum practices within a global company. In *Global Software Engineering, 2008. ICGSE 2008. IEEE International Conference on*, pp. 222–226. IEEE, 2008.

Cunningham, W. Episodes: A pattern language of competitive development. In *Pattern languages of program design 2*, pp. 371–388. Addison-Wesley Longman Publishing Co., Inc., 1996.

da Silva, F. Q., Costa, C. and Prikladinicki, R. Challenges and solutions in distributed software development project management: a systematic literature review. In *Global Software Engineering (ICGSE), 2010 5th IEEE International Conference on*, pp. 87–96. IEEE, 2010.

da Silva Estácio, B. J. and Prikladnicki, R. A Set of Practices for Distributed Pair Programming. In *International Conference on Enterprise Information Systems*, pp. 331–338, 2014.

da Silva Estácio, B. J. and Prikladnicki, R. Distributed Pair Programming: A Systematic Literature Review. *Information and Software Technology*, 63:1–10, 2015.

Damian, D. and Moitra, D. Guest Editors' Introduction: Global Software Development: How Far Have We Come? *Software, IEEE*, 23(5):17–19, 2006.

Damian, D. E. and Zowghi, D. RE challenges in multi-site software development organisations. *Requirements engineering*, 8(3):149–160, 2003.

Daneva, M. and Ahituv, N. What agile ERP consultants think of requirements engineering for inter-organizational ERP Systems: Insights from a Focus Group in BeNeLux. In *Evaluation & Assessment in Software Engineering (EASE 2012), 16th International Conference on*, pp. 284–288. IET, 2012.

Daneva, M., Van Der Veen, E., Amrit, C., Ghaisas, S., Sikkel, K., Kumar, R., Ajmeri, N., Ramteerthkar, U. and Wieringa, R. Agile requirements prioritization in large-scale outsourced system projects: An empirical study. *Journal of systems and software*, 86(5): 1333–1353, 2013.

Davies, R. and Sedley, L. *Agiles Coaching: Praxis-Handbuch für ScrumMaster, Teamleiter und Projektmanager in der agilen Software-Entwicklung*. Hüthig Jehle Rehm, 2010.

Davison, R., Martinsons, M. G. and Kock, N. Principles of canonical action research. *Information systems journal*, 14(1):65–86, 2004.

del Nuevo, E., Piattini, M. and Pino, F. J. Scrum-based methodology for distributed software development. In *Global Software Engineering (ICGSE), 2011 6th IEEE International Conference on*, pp. 66–74. IEEE, 2011.

DeMarco, T. and Lister, T. *Peopleware: Productive Projects and Teams Dorset House*. Dorset House Publishing Co., Inc., 1999.

Denzin, N. *The research act: a theoretical introduction to sociological methods*. McGraw-Hill, 2nd edition, 1978.

Denzin, N. K. and Lincoln, Y. S. *The SAGE handbook of qualitative research*. Sage, 2011.

Derby, E., Larsen, D. and Schwaber, K. *Agile retrospectives: Making good teams great*. Pragmatic Bookshelf Raleigh, NC, 2006.

Dingsøyr, T., Nerur, S., Balijepally, V. and Moe, N. B. A decade of agile methodologies: Towards explaining agile software development. *Journal of Systems and Software*, 85(6): 1213–1221, 2012.

Dogs, C. and Klimmer, T. *Agile Software-Entwicklung kompakt*. mitp, 2005.

Dorairaj, S., Noble, J. and Malik, P. Understanding Lack of Trust in Distributed Agile Teams: A grounded theory study. In *Evaluation & Assessment in Software Engineering (EASE 2012), 16th International Conference on*, pp. 81–90. IET, 2012a.

Dorairaj, S., Noble, J. and Malik, P. Understanding the Importance of Trust in Distributed Agile Projects: A Practical Perspective. In *Agile Processes in Software Engineering and Extreme Programming: 11th International Conference, XP 2010, Trondheim, Norway, June 1-4, 2010, Proceedings*, volume 48, p. 172. Springer Science & Business Media, 2010.

Dorairaj, S., Noble, J. and Malik, P. Effective Communication in Distributed Agile Software Development Teams. In *Agile Processes in Software Engineering and Extreme Programming: 12th International Conference, XP 2011, Madrid, Spain, May 10-13, 2011, Proceedings*, volume 77, p. 102. Springer Science & Business Media, 2011.

Dorairaj, S., Noble, J. and Malik, P. Knowledge management in distributed agile software development. In *Agile Conference (AGILE), 2012*, pp. 64–73. IEEE, 2012b.

Dorairaj, S., Noble, J. and Allan, G. Agile software development with distributed teams: Senior management support. In *Global Software Engineering (ICGSE), 2013 IEEE 8th International Conference on*, pp. 197–205. IEEE, 2013.

Dullemond, K., van Gameren, B. and van Solingen, R. How technological support can enable advantages of agile software development in a GSE setting. In *Global Software Engineering, 2009. ICGSE 2009. Fourth IEEE International Conference on*, pp. 143–152. IEEE, 2009.

Dumitriu, F., Oprea, D. and Mesnita, G. Issues and strategy for agile global software development adoption. *Recent researches in Applied Economics*, pp. 37–42, 2011.

Dybå, T., Kampenes, V. B. and Sjøberg, D. I. A systematic review of statistical power in software engineering experiments. *Information and Software Technology*, 48(8):745–755, 2006.

Eckstein, J. *Agile Software Development with Distributed Teams*. Dorset House Publishing Co., Inc., 2010.

Eckstein, J. *Agile software development with distributed teams: Staying agile in a global world*. Addison-Wesley, 2013.

Estler, H.-C., Nordio, M., Furia, C. A., Meyer, B. and Schneider, J. Agile vs. structured distributed software development: A case study. *Empirical Software Engineering*, 19(5): 1197–1224, 2014.

Falcon, A. Aristotle on Causality. In *The Stanford Encyclopedia of Philosophy*. Spring 2014 edition, 2014. URL `http://plato.stanford.edu/archives/spr2014/entries/aristotle-causality/`.

Falessi, D., Oliveira, R., Taylor, K., Fontana, R. M., Power, K., Vallon, R., Giardino, C., Rejab, M. M. and Wang, X. Trends and emerging areas of agile research: the report on XP2014 PhD symposium. *ACM SIGSOFT Software Engineering Notes*, 39(5):26–29, 2014.

Femmer, H., Kuhrmann, M., Stimmer, J. and Junge, J. Experiences from the Design of an Artifact Model for Distributed Agile Project Management. In *Global Software Engineering (ICGSE), 2014 IEEE 9th International Conference on*, pp. 1–5. IEEE, 2014.

Fernando, B. A. J., Hall, T. and Fitzpatrick, A. The impact of media selection on stakeholder communication in agile global software development: a preliminary industrial case study. In *Proceedings of the 49th SIGMIS annual conference on Computer personnel research*, pp. 131–139. ACM, 2011.

Fowler, M. and Highsmith, J. The agile manifesto. *Software Development*, 9(8):28–35, 2001.

Fraser, S., Rising, L., Ambler, S., Cockburn, A., Eckstein, J., Hussman, D., Miller, R., Striebeck, M. and Thomas, D. A fishbowl with piranhas: coalescence, convergence or divergence? In *Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications*, pp. 937–939. ACM, 2006.

Gilb, T. and Finzi, S. *Principles of software engineering management*, volume 4. Addison-Wesley Reading, MA, 1988.

Glass, R. L., Vessey, I. and Ramesh, V. Research in software engineering: an analysis of the literature. *Information and Software technology*, 44(8):491–506, 2002.

Gloger, B. *Scrum: Produkte zuverlässig und schnell entwickeln*. Hanser, 2011.

Grechenig, T., Bernhart, M., Breiteneder, R. and Kappel, K. *Softwaretechnik: mit Fallbeispielen aus realen Entwicklungsprojekten*. Pearson Deutschland GmbH, 2010.

Green, R., Mazzuchi, T. and Sarkani, S. Communication and quality in distributed agile development: an empirical case study. *Proceeding in World Academy of Science, Engineering and Technology*, 61:322–328, 2010a.

Green, R., Mazzuchi, T. and Sarkani, S. Understanding the role of synchronous & asynchronous communication in agile software development and its effects on quality. *Journal of Information Technology Management*, 21(2):8, 2010b.

Greenleaf, R. K. *Servant leadership: A journey into the nature of legitimate power and greatness*. Paulist Press, 2002.

Gregg, D. G., Kulkarni, U. R. and Vinzé, A. S. Understanding the philosophical underpinnings of software engineering research in information systems. *Information Systems Frontiers*, 3(2):169–183, 2001.

Gregor, S. and Jones, D. The anatomy of a design theory. *Journal of the Association for Information Systems*, 8(5):312–335, 2007.

Grenning, J. Planning poker or how to avoid analysis paralysis while release planning, 2002. URL `http://www.renaissancesoftware.net/files/articles/PlanningPoker-v1.1.pdf`.

Hallikainen, M. Experiences on agile seating, facilities and solutions: multisite environment. In *Global Software Engineering (ICGSE), 2011 6th IEEE International Conference on*, pp. 119–123. IEEE, 2011.

Hamid, A. M. E. Upgrading distributed agile development. In *Computing, Electrical and Electronics Engineering (ICCEEE), 2013 International Conference on*, pp. 709–714. IEEE, 2013.

Hannay, J. E., Sjøberg, D. I. and Dybå, T. A systematic review of theory use in software engineering experiments. *Software Engineering, IEEE Transactions on*, 33(2):87–107, 2007.

Hanssen, G. K., Westerheim, H. and Bjørnson, F. O. Tailoring RUP to a defined project type: A case study. In *Product Focused Software Process Improvement*, pp. 314–327. Springer, 2005.

Hanssen, G. K., Šmite, D. and Moe, N. B. Signs of agile trends in global software engineering research: A tertiary study. In *Global Software Engineering Workshop (ICGSEW), 2011 Sixth IEEE International Conference on*, pp. 17–23. IEEE, 2011.

Heeager, L. T. and Rose, J. Optimising agile development practices for the maintenance operation: nine heuristics. *Empirical Software Engineering*, In-Print 20:1762–1784, 2015, Online–First, 2014.

Herbsleb, J. D. and Mockus, A. An empirical study of speed and communication in globally distributed software development. *Software Engineering, IEEE Transactions on*, 29(6): 481–494, 2003.

Herbsleb, J. D. and Moitra, D. Global software development. *Software, IEEE*, 18(2):16–20, 2001.

Herbsleb, J. D., Atkins, D. L., Boyer, D. G., Handel, M. and Finholt, T. A. Introducing instant messaging and chat in the workplace. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 171–178. ACM, 2002.

Herriott, R. E. and Firestone, W. A. Multisite qualitative policy research: Optimizing description and generalizability. *Educational researcher*, pp. 14–19, 1983.

Highsmith, J. *Agile Project Management: Creating Innovative Products*. Addison Wesley, 2004.

Hildenbrand, T., Geisser, M., Kude, T., Bruch, D. and Acker, T. Agile methodologies for distributed collaborative development of enterprise applications. In *Complex, Intelligent and Software Intensive Systems, 2008. CISIS 2008. International Conference on*, pp. 540–545. IEEE, 2008.

Holmström, H., Fitzgerald, B., Ågerfalk, P. J. and Ó Conchúir, E. Agile practices reduce distance in global software development. *Information Systems Management*, 23(3):7–18, 2006.

Hong, N., Yoo, J. and Cha, S. Customization of scrum methodology for outsourced e-commerce projects. In *Software Engineering Conference (APSEC), 2010 17th Asia Pacific*, pp. 310–315. IEEE, 2010.

Hooker, R. Aristotle: The Four Causes-Physics II. 3, 1996. URL `http://richard-hooker.com/sites/worldcultures/GREECE/4CAUSES.HTM`.

Hossain, E., Babar, M. A. and Paik, H.-y. Using scrum in global software development: a systematic literature review. In *Global Software Engineering, 2009. ICGSE 2009. Fourth IEEE International Conference on*, pp. 175–184, 2009.

Hossain, E., Bannerman, P. L. and Jeffery, D. R. Scrum practices in global software development: a research framework. In *Product-focused software process improvement*, pp. 88–102. Springer, 2011a.

Hossain, E., Bannerman, P. L. and Jeffery, R. Towards an understanding of tailoring scrum in global software development: a multi-case study. In *Proceedings of the 2011 International Conference on Software and Systems Process*, pp. 110–119. ACM, 2011b.

IEEE. 610.12-1990 Standard glossary of software engineering terminology, 1990.

Inayat, I., Salim, S. S. and Kasirun, Z. M. Socio-technical aspects of requirements-driven collaboration (RDC) in agile software development methods. In *Open Systems (ICOS), 2012 IEEE Conference on*, pp. 1–6. IEEE, 2012.

Jacobson, I. *Object-oriented software engineering*. Addison-Wesley, 1994.

Jalali, S. and Wohlin, C. Agile practices in global software engineering-A systematic map. In *Global Software Engineering (ICGSE), 2010 5th IEEE International Conference on*, pp. 45–54. IEEE, 2010.

Jalali, S. and Wohlin, C. Global software engineering and agile practices: a systematic review. *Journal of Software: Evolution and Process*, 24(6):643–659, 2012a.

Jalali, S. and Wohlin, C. Systematic literature studies: database searches vs. backward snowballing. In *Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement*, pp. 29–38. ACM, 2012b.

Janes, A. and Succi, G. To pull or not to pull. In *Proceedings of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications*, pp. 889–894. ACM, 2009.

Jiménez, M., Piattini, M. and Vizcaíno, A. Challenges and improvements in distributed software development: A systematic review. *Advances in Software Engineering*, 2009:3, 2009.

Jones, D. T., Roos, D. and Womack, J. P. *Machine that Changed the World*. Simon and Schuster, 1990.

Kajko-Mattsson, M., Azizyan, G. and Magarian, M. K. Classes of distributed Agile development problems. In *Agile Conference (AGILE), 2010*, pp. 51–58. IEEE, 2010.

Kamaruddin, N. K., Arshad, N. H. and Mohamed, A. Chaos issues on communication in Agile Global Software Development. In *Business Engineering and Industrial Applications Colloquium (BEIAC), 2012 IEEE*, pp. 394–398. IEEE, 2012.

Kampenes, V. B., Dybå, T., Hannay, J. E. and Sjøberg, D. I. A systematic review of effect size in software engineering experiments. *Information and Software Technology*, 49(11): 1073–1086, 2007.

Kanwal, F., Bashir, K. and Ali, A. H. Documentation Practices for Offshore Agile Software Development. *Life Science Journal*, 11(10s), 2014.

Khan, M. I., Qureshi, M. A. and Abbas, Q. Agile methodology in software development (SMEs) of Pakistan software industry for successful software projects (CMM framework). In *Educational and Network Technology (ICENT), 2010 International Conference on*, pp. 576–580. IEEE, 2010.

Kitchenham, B. and Charters, S. Guidelines for performing systematic literature reviews in software engineering. Technical Report EBSE-2007-01, School of Computer Science and Mathematics, Keele University, 2007.

Kitchenham, B., Pickard, L. and Pfleeger, S. L. Case studies for method and tool evaluation. *IEEE software*, 12(4):52–62, 1995.

Klein, H., Knauss, E. and Rausch, A. Scaling Software Development Methods from Colocated to Distributed. In *Software Quality: 4th International Conference, SWQD 2012, Vienna, Austria, January 17-19, 2012, Proceedings*, volume 94, p. 71. Springer, 2012.

Klimpke, L., Kramer, T., Betz, S. and Nordheimer, K. Globally distributed software development in small and medium-sized enterprises in germany: Reasons, locations, and obstacles. In *ECIS 2011 Proceedings*, number 118, 2011.

Kniberg, H. *Scrum and XP from the Trenches*. Lulu.com, 2007.

Kniberg, H. and Skarin, M. *Kanban and Scrum-making the most of both*. Lulu.com, 2010.

Korhonen, K. Evaluating the effect of agile methods on software defect data and defect reporting practices-a case study. In *Quality of Information and Communications Technology (QUATIC), 2010 Seventh International Conference on the*, pp. 35–43. IEEE, 2010.

Korkala, M. and Abrahamsson, P. Communication in distributed agile development: A case study. In *Software Engineering and Advanced Applications, 2007. 33rd EUROMICRO Conference on*, pp. 203–210. IEEE, 2007.

Korkala, M., Pikkarainen, M. and Conboy, K. A case study of customer communication in globally distributed software product development. In *Proceedings of the 11th International Conference on Product Focused Software*, pp. 43–46. ACM, 2010.

Kruchten, P. *The rational unified process: an introduction*. Addison-Wesley Professional, 2004.

Kuhrmann, M., Mendez Fernandez, D. and Grober, M. Towards artifact models as process interfaces in distributed software projects. In *Global Software Engineering (ICGSE), 2013 IEEE 8th International Conference on*, pp. 11–20. IEEE, 2013.

Ladas, C. *Scrumban-essays on kanban systems for lean software development*. Modus Cooperandi Press, 2008.

Lakoff, G. and Johnson, M. *Philosophy in the flesh: The embodied mind and its challenge to western thought*. Basic books, 1998.

Larman, C. and Vodde, B. *Scaling lean & agile development: thinking and organizational tools for large-scale Scrum*. Addison-Wesley, 2009.

Lavazza, L., Morasca, S., Taibi, D. and Tosi, D. Applying SCRUM in an OSS Development Process: An Empirical Evaluation. In *Agile Processes in Software Engineering and Extreme Programming: 11th International Conference, XP 2010, Trondheim, Norway, June 1-4, 2010, Proceedings*, volume 48, p. 147. Springer Science & Business Media, 2010.

Lee, J. C., Judge, T. K. and McCrickard, D. S. Evaluating extreme scenario-based design in a distributed agile team. In *CHI'11 Extended Abstracts on Human Factors in Computing Systems*, pp. 863–877. ACM, 2011.

Lehtinen, T. O., Virtanen, R., Viljanen, J. O., Mäntylä, M. V. and Lassenius, C. A tool supporting root cause analysis for synchronous retrospectives in distributed software teams. *Information and Software Technology*, 56(4):408–437, 2014.

Lehtinen, T. O., Virtanen, R., Heikkilä, V. T. and Itkonen, J. Why the Development Outcome Does Not Meet the Product Owners Expectations? In *Agile Processes, in Software Engineering, and Extreme Programming: 16th International Conference, XP 2015, Helsinki, Finland, May 25-29, 2015, Proceedings*, volume 212, p. 93. Springer, 2015.

Lencioni, P. M. and Schieberle, A. *Die 5 Dysfunktionen eines Teams*. John Wiley & Sons, 2014.

Lewin, K. Action research and minority problems. *Journal of social issues*, 2(4):34–46, 1946.

Licorish, S. A. and MacDonell, S. G. How Do Globally Distributed Agile Teams Self-organise?-Initial Insights from a Case Study. In *ENASE*, pp. 157–164, 2013.

Lipnack, J. and Stamps, J. *Virtual teams: Reaching across space, time, and organizations with technology*. Wiley, 1997.

Maher, P. E., Kourik, J. L. and Chookittikul, W. Exploratory Study of Agile Methods in the Vietnamese Software Industry. In *Computing in the Global Information Technology (ICCGI), 2010 Fifth International Multi-Conference on*, pp. 300–304. IEEE, 2010.

Malone, T. W. and Crowston, K. The interdisciplinary study of coordination. *ACM Computing Surveys (CSUR)*, 26(1):87–119, 1994.

Mar, K. and Schwaber, K. Scrum with XP. *Informit.com*, 2002. URL `http://www.informit.com/articles/article.aspx?p=26057`.

Marques, A. B., Rodrigues, R. and Conte, T. Systematic literature reviews in distributed software development: A tertiary study. In *Global Software Engineering (ICGSE), 2012 IEEE Seventh International Conference on*, pp. 134–143. IEEE, 2012.

Martin, J. *Rapid application development*. Macmillan Publishing Co., Inc., 1991.

Martínez-Ruiz, T., Münch, J., García, F. and Piattini, M. Requirements and constructors for tailoring software processes: a systematic literature review. *Software Quality Journal*, 20(1):229–260, 2012.

McCann, J. and Galbraith, J. R. Interdepartmental relations. *Handbook of organizational design*, 2:60–84, 1981.

McKay, J. and Marshall, P. The dual imperatives of action research. *Information Technology & People*, 14(1):46–59, 2001.

Meyer, S., Knauss, E. and Schneider, K. Distributing a Lean Organization: Maintaining Communication While Staying Agile. In *Lean Enterprise Software and Systems: First International Conference, LESS 2010, Helsinki, finland, October 17-20, 2010, Proceedings*, volume 65, p. 99. Springer, 2010.

Miles, M. B. and Huberman, A. *Qualitative Data Analysis*. SAGE, 1994.

Mirakhorli, M., Khanipour Rad, A., Shams, F., Pazoki, M. and Mirakhorli, A. RDP technique: A practice to customize XP. In *Proceedings of the 2008 international workshop on Scrutinizing agile practices or shoot-out at the agile corral*, pp. 23–32. ACM, 2008.

Modi, S., Abbott, P. and Counsell, S. Negotiating common ground in distributed agile development: A case study perspective. In *Global Software Engineering (ICGSE), 2013 IEEE 8th International Conference on*, pp. 80–89. IEEE, 2013.

Morgan, D. L. *The focus group guidebook*, volume 1. Sage publications, 1997.

Mountain Goat Software, . Scrum Overview for Agile Software Development, 2005. URL `http://www.mountaingoatsoftware.com/agile/scrum/overview`.

Mudumba, V. and Lee, O.-K. A new perspective on GDSD risk management: agile risk management. In *Global Software Engineering (ICGSE), 2010 5th IEEE International Conference on*, pp. 219–227. IEEE, 2010.

Nagle, T., McAvoy, J. and Sammon, D. Utilising mindfulness to analyse agile global software development. In *ECIS 2011 Proceedings*, p. 119, 2011.

Nawaz, A. A Comparison of Card-sorting Analysis Methods. In *The 10th Asia Pacific Conference on Computer Human Interaction. 2012*, pp. 583–592, 2012.

Nawaz, A. I. and Zualkernan, I. A. The role of agile practices in disaster management and recovery: a case study. In *Proceedings of the 2009 Conference of the Center for Advanced Studies on Collaborative Research*, pp. 164–173. IBM Corp., 2009.

Nevo, S. and Chengalur-Smith, I. Enhancing the performance of software development virtual teams through the use of agile methods: a pilot study. In *System Sciences (HICSS), 2011 44th Hawaii International Conference on*, pp. 1–10. IEEE, 2011.

Niinimaki, T. Face-to-face, email and instant messaging in distributed agile software development project. In *Global Software Engineering Workshop (ICGSEW), 2011 Sixth IEEE International Conference on*, pp. 78–84. IEEE, 2011.

Noordeloos, R., Manteli, C. and Van Vliet, H. From RUP to Scrum in global software development: A case study. In *Global Software Engineering (ICGSE), 2012 IEEE Seventh International Conference on*, pp. 31–40. IEEE, 2012.

North, D. Introducing BDD. *Better Software*, March, 2006.

Nurmi, A., Hallikainen, P. and Rossi, M. Coordination of Outsourced Information System Development in Multiple Customer Environment-A Case Study of a Joint Information System Development Project. In *System Sciences, 2005. HICSS'05. Proceedings of the 38th Annual Hawaii International Conference on*, pp. 260a–260a. IEEE, 2005.

Ó Conchúir, E., Holmström Olsson, H., Ågerfalk, P. J. and Fitzgerald, B. Benefits of global software development: exploring the unexplored. *Software Process: Improvement and Practice*, 14(4):201–212, 2009.

Ohno, T. *Toyota Seisan Houshiki.[The Toyota Production System]*. Diamond, 1978.

Ohno, T. *Toyota production system: beyond large-scale production*. Productivity press, 1988.

Paasivaara, M. Coaching global software development projects. In *Global Software Engineering (ICGSE), 2011 6th IEEE International Conference on*, pp. 84–93. IEEE, 2011.

Paasivaara, M. and Lassenius, C. Could global software development benefit from agile methods? In *Global Software Engineering, 2006. ICGSE'06. International Conference on*, pp. 109–113. IEEE, 2006.

Paasivaara, M. and Lassenius, C. Agile coaching for global software development. *Journal of Software: Evolution and Process*, 26(4):404–418, 2014a.

Paasivaara, M. and Lassenius, C. Deepening Our Understanding of Communities of Practice in Large-Scale Agile Development. In *Agile Conference (AGILE), 2014*, pp. 37–40. IEEE, 2014b.

Paasivaara, M., Durasiewicz, S. and Lassenius, C. Using scrum in distributed agile development: A multiple case study. In *Global Software Engineering, 2009. ICGSE 2009. Fourth IEEE International Conference on*, pp. 195–204. IEEE, 2009.

Paasivaara, M., Heikkilä, V. T. and Lassenius, C. Experiences in scaling the product owner role in large-scale globally distributed scrum. In *Global Software Engineering (ICGSE), 2012 IEEE Seventh International Conference on*, pp. 174–178. IEEE, 2012.

Paasivaara, M., Lassenius, C., Damian, D., Raty, P. and Schroter, A. Teaching students global software engineering skills using distributed scrum. In *Software Engineering (ICSE), 2013 35th International Conference on*, pp. 1128–1137. IEEE, 2013a.

Paasivaara, M., Lassenius, C., Heikkila, V. T., Dikert, K. and Engblom, C. Integrating global sites into the lean and agile transformation at ericsson. In *Global Software Engineering (ICGSE), 2013 IEEE 8th International Conference on*, pp. 134–143. IEEE, 2013b.

Paasivaara, M., Behm, B., Lassenius, C. and Hallikainen, M. Towards rapid releases in large-scale xaas development at ericsson: A case study. In *Global Software Engineering (ICGSE), 2014 IEEE 9th International Conference on*, pp. 16–25. IEEE, 2014.

Patton, M. Q. *Qualitative Research and Evaluation Methods*. Sage Publications, Inc, 2002.

Pedreira, O., Piattini, M., Luaces, M. R. and Brisaboa, N. R. A systematic review of software process tailoring. *ACM SIGSOFT Software Engineering Notes*, 32(3):1–6, 2007.

Persson, J. S., Mathiassen, L. and Aaen, I. Agile distributed software development: enacting control through media and context. *Information Systems Journal*, 22(6):411–433, 2012.

Petersen, K. and Wohlin, C. Context in industrial software engineering research. In *Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement*, pp. 401–404. IEEE Computer Society, 2009.

Petersen, K., Feldt, R., Mujtaba, S. and Mattsson, M. Systematic mapping studies in software engineering. In *12th International Conference on Evaluation and Assessment in Software Engineering*, volume 17. sn, 2008.

Pichler, R. *Scrum. Agiles Projektmanagement erfolgreich einsetzen*. Heidelberg, 2008.

Poppendieck, M. and Poppendieck, T. *Lean software development: an agile toolkit*. Addison-Wesley Professional, 2003.

Power, K. Using Silent Grouping to Size User Stories. In *Agile Processes in Software Engineering and Extreme Programming: 12th International Conference, XP 2011, Madrid, Spain, May 10-13, 2011, Proceedings*, volume 77, p. 60. Springer Science & Business Media, 2011.

Pries-Heje, L. and Pries-Heje, J. Why Scrum works: A case study from an agile distributed project in Denmark and India. In *Agile Conference (AGILE), 2011*, pp. 20–28. IEEE, 2011.

Prikladnicki, R., Audy, J. L. N. and Evaristo, J. R. Distributed Software Development: Toward an Understanding of the Relationship Between Project Team, Users and Customers. In *ICEIS (3)*, pp. 417–423. Citeseer, 2003.

Ramesh, B., Cao, L., Mohan, K. and Xu, P. Can distributed software development be agile? *Communications of the ACM*, 49(10):41–46, 2006.

Ramesh, B., Mohan, K. and Cao, L. Ambidexterity in agile distributed development: an empirical investigation. *Information Systems Research*, 23(2):323–339, 2012.

Raza, B., MacDonell, S. G. and Clear, T. Research in global software engineering: a systematic snapshot. In *Evaluation of Novel Approaches to Software Engineering*, pp. 126–140. Springer, 2013.

Razavi, A. M. and Ahmad, R. Agile development in large and distributed environments: A systematic literature review on organizational, managerial and cultural aspects. In *Software Engineering Conference (MySEC), 2014 8th Malaysian*, pp. 216–221. IEEE, 2014.

Razzak, M. A. and Ahmed, R. Knowledge sharing in distributed agile projects: Techniques, strategies and challenges. In *Computer Science and Information Systems (FedCSIS), 2014 Federated Conference on*, pp. 1431–1440. IEEE, 2014.

Razzak, M. A. and Mite, D. Knowledge Management in Globally Distributed Agile Projects–Lesson Learned. In *Global Software Engineering (ICGSE), 2015 IEEE 10th International Conference on*, pp. 81–89. IEEE, 2015.

Rizvi, B., Bagheri, E. and Gasevic, D. A systematic review of distributed agile software engineering. *Journal of Software: Evolution and Process*, 27(10):723–762, 2015.

Robson, C. *Real world research: A resource for social scientists and practitioner-researchers*. Blackwell, 2nd edition, 1993.

Royce, W. W. Managing the development of large software systems. In *proceedings of IEEE WESCON*, volume 26. Los Angeles, 1970.

Runeson, P. and Höst, M. Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering*, 14(2):131–164, 2009.

Ryan, S. and O'Connor, R. Acquiring and sharing tacit knowledge in software development teams: An empirical study. *Information and Software Technology*, 55(9):1614–1624, 2013.

Sabherwal, R. The evolution of coordination in outsourced software development projects: a comparison of client and vendor perspectives. *Information and organization*, 13(3): 153–202, 2003.

Santos, P. S. M. d. and Travassos, G. H. Action research use in software engineering: An initial survey. In *Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement*, pp. 414–417. IEEE Computer Society, 2009.

Scharff, C. Guiding global software development projects using Scrum and Agile with quality assurance. In *Software Engineering Education and Training (CSEE&T), 2011 24th IEEE-CS Conference on*, pp. 274–283. IEEE, 2011.

Scharff, C. and Verma, R. Scrum to support mobile application development projects in a just-in-time learning context. In *Proceedings of the 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering*, pp. 25–31. ACM, 2010.

Scharff, C., Gotel, O. and Kul, V. Transitioning to Distributed Development in Students' Global Software Development Projects: The Role of Agile Methodologies and End-to-End Tooling. In *Software Engineering Advances (ICSEA), 2010 Fifth International Conference on*, pp. 388–394. IEEE, 2010.

Schatten, A., Biffl, S., Demolsky, M., Gostischa-Franta, E., Östreicher, T. and Winkler, D. *Best Practice Software-Engineering*. Springer, 2010.

Schenk, J., Prechelt, L. and Salinger, S. Distributed-Pair Programming can work well and is not just Distributed Pair-Programming. In *Companion Proceedings of the 36th International Conference on Software Engineering*, pp. 74–83. ACM, 2014.

Schnabel, I. and Pizka, M. Goal-driven software development. In *Software Engineering Workshop, 2006. SEW'06. 30th Annual IEEE/NASA*, pp. 59–65. IEEE, 2006.

Schnitter, J. and Mackert, O. Large-scale agile software development at SAP AG. In *Evaluation of Novel Approaches to Software Engineering*, pp. 209–220. Springer, 2011.

Schümmer, T. and Lukosch, S. Understanding Tools and Practices for Distributed Pair Programming. *J. UCS*, 15(16):3101–3125, 2010.

Schwaber, K. Scrum development process. In *Business Object Design and Implementation*, pp. 117–134. Springer, 1997.

Schwaber, K. *Agile project management with Scrum*, volume 7. Microsoft press Redmond, 2004.

Schwaber, K. *The enterprise and scrum*, volume 1. Microsoft Press Redmond, 2007.

Schwaber, K. and Beedle, M. *Agile Software Development with Scrum*. Prentice Hall PTR, 2001.

Sengupta, B., Chandra, S. and Sinha, V. A research agenda for distributed software development. In *Proceedings of the 28th international conference on Software engineering*, pp. 731–740. ACM, 2006.

Sharp, H., Giuffrida, R. and Melnik, G. Information Flow within a Dispersed Agile Team: A Distributed Cognition Perspective. In *Agile Processes in Software Engineering and Extreme Programming: 13th International Conference, XP 2012, Malmö, Sweden, May 21-25, 2012, Proceedings*, volume 111, pp. 62–76. Springer, 2012.

Shrivastava, S. V. and Rathod, U. Categorization of risk factors for distributed agile projects. *Information and Software Technology*, In-Print 58:373–387, 2015, Online–First, 2014.

Sindhgatta, R., Sengupta, B. and Datta, S. Coping with distance: an empirical study of communication on the jazz platform. In *Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion*, pp. 155–162. ACM, 2011.

Sison, R. and Yang, T. Use of Agile Methods and Practices in the Philippines. In *Software Engineering Conference, 2007. APSEC 2007. 14th Asia-Pacific*, pp. 462–469. IEEE, 2007.

Sjøberg, D. I., Dybå, T. and Jørgensen, M. The future of empirical methods in software engineering research. In *2007 Future of Software Engineering*, pp. 358–378. IEEE Computer Society, 2007.

Šmite, D., Wohlin, C., Feldt, R. and Gorschek, T. Reporting empirical research in global software engineering: A classification scheme. In *Global Software Engineering, 2008. ICGSE 2008. IEEE International Conference on*, pp. 173–181. IEEE, 2008.

Šmite, D., Moe, N. B. and Ågerfalk, P. J. Agility across time and space: summing up and planning for the future. In *Agility Across Time and Space*, pp. 333–337. Springer, 2010a.

Šmite, D., Moe, N. B. and Ågerfalk, P. J. Fundamentals of agile distributed software development. In *Agility Across Time and Space*, pp. 3–7. Springer, 2010b.

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

Šmite, D., Wohlin, C., Galviņa, Z. and Prikladnicki, R. An empirically based terminology and taxonomy for global software engineering. *Empirical Software Engineering*, 19(1): 105–153, 2014.

Sohan, S., Richter, M. M. and Maurer, F. Auto-tagging Emails with User Stories Using Project Context. In *Agile Processes in Software Engineering and Extreme Programming: 11th International Conference, XP 2010, Trondheim, Norway, June 1-4, 2010, Proceedings*, volume 48, pp. 103–116. Springer, 2010.

Sorathia, V., van Sinderen, M. and Pires, L. F. Towards a Unifying Process Framework for Services Knowledge Management. In *Exploring Services Science*, pp. 295–299. Springer, 2010.

Soundararajan, S., Arthur, J. D. and Balci, O. A Methodology for Assessing Agile Software Development Methods. In *Agile Conference (AGILE), 2012*, pp. 51–54. IEEE, 2012.

Srinivasan, J. and Lundqvist, K. Agile in India: Challenges and lessons learned. In *Proceedings of the 3rd India software engineering conference*, pp. 125–130. ACM, 2010.

Sriram, R. and Mathew, S. Global software development using agile methodologies: A review of literature. In *Management of Innovation and Technology (ICMIT), 2012 IEEE International Conference on*, pp. 389–393. IEEE, 2012.

Stake, R. E. *The art of case study research*. Sage, 1995.

Stankovic, D., Nikolic, V., Djordjevic, M. and Cao, D.-B. A survey study of critical success factors in agile software projects in former Yugoslavia IT companies. *Journal of Systems and Software*, 86(6):1663–1678, 2013.

Stapel, K., Knauss, E., Schneider, K. and Zazworka, N. FLOW mapping: planning and managing communication in distributed teams. In *Global Software Engineering (ICGSE), 2011 6th IEEE International Conference on*, pp. 190–199. IEEE, 2011.

Stapleton, J. *DSDM, dynamic systems development method: the method in practice*. Cambridge University Press, 1997.

Strauss, A. L., Corbin, J. M. et al. *Basics of qualitative research*. Sage, 1998.

Sundararajan, S., Bhasi, M. and Vijayaraghavan, P. K. Case study on risk management practice in large offshore-outsourced Agile software projects. *IET Software*, 8(6):245–257, 2014.

Sungkur, R. K. and Ramasawmy, M. Knowledge4Scrum, a novel knowledge management tool for agile distributed teams. *VINE*, 44(3):394–419, 2014.

Sutherland, J. Business object design and implementation workshop. *ACM SIGPLAN OOPS Messenger*, 6(4):170–175, 1995.

Sutherland, J. and Schwaber, K. *The scrum papers: Nuts, bolts, and origins of an agile process*. Scruminc., 2007.

Sutherland, J., Viktorov, A., Blount, J. and Puntikov, N. Distributed scrum: Agile project management with outsourced development teams. In *System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on*, pp. 274a–274a. IEEE, 2007.

Szőke, Á. Optimized Feature Distribution in Distributed Agile Environments. In *Product-Focused Software Process Improvement: 11th International Conference, PROFES 2010, Limerick, Ireland, June 21-23, 2010, Proceedings*, volume 6156, pp. 62–76. Springer, 2010.

Szőke, Á. A Feature Partitioning Method for Distributed Agile Release Planning. In *Agile Processes in Software Engineering and Extreme Programming: 12th International Conference, XP 2011, Madrid, Spain, May 10-13, 2011, Proceedings*, volume 77, pp. 27–42. Springer, 2011.

Tahir, F. and Manarvi, I. A. Agile Process Model and Practices in Distributed Environment. In *Concurrent Engineering Approaches for Sustainable Product Development in a Multi-Disciplinary Environment*, pp. 1169–1180. Springer, 2013.

Takeuchi, H. and Nonaka, I. The new new product development game. *Harvard business review*, 64(1):137–146, 1986.

Teiniker, E., Paar, S. and Lind, R. A practical software engineering course with distributed teams. In *Interactive Collaborative Learning (ICL), 2011 14th International Conference on*, pp. 195–201. IEEE, 2011.

Thomas, D. Web time software development. *Software Development*, 6(10):80, 1998.

Tripathi, N., Rodríguez, P., Ahmad, M. O. and Oivo, M. Scaling Kanban for Software Development in a Multisite Organization: Challenges and Potential Solutions. In *Agile Processes, in Software Engineering, and Extreme Programming: 16th International Conference, XP 2015, Helsinki, Finland, May 25-29, 2015, Proceedings*, volume 212, p. 178. Springer, 2015.

Vallon, R. Lean and Agile Software Development: Planung und Realisierung einer Verbindung von Kanban und Scrum. Master's thesis, Vienna University of Technology, 2011.

Vallon, R. Evaluation of Lean-Agile Multi-Project Management in a Medium-sized Development Environment. Master's thesis, Vienna University of Technology, 2012.

Vallon, R. Empirically Driven Design of the Agile Distributed Adaptable Process Toolkit (ADAPT). Technical report, Austrian Marshall Plan, 2015.

Vallon, R. and Grechenig, T. Ten Heuristics from Applying Agile Practices across Different Distribution Scenarios: A Multiple-Case Study. *Computer and Information Science*, 9 (2):Online–First, May 2016.

Vallon, R., Müller-Wernhart, M., Schramm, W. and Grechenig, T. Kombination von Agil und Lean in der Softwareentwicklung. *Springer Informatik-Spektrum*, In-Print 37(1):28–35, 2014, Online–First, 2012.

Vallon, R., Bayrhammer, K., Strobl, S., Bernhart, M. and Grechenig, T. Identifying Critical Areas for Improvement in Agile Multi-site Co-development. In *8th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE)*, pp. 165–172. SciTePress, 2013a.

Vallon, R., Strobl, S., Bernhart, M. and Grechenig, T. Inter-organizational Co-development with Scrum: Experiences and Lessons Learned from a Distributed Corporate Development Environment. In *Agile Processes in Software Engineering and Extreme Programming. 14th International Conference, XP 2013, Vienna, Austria, June 3-7, 2013. Proceedings.*, volume 149 of *Lecture Notes in Business Information Processing*, pp. 150–164. Springer Berlin Heidelberg, 2013b.

Vallon, R., Dräger, C., Zapletal, A. and Grechenig, T. Adapting to Changes in a Project's DNA: A Descriptive Case Study on the Effects of Transforming Agile Single-Site to Distributed Software Development. In *Agile Conference (AGILE), 2014*, pp. 52–60. IEEE, 2014.

Vallon, R., Wenzel, L., Brüggemann, M. E. and Grechenig, T. An Agile and Lean Process Model for Mobile App Development: Case Study into Austrian Industry. *Journal of Software*, 10(11):1245–1264, 2015.

Van Aken, J. E. Management research as a design science: articulating the research products of mode 2 knowledge production in management. *British journal of management*, 16(1): 19–36, 2005.

van Hillegersberg, J., Ligtenberg, G. and Aydin, M. N. Getting Agile Methods to Work for Cordys Global Software Product Development. In *New Studies in Global IT and Business Services Outsourcing: 5th Global Sourcing Workshop 2011, Courchevel, France, March 14-17, 2011, Revised Selected Papers*, volume 91, p. 133. Springer, 2011.

VanderLeest, S. H. and Buter, A. Escape the waterfall: Agile for aerospace. In *Digital Avionics Systems Conference, 2009. DASC'09. IEEE/AIAA 28th*, pp. 6–D. IEEE, 2009.

Verner, J., Brereton, O., Kitchenham, B., Turner, M. and Niazi, M. Systematic literature reviews in global software development: A tertiary study. In *Evaluation & Assessment in Software Engineering (EASE 2012), 16th International Conference on*, pp. 2–11. IET, 2012.

Verner, J. M., Sampson, J., Tosic, V., Bakar, N. A. A. and Kitchenham, B. A. Guidelines for industrially-based multiple case studies in software engineering. In *Research Challenges in Information Science, 2009. RCIS 2009. Third International Conference on*, pp. 313–324. IEEE, 2009.

VersionOne, . 9th Annual State of Agile Survey, 2014. URL `http://stateofagile.versionone.com/`.

Viswanath, U. Lean Transformation: How Lean Helped to Achieve Quality, Cost and Schedule: Case Study in a Multi Location Product Development Team. In *Global Software Engineering (ICGSE), 2014 IEEE 9th International Conference on*, pp. 95–99. IEEE, 2014.

Vriens, C. Certifying for CMM Level 2 and IS09001 with XP@ Scrum. In *Agile Development Conference, 2003. ADC 2003. Proceedings of the*, pp. 120–124. IEEE, 2003.

Wang, X., Conboy, K. and Cawley, O. Leagile software development: An experience report analysis of the application of lean approaches in agile software development. *Journal of Systems and Software*, 85(6):1287–1299, 2012.

Wieringa, R., Maiden, N., Mead, N. and Rolland, C. Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. *Requirements Engineering*, 11(1):102–107, 2006.

Wild, P., Clarkson, P. and McFarlane, D. A framework for cross disciplinary efforts in services research. In *Proceedings of the 19th CIRP Design Conference–Competitive Design*. Cranfield University Press, 2009.

Winkler, D., Biffl, S. and Kaltenbach, A. Evaluating tools that support pair programming in a distributed engineering environment. In *Proceedings of the 14th international conference on Evaluation and Assessment in Software Engineering*, pp. 54–63. British Computer Society, 2010.

Wirdemann, R. *Scrum mit User Stories*. Hanser, 2011.

Womack, J. P. and Jones, D. T. *Lean thinking: banish waste and create wealth in your corporation*. Productivity Press, 1996.

Wood, J. and Silver, D. *Joint application development*. John Wiley & Sons, Inc., 1995.

Woodward, E., Surdek, S. and Ganis, M. *A practical guide to distributed Scrum*. Pearson Education, 2010.

Xie, M., Shen, M., Rong, G. and Shao, D. Empirical studies of embedded software development using agile methods: a systematic review. In *Proceedings of the 2nd international workshop on Evidential assessment of software technologies*, pp. 21–26. ACM, 2012.

Xu, P. and Ramesh, B. A tool for the capture and use of process knowledge in process tailoring. In *System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on*. IEEE, 2003.

Yin, M. and Ma, J. A review of the agile and geographically distributed software development. In *Information Technology and Computer Application Engineering: Proceedings of the International Conference on Information Technology and Computer Application Engineering (ITCAE 2013)*, p. 173. CRC Press, 2013.

Yin, R. *Case study research*. Sage Publications, 2003.

Yoshii, A. and Higa, K. Analysis of the peculiarity of the japanese software development style in offshore software development. *IEEJ Transactions on Electrical and Electronic Engineering*, 6(1):46–50, 2011.

Zieris, F. and Salinger, S. Doing Scrum Rather Than Being Agile: A Case Study on Actual Nearshoring Practices. In *Global Software Engineering (ICGSE), 2013 IEEE 8th International Conference on*, pp. 144–153. IEEE, 2013.

APPENDIX A

Appendix

A.1 Glossary

ADAPT	Agile Distributed Adaptable Process Toolkit
AGILE	The Agile Conference (conference series)
AR	Action Research
BDD	Behavior Driven Development
CCC	Coordination, Control, Communication (CCC model)
CFD	Cumulative Flow Diagram
Cx	C1, ..., C7 abbreviating the 7 conceptual practices of ADAPT
Dev	Development
DoD	Definition of Done
DSD	Distributed Software Development
EUC	EU Country (country within the European Union)
GSD	Global Software Development
Gx	G1, ..., G10 abbreviating the 10 guidelines of ADAPT
ICGSE	International Conference on Global Software Engineering (conference series)
IS	Information Science
IST	Information and Software Technology (journal)
JoS: EP	Journal of Software: Evolution and Process (journal)
JSS	Journal of Systems and Software (journal)
PI	Principal Investigator
PMO	Project Management Office

Px	P1, ..., P29 abbreviating the 29 full practices of ADAPT
PO	Product Owner
QA	Quality Assurance
RQ	Research Question
SE	Software Engineering
SoS	Scrum of Scrums
SM	Scrum Master
SLR	Systematic Literature Review
SWEBOK	Software Engineering Body of Knowledge
TDD	Test Driven Development
TP	Testable Proposition
WiP	Work in Progress
XP	Extreme Programming *or* International Conference on Agile Software Development (conference series)

A.2 Final Set of 95 Included Studies in Systematic Mapping

1. Akbar and Hassan (2010)
2. Akbar et al. (2011b)
3. Akbar et al. (2011c)
4. Akbar et al. (2011a)
5. Akbar et al. (2012)
6. Almeida et al. (2011)
7. Alyahya et al. (2013)
8. Alsmadi and Saeed (2013)
9. Ansari and Ansari (2012)
10. Ansari et al. (2010)
11. Ashraf et al. (2012)
12. Badampudi et al. (2013)
13. Bandukda and Nasir (2010)
14. Bass (2012)
15. Bass (2014)
16. Batra et al. (2010)
17. Belsis et al. (2014)

18. Bocock and Martin (2011)
19. Ceria and Pallotti (2010)
20. Cocco et al. (2012)
21. Daneva and Ahituv (2012)
22. Daneva et al. (2013)
23. Dorairaj et al. (2010)
24. Dorairaj et al. (2011)
25. Dorairaj et al. (2012a)
26. Dorairaj et al. (2013)
27. Dumitriu et al. (2011)
28. Estler et al. (2014)
29. da Silva Estácio and Prikladnicki (2014)
30. Femmer et al. (2014)
31. Green et al. (2010a)
32. Green et al. (2010b)
33. Hallikainen (2011)
34. Hamid (2013)
35. Hong et al. (2010)
36. Hossain et al. (2011a)
37. Hossain et al. (2011b)
38. Inayat et al. (2012)
39. Jalali and Wohlin (2010)
40. Jalali and Wohlin (2012a)
41. Jalali and Wohlin (2012b)
42. Fernando et al. (2011)
43. Kamaruddin et al. (2012)
44. Kajko-Mattsson et al. (2010)
45. Kanwal et al. (2014)
46. Khan et al. (2010)
47. Klein et al. (2012)
48. Korkala et al. (2010)
49. Korhonen (2010)
50. Kuhrmann et al. (2013)
51. Lavazza et al. (2010)

52. Lee et al. (2011)
53. Lehtinen et al. (2014)
54. Licorish and MacDonell (2013)
55. Maher et al. (2010)
56. Meyer et al. (2010)
57. Modi et al. (2013)
58. Mudumba and Lee (2010)
59. Nagle et al. (2011)
60. Nevo and Chengalur-Smith (2011)
61. Noordeloos et al. (2012)
62. del Nuevo et al. (2011)
63. Paasivaara et al. (2012)
64. Paasivaara et al. (2013a)
65. Paasivaara et al. (2013b)
66. Paasivaara et al. (2014)
67. Paasivaara and Lassenius (2014a)
68. Paasivaara and Lassenius (2014b)
69. Persson et al. (2012)
70. Power (2011)
71. Pries-Heje and Pries-Heje (2011)
72. Ramesh et al. (2012)
73. Ryan and O'Connor (2013)
74. Schümmer and Lukosch (2010)
75. Scharff et al. (2010)
76. Scharff (2011)
77. Schenk et al. (2014)
78. Sharp et al. (2012)
79. Shrivastava and Rathod (2014)
80. Sindhgatta et al. (2011)
81. Sohan et al. (2010)
82. Srinivasan and Lundqvist (2010)
83. Sriram and Mathew (2012)
84. Stapel et al. (2011)
85. Stankovic et al. (2013)

86. Szőke (2010)
87. Szőke (2011)
88. Tahir and Manarvi (2013)
89. Teiniker et al. (2011)
90. Vallon et al. (2014)
91. Vallon et al. (2013b)
92. Winkler et al. (2010)
93. Yin and Ma (2013)
94. Yoshii and Higa (2011)
95. Zieris and Salinger (2013)

A.3 Proposed Data Extraction Checklist for Reporting Empirical Studies on Agile Distributed Software Development

General:
Include (yes, no, maybe)
Exclusion comment
Identifier
Title
Databases (comma-separated)
Authors' Names (comma-separated)
Authors' Affiliations (comma-separated: university/R&D or "INDUSTRY")
Authors' Countries (comma-separated)
Year
Target (conference or journal name)
Time stamp of Data Extraction and Researcher's Name

Research:
Type (solution, validation, evaluation, philosophical, experience, opinion)
Method (qualitative, quantitative, mixed)
Sub-Method (single-case study, multiple-case study, interviews, ...)
Means of Analysis (grounded theory, statistical, ...)

Empirical:
Empirical (yes, no, unclear)
Project Size (small, medium, large, unclear)
Project Duration (short, medium, large, unclear)
Participants (industry, students, unclear)
Knowledge Area (requirement, design, construction, testing, SE management, SE process, maintenance, tools & methods)
Application Domain (e.g. enterprise software, ...)
Successful (yes, no, unclear)

Distribution:
Global Development (yes, no, unclear)
Location (offshore, onshore, unclear)
Legal entity (outsourcing, insourcing, unclear)
Geographical Distance (far, near, unclear)
Temporal Distance (large, small, unclear)
Team Distribution Type (integrated, isolated, unclear)
Number of sites (0 for unclear, otherwise ≥ 1)
Supplier Countries (comma-separated)
Customer Countries (comma-separated)

Agile:
Agile Main Practice (Scrum, XP, Lean, ..., unclear)
Agility Level (not all teams, all teams, organization, unclear)
Working Agile Sub-practices (comma-separated)
Not working Agile Sub-practices (comma-separated)

Result:
Contributions of the Study (lessons learned, tool development, framework, model, ...)
Summary Comment by Researcher

A.4 Semi-structured Interview Guide for Evaluation Interviews

1. Background:
   - What is your current position (in research / in practice)?
   - How many years experience with agile collocated (in research / in practice)?
   - How many years experience with agile distributed (in research / in practice)?
   - How many years of other relevant experience?
2. *Short presentation to give an overview of the ADAPT framework's problem statement, motivation, methodology and results.*
3. Have you done Case Study research? What do you think of the methodology? What would you have done differently?
4. Have you done Action Research? What do you think of the methodology? What would you have done differently?
5. Have you conducted a systematic mapping/literature review before? What do you think of the methodology? What would you have done differently?
6. What is your opinion on the design criteria? What would you have done differently?
7. What is your opinion on the best practice? How could it be made for useful to a) the practitioner? b) the research community?

8. How would you see the a) practical b) research value of this framework? What could be next steps/improvements?
9. What is missing in the framework, based on your experience, either from a a) practical or b) research point of view?
10. What should be considered for future work? Your thoughts on:
    a) Implementation study
    b) Scaling mechanisms for the framework (integrating new cases)
    c) Practice cards
    d) Database/Website
    e) Pattern language
    f) What else comes to mind?
11. Your overall evaluation, positive or negative? Any other final comments?

Printouts provided during interview:

- ADAPT Testable Propositions TP1-TP5
- ADAPT v1.0 Compact Summary Table
- ADAPT Practice Cards Example

A.5 Tools Used

This thesis has been created using the following tools:

Main OS: Ubuntu 14.04 (former 12.04)

- LaTeX: pdfTeX 3.1415926-2.5-1.40.14 TeX Live 2013/Debian (Typesetting)
- Kile 2.1.3 (LaTeX editor)
- JabRef 2.9.2 (Reference manager/BibTeX editor)
- GNOME-Terminal 3.6.2 (orchestrating LaTeX builds)
- Meld 1.6.0 (useful for merging back and forth between publications and thesis)
- Google Web Forms (used for data extraction for systematic mapping study)
- Google Scholar (used as primary source for extracting BibTeX references)
- Evince 3.10.3 (fast and light-weight PDF reader)
- Adobe Reader 9.5.5 (most widely used PDF reader, for compatibility checking)
- Oracle Java 6: 1.6.0_45 (small program used for data extraction assistance in merging the csv files to one main file in systematic mapping study and getting rid of duplicate entries)
- LibreOffice 4.2.8.2 (spreadsheet analysis)
- GIMP 2.8.10 (image processing)

- http://ericwood.org/excel2latex/ (turn spreadsheet tables into LaTeX table stubs)
- http://text2mindmap.com (for mindmap-style figures of guidelines and practices in single-case analysis)

Side OS: MS Windows 7

- ATLAS.ti 7.5.2 (qualitative data analysis software)
- SmartDraw CI 22.0.0.3 (schematic diagrams and graphs)
- MS Powerpoint 2007 (schematic diagrams and graphs)
- MS Excel 2007 (supporting multi-case study and systematic mapping)

A.6 Curriculum Vitae

See following page.

CURRICULUM VITAE
DDIPL.-ING. RAOUL VALLON, BSC

Born in Vienna, 09-05-1986, Austrian
Hernalser Hauptstr. 14/9, A-1170 Vienna
raoul.vallon@gmail.com

EDUCATION

2013 – 2016 (expected) **PhD in Computer Science, Vienna University of Technology**
- Specializing in Agile and Distributed Software Development
- Thesis title: "Towards a Light-weight Distributed Software Development Process: Empirically Driven Design of the Agile Distributed Adaptable Process Toolkit (ADAPT)"
- Supervisor: Prof. Thomas Grechenig

08/2014 – 11/2014 **Research Leave, Center for Design Research, Stanford University, CA**
- Visiting researcher at the designX lab, hosted by Prof. Larry Leifer
- Working on the the design theory behind the ADAPT framework
- Austrian Marshall Plan scholar

10/2009 – 03/2012 **MSc. (with Honors) Software Engineering & Internet Computing, Vienna University of Technology**
- Specialized in Internet Technologies and Distributed Systems
- Thesis title: „Evaluation of Lean-Agile Multi-Project Management in a Medium-sized Development Environment"
- Supervisors: Prof. Thomas Grechenig and Michael Müller-Wernhart

09/2010 – 01/2011 **Erasmus Exchange, Universidad Politécnica de Madrid (UPM)**
- Successfully completed several courses of the European Master in Software Engineering (EMSE) program in English and Spanish
- Successfully completed two advanced Spanish courses (B1)

10/2008 – 06/2011 **MSc. (with Honors) Business Informatics, Vienna University of Technology**
- Specialized in Internet Computing
- Thesis title: "Lean and Agile Software Development: Conception and Realization of a Combination of Kanban and Scrum"
- Supervisors: Prof. Thomas Grechenig and Michael Müller-Wernhart

10/2005 – 10/2008 **BSc. Business Informatics, Vienna University of Technology**
- Specialized in Practical Software Engineering
- Thesis title: "Conformity Testing of Interactive Systems"
- Supervisors: Prof. Thomas Grechenig and Thomas Költringer

10/2004 – 06/2005 **Military Service, Fire Brigade, Military Airport Brumowski, Lower Austria**
Firefighter, promoted to "Gefreiter" (Private)

09/1996 – 06/2004 **A-Levels (with Honors), Billrothgymnasium Grg19, 1190 Vienna**
- Emphasis on languages German, English, Latin and French
- Specialized in Informatics and English

RELEVANT EMPLOYMENT

2012 – present **Research Fellow, Research Group for Industrial Software, Vienna University of Technology**

- Operational head of work group AMMA focusing on empirical software engineering and agile/lean software development processes
- Co-supervisor of several master and bachelor theses

03/2007 – 01/2012 **Teaching Assistant, Vienna University of Technology**

- Internet Security at Information & Software Engineering Group
- Database Systems and Semi-Structured Data at Artificial Intelligence Group
- Interaction Design & Usability Engineering at Research Group for Industrial Software

OTHER EMPLOYMENT

2012 – present **Senior Software Engineer and Agile Coach in various projects**

2008 – 2010 **Software Backend Developer and Software Tester in various projects**