Universität Stuttgart

Leadership Gap in Agile Teams: How Developers and Scrum Masters Mature

Von der Fakultät für Informatik, Elektrotechnik und Informationstechnik der Universität Stuttgart zur Erlangung der Würde eines Doktors der Naturwissenschaften (Dr. rer. nat.) genehmigte Abhandlung

Vorgelegt von
Simone V. Spiegler
geboren in Bad Nauheim, Deutschland

Hauptberichter:	Prof. Dr. Stefan Wagner
Mitberichter:	Prof. Dr. Kurt Schneider
	Prof. Dr. Michael Sedlmair

Tag der mündlichen Prüfung: Dienstag, 9.11.2021

Institut für Software Engineering der Universität Stuttgart

2021

Abstract

An increasing number of companies aim to enable their developers to work in an agile manner. One key success factor that supports teams in working in an agile way is fitting leadership. Therefore, companies try to define leadership in such self-organising teams. One agile leadership concept describes a Scrum Master who is supposed to empower the team to work in an agile manner. Yet, findings on such a leadership role are controversial. We still have not understood how leadership unfolds in a team that is by definition self-organising.

A few empirical studies suggest that leadership evolves while the team matures and that parts of the Scrum Master role, which implies leadership activities, are transferred to the Developers. Yet, this claim has never been empirically tested.

This thesis aims to understand how leadership evolves while taking maturity and organisational culture and structure into account.

By using Grounded Theory I continuously compare emerging data to existing concepts from research on team leadership, role theory, maturity and contextual factors, as well as to research on leadership in agile teams. As a result, I develop new theories on leadership in agile teams.

Observation and qualitative interviews with 53 practitioners of 29 Scrum teams from 11 divisions at the Robert Bosch GmbH identified a set of *nine*

leadership roles that are transferred from the Scrum Master to the Development Team while it matures.

The *role transfer process* is context-dependent. A *leadership gap* and a supportive *internal team environment* encourage the Developers in taking on leadership roles, while expectations deriving from a bureaucratic culture and structure diminish the transfer of leadership roles. Based on these results I suggest the *Agile Matching Theory* which implies that organisational context and internal team environment need to match agile features for the role transfer to occur.

Moreover, I build on my results and examine the presence and sharing of the nine leadership roles, named 9-Factor Theory. I conduct a quantitative survey with 67 participants containing 37 Scrum Masters and 30 Developers.

Descriptive statistics reveal that the Scrum Master and the Development Team score differently on the 9 factors and that leadership is most often distributed in teams that had been working between 3 and 5 months in an agile manner. Yet, I also find that the leadership roles predominantly remain with one dedicated Scrum Master. I claim that one committed leader does not become obsolete over time but is played to a lesser extend.

Based on my results I suggest to group the 9-Factor Theory into three clusters: the Scrum Master is rather linked to *psychological team factors* (1), while the Development Team tends to be linked to *product-related factors* (2). *Organisational factors* (3) are less often present.

I suggest that it depends on the context if Developers start taking on leadership roles. Teams learn how to take on leadership roles over time. A regular Retrospective helps in building a shared understanding on sharing of leadership roles. Role sharing between the Developers and the dedicated Scrum Master is based on continuously finding an equilibrium between agile team features and maturity, culture and structure. Dedicated leaders should provide a leadership gap by stepping back, and trust and freedom while taking the context into account. Moreover, the organisational context in rather bureaucratic environment should gradually be changed towards a better fit with agile features on team level.

My study does not only provide more theoretical underpinning to human aspects of the agile manner but also builds groundwork for future quantitative testing of leadership in agile teams.

Future studies should build on our theories and draw data from a larger sample size.

Zusammenfassung

Immer mehr Unternehmen streben danach, dass Entwickler agil zusammenarbeiten. Ein Erfolgsfaktor, der Teams zu einer agilen Arbeitsweise befähigt, ist eine dazu passende Art von Führung. Deshalb versuchen Unternehmen herauszufinden, was Führung in selbst-organisierten agilen Teams bedeutet. Ein Führungskonzept ist im Scrum Master verkörpert. Diese Rolle hat das Ziel, ein Team zum agilen Arbeiten zu befähigen. Die bisherigen Erkenntnisse hinsichtlich dieser Führungsrolle sind jedoch widersprüchlich. Es ist noch immer unklar, wie sich Führung in selbst-organisierten Teams entfaltet.

Eine kleine Anzahl an empirischen Studien suggeriert, dass sich Führung in Abhängigkeit vom Reifegrad eines Teams verändert. Es wird vermutet, dass die Scrum Master-Rolle, und somit Führung, teilweise auf die Entwickler übertragen wird. Diese Behauptung ist jedoch bisher noch nicht empirisch untersucht worden.

Diese Doktorarbeit erforscht, wie sich Führung in Zusammenhang mit der Reife eines Teams sowie der Organisationskultur und -struktur entfaltet.

Durch die Anwendung von Grounded Theory erheben und analysieren wir qualitative Daten in mehreren Iterationsschleifen. Dabei verknüpfen wir wissenschaftliche Erkenntnisse der Teamführung, Rollentheorie, des Reifegrads und kontextuellen Faktoren sowie der Führungsforschung in agilen Teams. Ein Ergebnis dieser Arbeit sind neue Theorien bezüglich der

Führung in agilen Teams.

Durch qualitative Interviews und Beobachtungen von 53 Individuen von 29 Scrum Teams von 11 Geschäftsbereichen der Robert Bosch GmbH haben wir *neun Führungsrollen* identifiziert. Diese werden von der Scrum Master-Rolle auf die Entwickler übertragen, während sie sich gemeinsam weiterentwickeln.

Des Weiteren hat die Datenanalyse ergeben, dass der *Transferprozess* der Rollen kontextabhängig ist. Eine *Führungslücke* und ein unterstützendes *Teamklima* ermutigen das Team dazu, selbst Führungsrollen zu übernehmen. Wohingegen Erwartungen, die auf Grund einer bürokratischen Kultur und Struktur entstehen, diese Übertragung von Führungsrollen auf das Team reduzieren. Auf Grund von diesen Erkenntnissen begründen wir die *Theorie der Agilen Passung*, die impliziert, dass der Organisationskontext mit einem agilen Teamklima in Einklang sein muss, damit der Rollentransfer stattfindet.

Basierend auf diesen Ergebnissen, haben wir die Ausprägung und das Teilen der neun Führungsrollen (9-Faktoren-Theorie) in einer quantitativen Umfrage untersucht. An dieser haben 67 Teilnehmer, davon 37 Scrum Master und 30 Entwickler, teilgenommen.

Deskriptive Statistik hat ergeben, dass Scrum Master und Entwickler in den neun Faktoren unterschiedlich abschneiden. Am häufigsten werden Führungsrollen in Teams geteilt, die zwischen drei und fünf Monaten in einer agilen Arbeitsweise zusammengearbeitet haben. Die Ergebnisse haben auch gezeigt, dass die Führungsrollen überwiegend von der offiziellen Scrum Master-Rolle gespielt werden. Unser Fazit ist daher, dass eine festgelegte Führungsrolle im Lauf der Zeit nicht überflüssig wird, sich jedoch im Zeitverlauf verändert.

Auf Grund unserer Ergebnissen schlagen wir vor, die 9-Faktoren-Theorie in drei Cluster aufzuteilen: der Scrum Master-Rolle werden eher *psychologische Teamfaktoren* zugeschrieben (1), während das Team eher auf *produktbezogene Faktoren* gemünzt ist (2). Faktoren, die sich auf die *Organisation* beziehen (3), sind seltener in einem Team vorhanden.

Wir vermuten daher, dass der Kontext beeinflusst, ob ein Team dazu bereit ist, Führungsrollen zu übernehmen. Eine regelmäßige *Retrospektive* unterstützt das Team darin, ein gemeinsames Verständnis zum Teilen von Führungsrollen zu entwickeln. Das Teilen von Führungsrollen zwischen den Entwicklern und dem Scrum Master hängt von einem *Gleichgewicht* zwischen agilen Teameigenschaften, dem Reifegrad, der Kultur und der Struktur ab. Dieses Gleichgewicht wird kontinuierlich neu verhandelt. Führungskräfte sollten eine Führungslücke bereitstellen, sowie dem Team Vertrauen und Freiheit schenken, während sie den entsprechenden Kontext in Bedracht ziehen. Darüber hinaus sollte sich der Organisationskontext in einem eher bürokratischen Umfeld Schritt für Schritt an agile Teameigenschaften annähern.

Unsere Studie trägt nicht nur zu einem tieferen Verständnis bezüglich der Zusammenarbeit in einem agilen Setting bei, sondern es bildet auch ein theoretisches Fundament für weitere quantitative Führungsforschung in agilen Teams.

Zukünftige Forschungsprojekte sollten auf unseren Theorien aufbauen und Daten eines größeren Samples erheben.

Acknowledgements

This thesis was a cooperation between the Robert Bosch Automotive Steering GmbH, Schwäbisch Gmünd, Germany, and Prof. Dr. Stefan Wagner at the Institute of Software Engineering, University of Stuttgart, Germany.

The past 3 years have been an enriching journey that empowered me to grow personally. Tremendous support and inspiring discussions guided me on my way. This thesis would not have been possible without the support by my friends and supervisors, individuals from Bosch, and the research community. I am truly thankful for the encouragement.

First of all, I would like to thank the Robert Bosch Automotive Steering GmbH in Germany for funding this project. Particularly my supervisor Dr. Christoph Heinecke for believing in my skills, for true commitment and trust, for being more of a mentor and friend than a supervisor, for all the talks, jokes, feedback and advice.

Moreover, I would like to thank Prof. Dr. Stefan Wagner at the Institute of Software Engineering, University of Stuttgart, Germany, for academically supporting this project. Thank you for providing trust, freedom and space to create my own thoughts, but also for asking the right questions at the right time and for structuring my thoughts when I felt lost.

I would like to thank all my friends for their support. Particularly, I would like to thank Caro, Fred and Xenia for proof reading parts of this thesis and some of the papers, for challenging my thoughts, for asking the right questions, for helping me structure my thoughts and for enjoying life with me. For being there for me in my low lights and highlights during the past years. For helping me grow personally.

Moreover, I would like to thank Thorsten for helping me grow, for fruitful discussions and for reading parts of this thesis and providing feedback.

This thesis was supported by the Robert Bosch Automotive Steering GmbH. I would like to thank the final thesis students Kevin, Tom and Verena for transcribing numerous qualitative interviews, asking challenging questions and igniting inspiring discussions on self-organisation. My disciplinary supervisors Stefan and Dirk for supporting this project. Vera for interesting discussions and providing excellent HR related support. My local department for welcoming and integrating me.

This thesis was extremely supported by the Agile Community all over the Robert Bosch GmbH. Numerous experts on the agile way of working welcomed me for knowledge exchange meetings. Particularly, I would like to thank the Agile Transformers, the Scrum Master community and the Agile Lunch. I had the opportunity to approach almost anyone throughout the company which I am deeply grateful for. I would like to thank the agile teams that allowed me to learn from them and all the volunteers who took part in our surveys. Without your support, effort and time, this thesis would not have been possible. Particularly, I would like to thank Peter and Theo who empowered me to do the quantitative study. I would like to thank Anita who was one of the first persons I met at the company when starting this thesis and who truly inspired this thesis. Among others, I would like to thank Martin, Magda and Frederik for good discussions.

Moreover, I received helpful advice and guidance by my mentors Jennifer

and Thomas. We had very interesting discussions which helped me to personally grow.

While working on the thesis I joined many events organized by the PhD students at the Robert Bosch GmbH which was fun but also provided support. Particularly, I would like to thank the WiSo's for hanging out together and being there for each other. I would especially like to thank Ann-Kathrin, Jonas, Flo, Sarah, Ben, Eddy and Sina.

Moreover, I would like to thank my colleagues from the Institute of Software Engineering for good discussions and support. I would especially like to thank Daniel, Marvin and Justus with whom I not only shared an office but also uncountable coffee breaks, discussions and jokes. I would like to thank especially Daniel for providing valuable and honest feedback on our paper.

I would like to thank the research community on agile software development for making me feel welcomed to contribute to the community, for delivering valuable feedback and inspiration at the conferences. Particularly, I would like to thank the Swedish guys Johannes and Lucas: Johannes for fruitful discussions on what leadership actually is and for an inspiring conference in Paris. Lucas for an interesting discussion via skype and for sending me a hard-copy of his PhD thesis on maturity all the way from Sweden. This thesis was a true asset to my bookshelf and inspired me many times.

Last but not least I would like to thank my family who raised me as a critical thinker and independent person and who taught me to pursue my dreams.

Cite

"*Our traditional views of leaders – as special people who set the direction, make the key decisions, and energize the troops – are deeply rooted in an individualistic and non-systemic worldview. Especially in the West, leaders are heroes – great men (and occasionally women) who rise to the fore in times of crises. Our prevailing leadership myths are still captured by the image of the captain of the cavalry leading the charge to rescue the settlers from the attacking Indians. So long as such myths prevail, they reinforce a focus on short term events and charismatic heroes rather than on systemic forces and collective learning.*"

— Peter Senge

Contents

Chapter 1 Introduction

1.1 Motivation

A steadily increasing number of organisations aim at developing their products in a more agile way [Sta14]. Agile teams are said to work cross-functionally and self-organised in iterative learning loops towards a common goal [Con09]. Even though an increasing number of organisations strive to implement agile teams, it is not entirely clear how teams can adopt the agile way of working [MDK09; NMM05]. Especially rather bureaucratic companies seem to struggle in their agile transformation [MDD09; NMM05]. Fitting leadership behavior is found to be one key success factor for evolving into an agile self-organised team [GGJ19]. However, which kind of leadership agile teams need is not yet clear.

While narratives postulate great success stories on agile teams, empirical evidence often reveals how difficult it is for teams to work in an agile way [GBS+16; HM16; HN17; MAD12]. While the agile approach emerged during the 1990th and created a hype on implementing agile self-organising teams, the idea of self-organisation is not entirely new. In fact, the successful implementation of self-organised teams into the industrial sector has been of ongoing interest over the last 70 years.

Researchers consider the topic from different angles among which are socio-technical systems [MS87; SJ17; TB51], knowledge management [TN86], complexity theory [Bäc19; Sch97], role theory [Hod13; Yan96; ZAM09] and agile project management [SS20]. One recurring topic of interest is the role of leadership in a team that is by definition self-organising.

Research on leadership in agile teams mostly differentiates between a leader as peer or coach to the team who provides appropriate boundary conditions (e.g. [TN86]) and an autonomous team that self-organises its operational work (e.g. [HNM12b]). While some researchers suggest a facilitator who serves as a peer to team members [TN86] or a leader who empowers the team to lead itself [MS87], other researchers do not consider a formal leadership role *of* the team but instead emphasize self-organising roles *within* the team [HNM12b; ZAM09].

At present, the most widely known agile approach is Scrum [SB02; SS20] which promises to create agile teams. Scrum divides leadership roles between three different parties: the Product Owner, the Scrum Master and the Developers. The Product Owner cares for the interaction with the customer and sets the requirements for a product. The Scrum Master facilitates the Scrum process, enables the Development Team to work cross-functionally and self-organised, protects the team from external disruption and helps the organisation to adapt to the agile way of working [SS20]. The Developers self-assign tasks towards a shared goal [CH01].

Empirical research on Scrum teams found that the Scrum Master sometimes acts as a barrier to teams working in an agile manner in early stages. The reason is that Scrum Masters tend to stick to a command-and-control mode [MDD10]. Notwithstanding, dismissing a formal leader role altogether was found to decrease teamwork. Research revealed that self-organised teams without a dedicated leadership role establish destructive team norms [Bar93]. However, teams applying agile methods for three years on average appear not to struggle with the Scrum Master and are even supposed to share the leadership role [SJ17].

Yet, there is still an inherent paradox between self-organised teams on the one hand and a formal leadership role on the other hand [MS87].

The diverging results on the Scrum Master role could be explained by changes in the maturity of an agile team. A Development Team learns how to be agile while undergoing different maturity stages [GTF17] and by gradually taking on leadership activities of a Scrum Master. Hence, agility of a team is a process that unfolds over time [WM18]. Yet, to the best of our knowledge, there is no empirical analysis of the changing leadership role of the Scrum Master. Furthermore, most studies have examined the Scrum Master role applying qualitative methods (e.g. [Bäc19; MDD10; SJ17]), while there is a lack in studies to explore these roles quantitatively and to understand how much they change.

Also Development Teams often struggle with taking on leadership roles due to contextual hindrances. Specifically teams in bureaucratic companies appear to face obstacles in working in an agile manner [BT05; NMM05]. This type of organisation contradicts the agile way of working [NMM05]. It is used to rely on rigid planning as opposed to iterative team learning [BT05], on operating in a functional departmentalised structure as opposed to cross-functional teams [NMM05] and on a hierarchical culture as opposed to self-organised teams [MAD12]. Often management is found to undermine the self-organising nature by command-and-control behavior [Hod13].

We can assume that teams that aim at taking on leadership roles struggle to do so if the organisational level aims at sustaining hierarchical structures. Yet, scarce research examines the link between the organisational context and agile team behaviour [CH01; HN17; SMD11]. This thesis specifically focuses on implementing agile teams in bureaucratic organisations.

Recent studies [Bäc19; GL20; SMD11] have called for more research on (a) leadership in agile teams, (b) while taking contextual factors into account and (c) collecting data on leadership from diverse perspectives (d) with mixed methods.

This thesis addresses the suggested topics and focuses on the changing leadership role in bureaucratic organisations while approaching the Scrum Master and the Development Team.

1.2 Research Objective

To be able to support organisations in their agile transformation, my research objective is to explore leadership and how it evolves in agile teams using the example of the Scrum Master. I believe that investigating the changing leadership role of the Scrum Master will provide valuable insights into how teams can adopt the agile way of working.

In order to explain the changing leadership role I refer to role theory, team leadership, maturity and contextual factors. I state that the Scrum Master transfers nine leadership roles (labeled 9-Factor Theory) to the Development Team over time and thus, leadership changes while the team matures. I suggest that this role transfer is influenced by the internal team environment and contextual organisational factors.

The research questions are therefore:

- Which roles does the Scrum Master play to support the team to work in an agile way? (RQ1)
- In which way do developers take on the Scrum Master role overtime? (RQ2)
- How are roles transferred from the Scrum Master to the Developers? (RQ3)
- What is the underlying internal team environment required for the role transfer to occur? (RQ4)
- How can the Scrum Master foster the internal team environment? (RQ5)
- Which expectations on the Scrum Master limit the changing leadership role? (RQ6)
- How do organisational culture and structure influence the role transfer? (RQ7)
- Which leadership roles does the Scrum Master play? (RQ8)
- Which leadership roles does the Development Team play? (RQ9)

- Are leadership roles distributed between a Scrum Master and the Development Team, and if so, is the role more often shared in mature as compared to immature teams? (RQ10)

1.3 Research Strategy

Research differentiates between two different underlying types of reasoning: inductive reasoning and deductive reasoning which classify two different ways of argumentation [Mye19]. The aim of inductive reasoning is to generate hypotheses and develop new theory. Inductive reasoning is an interpretive approach. Data is often embedded into a particular context and captures social construction of the reality of participants [Mye19]. Deductive reasoning aims to test hypotheses that are build on existing theory [May10]. It applies a positivist view and assumes that reality is objective which can be measured by properties.

The aim of this thesis is to provide a theoretical understanding of changing leadership in agile development teams. To tackle the research objective this thesis applies inductive reasoning. We hence assume a social construction of reality. This allows us to capture the meaning and interpretation individuals give to human behavior during the agile transformation.

A minor up-front literature search revealed that publications on leadership in agile software development teams are scarce and lack in theoretical underpinning. Therefore, this research project used an exploratory approach. The theory on changing leadership derived from an iterative process within which industry-based data and scientific literature were constantly compared.

This research is divided into two consecutive parts: a qualitative exploration (part I), and a quantitative exploration (part II). The theory selection and theory building in chapter 4 emerged during the qualitative exploration (part I). The quantitative exploration (part II) deepened the results of the qualitative exploration.

The following two main research parts guide this research project:

(I) Part I is a Grounded Theory based study referring to observations and qualitative interviews. It aims at exploring the evolving leadership role in agile teams using the example of the Scrum Master. I chose Grounded Theory because this method is applied in research fields with scarce knowledge and aims at theory building [GS17]. Grounded Theory follows an iterative approach in which each step is based on the previous step.

To explore evolving leadership in agile teams I collected data in 11 business divisions of the corporation Robert Bosch GmbH, primarily operating in the automotive industry. 75 agile practitioners took part in the qualitative interviews. While collecting data in the field, emerging findings were constantly compared with scientific literature.

To explain agile leadership concepts from social psychology were combined with current state-of-the-art in agile software development literature. The aim was to build a bridge between human factors in empirical software engineering and in social psychology. This brings a new perspective to research on agile teams. Chapters 3 and 4 elaborate on the literature. It is important to mention that even though the literature is described before part I, relevant literature was identified during the Grounded Theory study described in part I and is not based on a heavy up-front literature search.

The result is a substantive theory on changing leadership in agile teams. The Grounded Theory study answers research questions RQ1 to RQ7.

(II) The second part reports on a quantitative exploration and digs deeper into the findings related to RQ1, RQ2 and RQ3 in part I. It answers research questions RQ8 to RQ10. The quantitative exploration builds on the findings of the Grounded Theory and the identified nine leadership roles and designed an online survey, aimed to quantify the presence of the 9 factors (leadership roles) and the maturity of the team. 67 participants from more than 19 different Scrum teams of the Robert Bosch GmbH took part in the study.

Through a quantitative exploration, I build groundwork on examining leadership in agile teams quantitatively and shed light on the distribution of leadership roles among the Scrum Master and the Developers with respect to team maturity. The quantitative exploration does not aim to test the process of the role transfer from one Scrum Master to the Developers.

Table 1.1: Research Strategy

Research Part	Research Activities	Aim
Part I Qualitative exploration Grounded Theory	Minor literature search Unstructured Interviews Continuous literature search Semi-structured interviews Field observations Continuous data analysis Major literature search Theory building	Understanding changing leadership in agile teams.
Part II Quantitative exploration Descriptive statistics	Quantitative survey Analysis	Groundwork for testing leadership in agile teams.

The research methods and data collection procedure are described in detail in the respective chapters.

1.4 Contribution

This thesis supports organisations in their agile transformation by providing practical insights on implementing agile teams in bureaucratic companies. I suggest the changing Scrum Master to be a possible solution to supporting teams in evolving into a truly agile team.

The major contribution of the study is theory building on the changing leadership role (part I) and a quantitative exploration of the presence, change and sharing of leadership roles in agile teams (part II).

Part I reports on a Grounded Theory study. The results, and hence my theoretical contribution, is the description and explanation of:

- *nine leadership roles* of a Scrum Master that are gradually transferred to the Development Team while it matures
- the *role transfer process* including a *leadership gap* which enables team members to take on a leadership role themselves
- an *internal team environment* of being on equal terms, psychological safety, transparency, shared mental models, team orientation, team potency and self-monitoring as enablers of the role transfer
- *contextual factors* of high power distance, specialist culture and rigid organisational structure as hindering factors of the role transfer
- the *Agile Matching Theory* which implies a mandatory fit between contextual factors and the internal team environment for the role transfer to occur

Furthermore, part II describes the quantitative exploration of the leadership roles, which is labeled the 9-Factor Theory, and reveals that:

- the leadership roles are shared to a varying extent between one dedicated Scrum Master and the Developers
- leadership roles were shared most often in teams that had been working between 3 to 5 months in the agile manner
- the percentage of teams who did share the roles was about 20%. No Development Team predominantly took over the Scrum Master role
- despite sharing of some of the 9 factors (leadership roles), most roles predominantly stick with one dedicated Scrum Master, such that the committed Scrum Master did not become obsolete

Based on the results, I suggest to group the 9-Factor Theory along three different clusters: psychological team factors, organisational factors and product-related factors. While psychological factors were linked most often to the Scrum Master, the Developers tended to take on product-related factors. Organisational factors were assigned less often to both parties. I suppose that the results are valuable input for further quantitative testing of the 9-Factor Theory.

The findings provide empirical evidence on leadership in agile teams which help companies in their agile transformation.

Practitioners get insights on leadership roles in an agile setting and understand how the context influences leadership. Scrum Masters get inspiration on how to play their role and empower the team to take on leadership roles. Agile practitioners will understand that an agile team needs time to take on leadership. Managers will increase their understanding on the value of a Scrum Master and get input on providing the appropriate context for leadership in agile teams. Furthermore, I develop a quantitative team survey for determining leadership roles in agile teams which can be used by practitioners.

1.5 List of Publications

Parts of the contributions presented in this thesis have been published in:

1. S. V. Spiegler, C. Heinecke, S. Wagner. “Leadership Gap in Agile Teams: How Teams and Scrum Masters Mature.” In: International Conference on Agile Software Development. Springer. 2019, pp. 37–52.
2. S. V. Spiegler, C. Heinecke, S. Wagner. “The influence of culture and structure on autonomous teams in established companies.” In: International Conference on Agile Software Development. Springer. 2019,pp. 46–54.
3. Söderqvist, Johannes Berglind and Simone Spiegler, "How to enable leadership among self-organising developers." Paper presented at the R&D Management Conference, Paris, 2019, June.
4. S. V. Spiegler, D. Graziotin, C. Heinecke, S. Wagner. “A Quantitative Exploration of the 9-Factor Theory: Distribution of Leadership Roles Between Scrum Master and Agile Team.” In: International Conference on Agile Software Development. Springer. 2020, pp. 162–177.
5. Spiegler, S. V., Heinecke, C., Wagner, S. (2021). An empirical study on changing leadership in agile teams. Empirical Software Engineering, 26(3), 1-35.

1.6 Outline

Eight chapters guide this thesis. Following this introductory chapter, Chapter 2 introduces background information on traditional and agile software development and outlines the change from traditional to agile teams.

Chapter 3 provides a literature review on leadership in agile teams. Section 3.1 introduces the term leadership and illustrates different perspectives on the term. 3.2 provides an overview on agile leadership, and elaborates on the leadership role of a Scrum Master in specific in Section 3.2.2. A discussion closes Chapter 3 and identifies the changing leadership role of a Scrum Master as a research gap.

Chapter 4 refers to the selected literature that will be used to build a theory explaining the evolving Scrum Master role in Section 4.2.

Chapter 5 refers to part I by describing a Grounded Theory study. Section 5.1 outlines the study design by explaining the research method, the company context, the data collection procedure and data analysis. Section 5.2 illustrates the results deriving from data analysis and suggests propositions for future testing. Chapter 5 closes with a summary, discussion and limitations related to part I.

I would like to emphasize that the extensive literature review and theory building emerged during the Grounded Theory study over a period of approximately two years. However, for a better understanding of this thesis, I decided to first describe the literature review, theory selection and theory building and later on elaborate on the Grounded Theory study in part I.

Chapter 6 refers to part II of this thesis which reports on the quantitative exploration of the key propositions deriving from Chapter 5. Section 6.1 recapitalizes the findings deriving from the literature review and the Grounded Theory study. Section 6.2 describes the research design, including research method, data collection and analysis. It briefly repeats the company context as describe in 5.1, while it adds new insights on the Scrum Master description in the specific company context. Moreover, it characterises the sample which differed from part I. The results are reported in Section 6.3 followed by a discussion, practical implications, limitations and future work related to the quantitative exploration.

Chapter 7 summarizes the findings in relation to the research questions and discusses the results of part I and II while referring to existing literature. Moreover, Section 7.3. outlines practical implications for the Developers, the Scrum Master, the Product Owner, the management and the organisation. Section 7.4. outlines further limitations while building a link between part I and part II. Moreover, it refers to a research cooperation with another large company active in the automotive industry during this thesis that additionally supports our findings. Section 7.5 closes Chapter 7 with future research directions.

Chapter 8 completes this thesis with a brief summary.

Chapter 2

Background

This chapter aims at providing background information to place the research topic into a broader context.

During the last decades software development projects have used distinct approaches to the way software is build. Researchers most often differentiate between traditional development and agile development. Software processes and models in different periods of time emerged from the respective historical context. The following will scratch on traditional software development. For a detailed overview, please refer to Boehm [Boe06] for the historical background and to Ruparelia [Rup10] for the distinct methods.

Section 2.1 refers to the background of traditional software development by describing the historical context, deriving development methods and way of interaction among people. Section 2.2 explains the need for a change of traditional development methods to more agile development. Agile software development is characterised in Section 2.3. Section 2.3.2 explains the Scrum framework and related Scrum roles in specific, while Section 2.3.3 refers to the underlying human behavior in agile development by describing agile teams.

2.1 History of Software Development

2.1.1 Traditional Software Development

In general, every development project considers requirements specification, implementation and modification [Som04]. Yet, in which order the steps are organized, if they are overlapping or conducted in consecutive order and who is involved how in which step and when to go to the next step varies.

The 1950's refer to early software development. Often hardware engineers became software developers and treated it like hardware engineering [Boe06]. Before running the code by using punch cards on the mainframe computer, engineers would do extensive and careful planing by using handwritten notebooks, since running the large machines was very expensive [Boe06].

In the 60's the first organized way of developing software was the code and fix approach, in which coding was followed by requirements, testing and maintenance [Boe06]. Yet, this approach was expensive and managers aimed to keep costs for development low [Boe88; Roy87]. Moreover, in the 60's authority was taken into question. Students at university expressed great admiration for programmers who could develop quick-and-dirty over night to meet deadlines [Boe06].

Authorities felt like developing software was invisible and it was difficult to determine if it was on track [Boe06]. Moreover, adjustments were expensive. Therefore, managers felt the need to control, to manage and to organize the way engineers developed software [Roy87]. The idea of a defined order of consecutive development steps and transition criteria was born [Boe06].

During the 70's a number of methods emerged that build on control and use of hierarchy to manage transition through the distinct stages [Boe06]. Moreover, it provided tools to present results in each step via structure charts, state transition diagrams and stimulus response threads [Boe88]. One method that became popular during this time is the waterfall model.

2.1.2 The Waterfall Model

One traditional development method is the waterfall model, rooted in the cascade model by Benington [Rup10], and originally suggested by Royce in 1970 as software development based on specification [Roy87]. The cascade Model by Benington suggests to develop software along the consecutive stages *operational analysis, operational specification, design and coding specification, development, testing, deployment and evaluation* [Rup10].

Royce warns that a sequential order of the steps could lead to a doubling of time or budget if testing at the end of the development process failed. To avoid returning from testing to requirements or design and to restart the whole process, he introduces a feedback loop in which each preceding phase can be revisited. Thus, Royce developed the model further by integrating a repetition of each preceding step [Roy87]. The waterfall model refers to the phases *system requirements, software requirements, analysis, program design, coding, testing and operations* [Roy87].

It is a document-driven model and Royce recommends heavy documentation at the end of each stage that allows for tight control [Roy87].

Furthermore, Royce suggests to integrate the customer at three defined formal reviews. He considers it too risky to integrate the customer between requirement definition and operation and proposes to invite the customer twice in distinct program design phases and once after testing for a final software acceptance review. It is up to management to decide upon the time and the person who will deliver the software. [Roy87]

Even though the original idea of the waterfall model was a consecutive repetition of the different steps it evolved into a linear process among practitioners due to government process standards and contract negotiations about requirements [Boe06]. It is described as a rigid up-front planning by setting requirements and building design followed by a plan execution terminating in the implementation of an entire system at once. A stage-gate with a set of criteria that needs to be fulfilled to pass on to the next step for each phase is suggested to serve as a source of control [Boe88]. Time schedules, target dates and budget are said to guide the project.

Later on practitioners outline several weaknesses. Neglecting the iterative loops, practitioners often link it to a heavy up-front planning followed by a rigid process along sequential steps, a lack of customer integration, and expensive changes due to a revision of planning and designing in case of unsuccessful testing at the end of the development project.

A number of other models build upon the waterfall model and developed it further, among them the b-model [BO88], the spiral model [Boe88] and the v-model [FM91]. For example, the spiral model promised a better integration of customers by prototyping [Boe88]. The b-model included a maintenance cycle for continuous improvement [BO88].

Yet, most often agile methods are compared to the practitioner's sequential interpretation of the waterfall model by Royce.

2.1.3 Traditional Software Development Teams

Traditional development teams are embedded into a hierarchical organisational structure based on several layers and assigned authority over others [TH04]. Traditional software development is a mechanistic approach which can be described by the underpinning believe that problems can be fully specified and understood [NMM05]. This implies that there is an optimal and predictable solution for any challenge. Extensive up-front planning and tight control are considered to be success factors for traditional projects [Roy87].

It is believed that an individual manager provides the solution to any problem at hand by knowing and understanding everything [NMM05]. Moreover, the manager is responsible for most decision-making [NMM05]. The person in charge demonstrates command and control behavior. The manager gives orders to the Developers about what needs to be done by assigning tasks and roles to individuals [NMM05]. The manager tracks developers' work progress by heavy documentation [Boe06; NMM05; Roy87].

The development process defines specific tasks for each step including the desired results [NMM05]. Developers cooperate via strict labour division and assigned roles with specialization [TH04]. Individuals are used to work alone or in homogeneous groups of analysts, designers, testers or architects [NMM05]. The developer does not have to understand the broader idea of the project but excels the specified assigned task [TH04].

Royce [Roy87] suggests to strictly separate between designer and tester. He raises the opinion, that a test specialist would do a better job at testing than a designer, despite acknowledging that the designer understands the area he or she has developed. The required work material for the tester is a thorough documentation of the preceding steps. Therefore, developers create heavy documentation that stores knowledge and serves as a tool for communication between process stages. Therefore, traditional teams rely on formal ways of communication.

2.2 The Need for Change

In the 1990's an increasing number of companies abandoned traditional software development. The root cause is considered to be the internet and the World Wide Web [Boe06]. The internet makes it possible to spread news rapidly and to access information almost immediately [BLPS01]. User feedback can be integrated quickly into product development and customer orientation is even more important [BLPS01]. Moreover, computers have almost no speed-limits any more and memory and storage capacity are not far from unlimited [Boe06], also via cloud applications. As a consequence technology develops rapidly and new products access the market fast.

Moreover, product development is often characterized as complex which means that a high number of interaction among multiple individual parts, including tasks, people and technology, leads to obscurity of cause and effect [SLKC01]. During project progress circumstances and requirements continuously and rapidly change. Knowledge at the beginning of a project might become obsolete during the project and replaced by new insights. Development projects involve a high level of uncertainty and the exact outcome is unpredictable in advance.

Velocity, complexity and uncertainty lead to a change in the way people in an organisation interact and collaborate. For example, firms interact differently with the environment by building more customer-oriented products and increasing speed-to-market [BLPS01], while individuals have to make decisions on incomplete information [CH01; Mad07]. To minimize risks, projects are managed via short iterations.

Companies that are built upon rather traditional development methodologies face several challenges when changing the way in which products are developed [MAD12; NMM05]. Existing companies cannot transform into faster and more adaptable companies by simply exchanging existing tools and methods with new ones [HN17; NMM05; SNM01]. A change requires several years of time and effort on many different levels [HN17; MAD12; NMM05]. Companies change business models and architecture, project management, processes, tools and methods, management, leadership and collaboration and deriving structure and culture [HN17; MAD12; NMM05].

2.3 Agile Software Development

2.3.1 Agile Software Development

Agile development is one way that promises fast reaction to a rapidly changing environment [WC03] and increasing complexity [Sch97].

Even though referring to a hardware development context, Nonaka and Takeuchi [TN86] are often cited as the pioneers to describe 'agile' software development in 1986. Their paper describes a successful product development process based on self-organised teams that continuously learn and exchange knowledge in order to build new products based on given requirements and high trust of management [TN86].

In the meanwhile, the concept 'agile' evolved rapidly. Anecdotal narratives on success stories about agile teams by practitioners and consultants have launched a hype on the agile way of working [Sta14]. Despite a history of about 30 years of agile software development, there is no common definition of the term 'agile' [LSA11]. Sometimes, the word agile is criticised as the 'silver bullet' since it promises just anything [LSA11]. Some authors refer to building high performing teams, while others focus on optimizing the entire organisation [LSA11].

In 2001 a group of representatives of different agile development methods met in a ski resort in the Wasatch mountains of Utah to discuss and align on the meaning of ’agile’ [Hig20]. The result was the ’Agile Manifesto’ which is considered the first document to explicitly describe and generalize agile software development. The Agile Manifesto enumerates 12 principles and the following four values:

- individuals and interactions over processes and tools
- working software over comprehensive documentation
- customer collaboration over contract negotiation
- responding to change over following a plan

The four values do not describe an either/or decision between different options but urge to emphasize more on the values on the left side than on those on the right side.

The aim of agile development is to continuously build fast deliverable features of a product with high customer integration, and therefore, to welcome and receive fast feedback. Consequently, projects can adapt quickly to changing requirements. Agile methods are frameworks to empower development teams to organize their knowledge and self-organise their work.

There are a number of agile software development approaches among which are: Feature Driven Development [PF02], XP [Bec00], Kanban [HS14], Lean Software Development [Pop07], Scrum [SB02] and Adaptive Software Development [HH02b]. Each approach focuses on different aspects of software development [LSA11]. For example, while XP aims at the implementation of software [Bec00], Scrum is an iterative development process in project management [SB02].

Since this thesis examines teams that apply Scrum I describe this framework in detail.

2.3.2 Scrum

Schwaber [Sch97] wrote a white paper on the idea of the Scrum framework which he introduced to the Software community during the OOPSLA conference in 1995. He borrows the word 'Scrum' from a paper on product development teams by Nonaka and Takeuchi to build the presented Scrum framework and states that Scrum is already applied successfully in software development companies.

The white paper [Sch97] refers to complexity theory and defines Scrum to be a development process that symbolizes a 'controlled black box'. Scrum was originally invented for managers to not control development teams directly, but to build a process framework as a tool for control.

The paper evolved into the Scrum Guide which was published in 2010 and has been revised seven times until 2020 by support of agile practitioners via a community website [Cor20].

2.3.2.1 The Scrum Framework

Scrum consists of Scrum events, roles and artifacts and is based on common values and principles [SS20].

Scrum is based on the following values: *commitment* to reach the common goals, *courage* to do what is right, *focus* on the respective sprint goal, *openness* about work and accomplishing it, and *respect* each other to be competent, independent people. Teams are said to *trust* one another.

Scrum divides the product development project into regular short iteration cycles. At the end of each cycle, developers provide a new product feature to the customer and receive feedback for further improvement. A single iteration cycle is called 'Sprint', lasts 4 weeks maximum, and contains four different types of events: the Planning, the Daily, the Review and the Retrospective, in respective and repetitive order.

1. The Sprint Planning helps the team to decide what the next product increment should contain and how the respective goal will be reached in a self-organised way.

2. The Daily serves as a daily meet-up for transparent synchronization of work progress of the Developers and short agreement on how to continue the next 24 hours to focus on the committed sprint goal.
3. The Sprint Review is an opportunity for the team to present its progress to the Product Owner or directly to the customer and for receiving feedback. It is supposed to be an informal meeting and not a status report.
4. The Retrospective is facilitated by the Scrum Master and helps the team to reflect upon what is going well in the team, what needs improvement and how to do so.

There are different artefacts which allow the team for a maximum of transparent communication, coordination and synchronization to develop a shared understanding. The Product Backlog enumerates all features, functions, requirements, enhancements, and fixes that are needed in the product and which are defined by the Product Owner. The Sprint Backlog contains specific items from the Product Backlog. The items describe what the Developers promise to be the next deliverable features by the end of one Sprint. Scrum is said to help delivering the highest business value in a limited scope of time by prioritization.

2.3.2.2 Roles in Scrum

The Scrum Guide does not mention managers and divides the work between three different roles: the Developers, the Product Owner and the Scrum Master [SS20]. In the following I will describe each role according to the Scrum Guide [SS20].

Developers work self-organised and decide independently from the Scrum Master and the Product Owner how they turn the Product Backlog into deliverable features. A team is cross-functional and consists of dedicated full-time developers with all the skills needed to create the feature. Despite potential specialization in testing, architecture, operations or business analysis the team members are accountable as a team. In this thesis, I use the term *developers*, *team members* and *Development Team* interchangeably.

The Product Owner is responsible for value maximization for the customer. The role keeper is responsible for the Product Backlog and related prioritization. The Product Owner defines the focus for the up-coming sprint and creates a transparent and common understanding on what needs to be done. If someone intends to change the Product Backlog or prioritization, the Product Owner needs to be addressed. This thesis focuses on how the Scrum Master role is shared between a formal role keeper and the Developers and refers to the Product Owner as a contextual factor (more in Section 5.2.3.4).

The Scrum Master role is described to serve the organisation, the Product Owner and the Developers. The role teaches the Scrum method, serves as a facilitator of the Scrum process and makes sure that the rules and processes are followed.

The Scrum Master is said to be a true leader instead of a manager and to replace the traditional project manager. The Scrum Master is assigned an indirect authority and even though it embodies an assigned role, he or she has to earn the trust of the Developers by demonstrating true commitment to the team. The major challenge of this role is to develop from control of teams to empowering for self-organisation, cross-functionality and a maximum of productivity.

The role protects the team from interruption during the sprint and removes impediments for the Developers and the Product Owner so that they can drive development. Additionally the Scrum Master teaches the Product Owner how to maximize ROI and reach the respective sprint goal through Scrum. Furthermore, the Scrum Master serves as a change agent to the organisation and makes the stakeholders understand Scrum. It is the Scrum Master's responsibility to fit the Scrum process to the respective organisational context.

Several studies have examined the role of a Scrum Master while referring to leadership theories [Bäc19; MDD10; SJ17], which is also the focus of this thesis. I provide a literature review on research on the leadership role of the Scrum Master in Section 3.2.2.

2.3.3 Agile Teams

Scrum teams cannot be automatically labeled as 'agile'. Schwaber [Sch04] points out that even though a team applies the Scrum framework according to the book it may still not behave in an agile way. The connection among people is more important than keeping to a certain process for successful software development [CH01]. Therefore, the following sections describe the human interaction of agile teams.

2.3.3.1 From Traditional to Agile Teams

While traditional project management relied heavily upon planning, documentation and establishing routines and standards, a complex project with frequently changing requirements cannot be planned and controlled upfront [Bon10; BT05; NMM05]. Therefore, projects shift from traditional project management which follows rigid structures and processes to agile project management which is more flexible, adaptable and customer-oriented [NMM05]. Agile software development is based on an organic approach which can be described by the underpinning believe that problems cannot be fully understood when they emerge and need social collaboration to be solved successfully [NMM05].

Instead of long-term planning, projects plan in short iteration cycles, and develop customer-oriented by continuously integrating feedback [NMM05]. Continuous team learning helps agile teams to cope with complexity and frequently changing requirements [VW09]. Learning behavior implies that individuals interact frequently with each other, share their knowledge openly, learn quickly from each other, include early feedback from customers and learn on the job while developing [VW09]. Thus, agile teams continuously adapt to new conditions and according to lessons learned [Sch16].

While project members in a traditional setting rather worked in solitary and departmentalised [NMM05], agile teams work cross-functionally within and across (organisational) boundaries [Con09]. Despite cross-functional cooperation, developers continuously balance specialization and cross-functionality [HNM12a].

While managers used to assign tasks and roles to individuals, Developers self-organise their own work, e.g. by picking their tasks voluntarily [CH01]. Traditional management behavior that plans, decides and controls is considered to be less effective in a complex product development [Bäc19]. A fast moving environment requires decision-making on the appropriate level to increase speed of product development and handle customer demands quickly [MAD12]. Even though several authors claim that the project manager maintains responsibility for project management duties the role changes and is supposed to be more of a facilitator who fosters self-organisation of teams [Tay16; TN86; VW09].

2.3.3.2 Self-organising Teams

Research on agile teams refers to the concept self-organisation, but also to other seemingly alike concepts, including autonomy and self-management.

Self-organising teams manage their own work and organize their team tasks independently [CH01], while the manager provides the context within which Developers can work self-organised [TN86]. Managers provide an environment that facilitates group decision-making, and focus on setting the boundary conditions and on providing a vision [Bon10; CH01; MAD12; TN86].

Autonomous teams define their own goals, make decisions regarding task allocation and execution and solve their problems independently [LWRJ05]. The effectiveness of autonomous teams is influenced by interference by management [HP06]. For example, a team may take over more or less responsibility depending on the rights a team perceives.

Self-managed teams can be defined as "a group of individuals with diverse skills and knowledge with the collective autonomy and responsibility to plan, manage and execute tasks interdependently to attain a common goal." [**p.4**; MP18]. *Team leadership* describes that one or several individuals of the self-managed team take on leadership activities and is sometimes used to describe the Scrum Master role [MCDE15; MDD10]. The capability of teams to self-manage depends on the cultural and structural context within which they operate [MP18].

Table 2.1: Team Types with a High Level of Freedom

Concept	Description
Self-organising teams [CH01]	Teams manage their own work and organize their tasks independently.
Autonomous teams [LWRJ05]	Teams define their own goals, make decisions regarding task allocation and execution and solve their problems independently.
Self-managed teams [MP18]	A group of individuals with diverse skills and knowledge with the collective autonomy and responsibility to plan, manage and execute tasks interdependently to attain a common goal.
Agile Teams	Teams work cross-functionally, self-organised and in iterative learning loops towards a common goal.

While some authors emphasize the differences between the concepts (e.g. [OB16]), other authors treat agile teams, self-organised teams, autonomous and self-managed teams interchangeably (e.g. [MP18]).

I believe that the concepts overlap. Even though the different team types promise a high level of freedom to organize work independently, the teams are limited or enhanced by the context within which they are embedded [DW03; HP06; MP18]. For example, a compelling vision and high level of trust by superior management empowers the team to have a common goal, organize their work accordingly and deliver value [CH01; HH02a; HM16]. Furthermore, the concepts predominantly refer to mature teams, while only a few studies examine how self-organisation can be developed [HN17].

A few researchers describe an agile team to undergo different development stages until they become a high performing, truly agile team [CH01; GTF17; HN17]. Cockburn [CH01] refers to the Japanese philosophy of Shu-Ha-Ri and describes Scrum as a maturity model for agile adaption. Shu implies that teams who are new to the Scrum framework should imitate Scrum according to the book. Ha describes to gradually develop an understanding of the underlying reason behind the Scrum ceremonies. Ri expresses that an individual is now ready to be a teacher of Scrum. This stage may involve adapting the Scrum framework according to the context. Depending on the maturity level teams may practice the agile manner differently [CH01; GTF17]. Nevertheless, more research is needed that explores how teams can advance the agile way of working [HN17].

I believe that leadership contributes to understanding how a development team can develop into a truly agile self-organised team. My objective is to contribute to research on agile adaption with a specific focus on leadership in agile teams by investigating the role of a Scrum Master. I believe that the core aim of the leadership role of a Scrum Master is to enable the Development Team to work in an agile way while it matures. Furthermore, I suggest that leadership can be shared between one dedicated Scrum Master and the Developers.

Chapter 3
Literature Review

This thesis aims to contribute to research on leadership in agile teams. The aim of this chapter is to gain an overview on the literature on leadership in agile teams and to identify a matter of contribution on the topic.

This chapter is organized as follows: Section 3.1 provides different perspectives on the term leadership and therefore allows for a basic understanding of leadership as used in this thesis. Section 3.2.1 summarizes literature on agile leadership, while Section 3.2.2 describes leadership research on the Scrum Master role in specific. Finally, the findings are discussed critically in Section 3.2.3 and a topic of contribution is identified.

3.1 Leadership

Leadership has been a heavily explored research topic for more than one century already [Gri10b]. Even though there is an infinite number of definitions and theories on leadership, there is still no agreement on the essence of leadership [CLP10; Gri10b; Sto74; Yuk13]. Some perspectives on leadership strongly disagree on who can be considered a leader, how leadership is created and what the aim of leadership is [CLP10; Gri10b]. Since leadership is a heavily explored research topic it would exceed the scope of this thesis to provide detailed information on leadership. This overview only scratches on leadership to provide background information for understanding this thesis. Indepth information on leadership concepts and theories can be found in comprehensive textbooks, e.g. Yukl [Yuk13].

Research differentiates between a manager, a formal and informal leader and leadership. Even though management and leadership are linked to each other, most researchers do not consider them to be congruent [Yuk13].

While a traditional manager is often associated with decision-making power, control, monitoring, planning and coordinating [Col04], a leader is described to empower individuals to work well together by sharing power with the team [KW88] and by providing boundary conditions within which followers can nurture [CH01]. A formal manager in a complex environment should show more facets of a leader since complex environments with frequently changing requirements cannot be controlled and organized [Bäc19; NMM05].

A manager refers to a formal position, while a leader and leadership are sometimes related and sometimes unrelated to a position and formal authority [KW88]. Even though a leader can be one individual who influences a follower, a leader can also be or become a follower [Yuk13]. For example, in one situation person A may be the leader and person B may be the follower, while in another situation person B may be the leader and person A may be the follower. Yet, studies on leadership roles without formal authority over others remain scarce [Bäc19; MDK10].

Besides, while most research examines the relationship between a leader and an individual, research also examines the relationship between an individual and a team (e.g. [Cha91; MDK09; MDK10]).

In contrast to an individual leader, leadership can be considered as a social influence process within which anyone in the team can take over leadership activities [MDK10]. Shared leadership may either mean that anyone in the team takes on any leadership activity or that certain leadership activities are linked to defined roles (e.g.[MDK10].).

Furthermore, some authors describe leadership to be a result of social interpretation of reality which derives from the prevalent culture under study [CLP10]. For example, a development team that is embedded in a culture of strong hierarchy is used to command-and-control behavior by management and may not accept empowering behavior [GTF17]. A manager who shares power instead of commanding followers what to do may be perceived as an incompetent manager. However, the influence of the context and situation on the emergence of leadership is often neglected [KMC16].

Moreover, researchers disagree on the definition of leadership. An often cited quote states that "there are almost as many definitions of leadership as there are persons who have attempted to define the concept". [**p.7**; Sto74]. Most often research defines leadership as social influence process towards a common goal [Yuk13]. Latest research describes leadership as a collective social process emerging through the interactions of several individuals often expressed in complexity theory [UMM07]. Complexity leadership behavior claims that outcomes cannot be controlled for in advance anymore, therefore, are not directly manageable and rather emerge through the interaction of groups [Bäc19].

Nevertheless, there is no right or wrong definition of leadership but defining leadership depends on the pursued goal of any study [Gri10b]. So far, there is no definition of leadership in agile teams. I will explain the perspective on leadership in this thesis in Section 4.2.1.

3.2 Agile Leadership

3.2.1 Overview

In the following I provide a general overview on leadership and agile teams. I conducted a literature review using keywords related to ’agile’, ’agile team’ or ’scrum’ in combination with ’leadership’, ’leader’, ’scrum master’ on ebsco, Web of Science Scopus, IEEE Xplore, ACM Digital library, AIS (Association for Information Systems) eLibrary, Springer Link, and Google Scholar. I had a thorough look at the titles and abstracts of the results while considering quality criteria including publisher, reliability, objectivity and validity.

There is a limited number of studies on human behavior in agile software development and there is even less research on leadership in agile teams. While multiple experience reports and theoretical essays narrate on leadership in agile teams, only a few articles are based on theory and valid methods. A large majority of the resulting literature is subjectively written, and provides neither a theoretical background nor a valid method of data collection. Often the story is based on personal experience and thus subjective interpretation of the author him- or herself whilst empirical research on leadership in agile teams remains scarce.

While the majority of the studies is based on qualitative research including interviews and case studies (e.g. [GL20; SJ17; VW09]), a few studies are based on quantitative data (e.g. [HK18; Kak17; LLDL15]). Researchers tend to examine agile teams by mainly interviewing managerial positions, whereas interviewing team members and Scrum teams in specific remains scarce.

Despite an increasing interest in the topic of leadership in agile teams in recent years, scientific literature does not agree upon defining leadership in agile teams. *Agile Leadership* depicts a term often referred to by practitioners. I have not come across any definition of the term agile leadership in research on leadership in agile teams. Moreover, researchers tend to not define the term leadership.

The role and perspective on leadership in agile teams differs and there is no common agreement on who can be considered a leader in an agile team. The literature research identified several parties who exert leadership. Most studies focus on a leader who is external to the agile team. An external leader is for example a superior manager [Pri10; TN86], a project manager [Bon10; Pri10; VW09] or a manger who leads several teams [Fis19]. Besides a formal position, people with the right mind-set and knowledge can help teams to innovate successfully and, therefore, take on leadership [BTPH17; HNM12a]. Moreover, coaching is said to be more important than decision-making, since teams develop expertise in innovation with high uncertainty and ambiguity [BTPH17].

A few authors examine the leadership role of a Scrum Master. Research categorizes the role contradictory: either one individual role keeper serves as a facilitator and coach [Bäc19; GL20; GTF17] or a team shares leadership activities [MDK09]. Moreover, authors describe a shift from one individual leader to sharing of the role [MDD10; SJ17].

I focus on presenting research on the leadership role of a Scrum Master. Managers are considered a contextual factor in this thesis and will be further described in Section 5.2.3.4.

3.2.2 Leadership in Agile Teams

The leadership perspective on the Scrum Master varies among researchers. I will first elaborate on research that focuses on individuals as leaders and afterwards outline research that links the Scrum Master to shared leadership.

Bäcklander [Bäc19] refers to complexity leadership behavior and considers the Scrum Master to be an *enabling leader*. Complexity leadership theory [UMM07] states that leadership in knowledge-intensive companies that face complexity need leadership that focuses on outcomes that allow for change, adaptation and innovation.

Bäcklander [Bäc19] emphasizes that leadership aims at influencing the social interaction among individuals and describes the role of leadership to enable team dynamics for emergent outcomes. Leadership influences team interaction such that developers create innovative products as a team. This may also involve influencing the context within which teams are embedded.

The study by Bäcklander draws data from a newly established software company. Since its implementation the role without disciplinary power had evolved from specifically focusing on the Scrum Method to caring for team dynamics, being a facilitator and working with an agile mind-set. The enabling leader aims at improving team dynamics, instead of reaching a specific team result, and at continuously balancing between providing a structure, e.g. with the help of the Retrospective, and not providing a structure, e.g. by being absent. The author concludes that complexity leadership and team literature are tightly linked together, since enabling leadership provides space within which team processes, such as team learning [Edm99; MMRG08], and therefore adaptive behavior and innovation, can happen.

The 2017 version of the Scrum Guide characterizes the Scrum Master to be a servant leader. Servant leadership describes that a leader serves followers [Gra91]. The relationship between a leader and followers is built upon mutual caring and trust which is supposed to lead to autonomy and empowerment of followers [Gra91].

Holtzhausen and De Clerk [HK18] conduct a quantitative study on *servant leadership behavior* of a Scrum Master in more than ten different organisations. Their findings illustrate that servant leadership behavior leads to higher team performance if the Scrum Master is simultaneously the disciplinary team leader of the respective team. A Scrum Master without disciplinary power who demonstrates servant leadership is found to be less effective.

A study on group development of agile teams [GTF17] refers to *situational leadership theory* which derives from the contingency stream. Contingency theory suggests that the situation influences the effectiveness of a leadership style [Fie67]. While the original theory by Fiedler claims that an individual cannot change the personal leadership style and should therefore be selected depending on the situation, Gren et al. [GTF17] refer to the further development by Hersey et al. [HBJ07] who coin situational leadership and describe that the leader should change behavior if the followers develop. Thus, the Scrum Master behavior should adapt to the maturity stage of a team. Gren et al. [GTF17] speculate that a rather immature team needs structure and order to organize themselves. They mention *agile leadership* which they describe as facilitation by the Scrum Master. They speculate that an immature team will not accept agile leadership behavior, while a mature team is able to self-organise which allows the Scrum Master to step back [GTF17]. An additional study builds on this idea and provides further support for the claim that leadership behavior adapts to group maturity [GL20]. Moreover, data reveals that leaders adapt their behavior to company specific culture and structure [GL20].

In contrast to the aforementioned perspective on the Scrum Master being linked to one individual leader, other authors [MDD10; SJ17] refer to leadership sharing between one dedicated individual and the whole team.

Moe, Dingsøyr, and Dybå [MDD10] refer to *team leadership* from research on self-manged teams to describe the Scrum Master. They explore teamwork challenges of a newly established Scrum team over a period of nine months and discover that one dedicated Scrum Master took over a team leadership role most of the time. The formally appointed person even reduced self-organisation of the team by starting to control team members. Therefore, team members stopped revealing their impediments which resulted in weak team leadership and lack in trust. However, the authors acknowledge that while the team matured, team members started to take on more responsibility. Moreover, the authors state that teams need management support and resources to grow into self-organisation.

Srivastava and Jain [SJ17] provide a leadership framework of the Scrum Master role in locally distributed teams that had been working in an agile way for three years on average. They refer to *super leadership* [MS87] which also derives from research on self-managed teams. Super leadership describes a Scrum Master to enable a team to lead itself. They conclude by suggesting a rotational leadership role of a Scrum Master which is shared among team members depending on the situation. Also Moe et al. [MDK09] describe rotating leadership depending on knowledge, skills and capabilities. They further propose that the team needs a team leader who designs the team and manages the boundaries of the team.

3.2.3 Discussion of the Literature

Despite an increasing interest in leadership in agile teams, research with valid methods and theoretical underpinning remains scarce. Even though a lack in appropriate leadership was often found to decrease agile team performance [CH01; HM16; HN17; MAD12] a shared understanding on what leadership in an agile context denotes is yet missing [Den15]. Until today, no one measured nor defined agile leadership successfully. Thus, there is more research needed examining focus topics of leadership in agile teams while referring to scientific literature.

This thesis focuses on leadership in agile teams using the example of the Scrum Master. The scarce empirical findings reveal contradicting results on the Scrum Master role. While some authors refer to the Scrum Master as an individual leader [Bäc19; GL20; GTF17; HK18], others link the role to shared leadership [MDD10; MDK09; SJ17]. Some authors describe that a Scrum Master struggles with command-and-control behavior while developers neglect to take on leadership responsibilities [NRBB17; SMD11]. Consequently, researchers still struggle with the inherent paradox of leadership in a team that is actually supposed to be self-organising. Thus, leadership in agile teams is not yet sufficiently understood.

Some authors refer to a changing leadership role by referring to team maturity. Gren et al. [GTF17] refer to situational leadership and state that an immature team needs more command-and-control behavior by a formal leader while a mature team rather needs agile leadership. Yet, even though situational leadership provides a feasible explanation, this theory still assumes that there is an individual leader guiding the Developers. Hence, describing a leader-follower relationship. Likewise, an enabling leader [Bäc19] and a servant leader [HK18] refer to leader-follower relationships and do not provide answers how shared leadership responsibilities emerge.

Moe et al. [MDD10] refer to *team leadership* from research on self-managed teams to describe sharing of the leadership role by one dedicated Scrum Master and the Developers over time. Likewise, Srivastava and Jain [SJ17] use the concept *super leadership* from research on self-managed teams to describe the rotational Scrum Master. However, the authors do not describe how team leadership evolves while the team matures.

My aim is to explore and portray the characteristics of leadership in immature as compared to mature teams and to describe how leadership changes. I believe that the Scrum Master plays diverse leadership roles which can be transferred from one individual to distinct team members during the maturity journey.

I intend to explain the changing Scrum Master by integrating research from self-managed teams into research on agile development teams. I do not only refer to team leadership as Moe et al. [MDD10] and Srivastava and Jain [SJ17] but additionally refer to role theory. Role Theory is also used to explain leadership in self-managed teams [Yan96; ZAM09]. Besides integrating existing research on leadership in self-managed teams to leadership in agile teams, I build new theories to explain how a team can adapt the agile manner.

Furthermore, this study follows the suggestion by Crevani [CLP10] who states that research on leadership should take the culture into account. Bäcklander [Bäc19] conducts her study in a start-up environment, while this thesis collects data in established companies. Established companies are based on hierarchical culture and departmentalised structures deeply imprinted in the organisation which influences the way leadership unfolds in real company setting [GL20]. Teams may struggle to share leadership while leaving departmental silos and hierarchy behind. Research related to self-managed teams refers to contextual factors that influence the self-managing capability of a team [MP18]. However, also research examining agile teams refers to the influence of the context [GBS+16; GL20; HM16; MAD12; NMM05]. I explore in which way the context influences developers to take on leadership roles.

How a team can learn to work in an agile self-organising way is still left unexplained. I aim to contribute to research on agile teams by focusing on leadership. To the best of my knowledge, the theoretical suggestion of a changing leadership role while the team matures [GTF17] has never been empirically tested. Despite referring to self-organised teams most research describes an individual to be the leader of the agile team. This thesis aims to provide empirical support that leadership evolves while the team matures in such a way that leadership activities are shared between the Scrum Master and the Developers over time.

To add to literature on human behavior in agile development I refer to team leadership, role theory, maturity and contextual factors. These theoretical streams help us explain how leadership activities shift from a dedicated Scrum Master to the Developers while the team matures. Research on self-managed teams provides a valuable opportunity to contribute to leadership in agile teams.

While empirical data on the Scrum Master is often missing or merely based on single case studies, I provide qualitative and quantitative data from multiple teams operating at different companies of one conglomerate. Besides qualitative data I provide quantitative data to increase scientific proof of our concept.

Scarce research refers to existing body of knowledge from leadership literature and to empirical data to explain the Scrum Master role. Scientists call for more research on team leadership in agile teams that are by definition self-organised [SMD11]. This research tackles this call for action with the objective to: (a) integrate team leadership and role theory to research on human behavior in agile software development, (b) describe the changing Scrum Master role, (c) provide empirical data, (d) contribute to a better understanding on leadership in agile teams to advance research on human behavior in agile development research.

Chapter 4

Theory Selection and Theory Building

While leadership is sometimes found to be centred on one individual, the role is also described to be shared among one individual and the agile team. This thesis aims at exploring the controversial empirical findings on leadership in agile teams. The key message of my concept postulates that leadership evolves while the team matures. I intend to explain the changing leadership role by combining research on the Scrum Master from the agile software community with role theory and research on team leadership in self-managed teams.

Section 4.1 elaborates on literature that is used to build the theory in Section 4.2. After an introduction to team leadership several team leadership concepts from research on self-managed teams are explained in Section 4.1.1. Subsequently divers perspectives and concepts related to role theory (Section 4.1.2) and maturity (Section 4.1.3) are illustrated. Moreover, agile team features (Section 4.1.4) and contextual factors (Section 4.1.5) that influence agile teams are described.

Section 4.2 expands Section 4.1 and builds the theory on the changing Scrum Master role which emerged through the Grounded Theory study reported in Chapter 5. First, the general perspective on the leadership role of a Scrum Master is explained in Section 4.2.1. By relating to team leadership, role theory and agile team features Section 4.2.2 describes nine distinct leadership roles. Furthermore, Section 4.2.3 elaborates on the role transfer process by relating to maturity. Moreover, the Section refers to the factors influencing the role transfer labeled internal team environment and contextual factors. Section 4.2.5 builds a bridge between contextual factors and agile team features by describing the Agile Matching Theory.

I therefore integrate literature on self-managed teams into research on agile software development teams to provide a leadership model that is applicable unrelated to a specific framework such as Scrum.

4.1 Theory Selection

4.1.1 Team Leadership

Agile software development research refers to literature on team leadership from self-managed teams to explain leadership in agile teams [MDD10; SJ17]. Yet, the research that refers to team leadership to describe an agile team only cites a few studies from team leadership literature. I believe that there are more studies in team leadership literature that help us explain agile teams. In the following I refer to the concepts that contribute to understanding leadership in agile teams and that will be used to explain our conceptual model.

Until today self-managed teams have been implemented in various settings reaching from coal mines [TB51], to assembly lines [MS87], to software projects [MDD10]. While self-managed teams have been studied across various contexts, one recurring research topic relates to the role of leadership in a team that is by definition self-managed. Researchers have provided different leadership concepts to answer the research question.

The concepts differentiate in who is considered a leader, and how and in which way leadership exerts influence. Some concepts describe that the team leader has disciplinary power over the team [DW03; MS87], while others describe that the leader is a peer to the team [Bar93]. Traditional views on team leadership suggest that a formal leader exercises power and authority over the team, thus, is considered to be a position within hierarchy [DW03; MS87]. For example, this can be a stage setter who sets the overall direction, and designs the team constellation and its context [HH02a]. Latest research on team leadership suggests that leadership is unrelated to a formal position and that anyone with the right set of skills may take over a leadership role in a team [MDD10]. Moreover, some concepts relate to the idea that the team may take over leadership activities that were formally done by one dedicated leader [DW03; MDD09].

I believe that the different concepts from team leadership literature help us understand the Scrum Master role, and I will therefore elaborate on them in detail in the following.

Manz and Sims [MS87] describe an un-leader who displays super-leadership [Cha91] which empowers others to lead themselves. Manz and Sims explain:

> "Our position is that true leadership comes mainly from within a person, not from outside. At its best, external leadership provides a spark and supports the flame of the true inner leadership that dwells within each person. At its worst, it disrupts this internal process, causing damage to the person and the constituencies he or she serves." [**p.18**; Cha91]

Manz and Sims coin the term super leadership. The essence of this concept is that the formal leader disenchants the capability of the people surrounding her or him. The leader's inner balance is based on strong self-leadership and high moral standards. Furthermore, the leadership behavior focuses on four common goals: (1) creating environments that support positive attitudes, (2) empowering employees to set personal goals, (3) encouraging observation and feedback amongst subordinates, (4) stimulate teams to support and motivate each other. The authors state that the dedicated leader becomes obsolete over time and suggest six encouraging behaviors that enable a team to lead itself: self-goal setting, self-criticism, self-reinforcement, self-expectation, rehearsal and observation/evaluation [MS87].

Besides the formal leader who supports the team to lead itself, Manz and Sims [MS87] suggest a facilitator within the team. The role is an additional team member and thus considered to be a peer. The facilitator helps the team to organize itself and to coordinate job assignments. Furthermore, the role makes sure that the team has access to all materials needed for conducting the work.

Another study on self-managed teams points at the emergence of destructive team values in the absence of a formal leader [Bar93]. A longitudinal observation of self-managed teams revealed that some team members would overrule other team members and created a toxic team atmosphere. Therefore, the team voluntarily agreed upon implementing a facilitator who served as a neutral observer and mediator among different individuals and subgroups. This person served as a peer to the team members and helped the team to have cooperative discussions. [Bar93]

This role was responsible to coordinate material with other departments, handle requests of the formal group leader on behalf of the team, deliver meetings and track production errors. Furthermore, the hiring process for the new facilitator was as follows: firstly, the team nominated potential facilitators. Secondly, the team and the group leader would interview the person. Thirdly, they would decide together whom to chose. Moreover, the role was linked to an increase in salary but not in hierarchical position. [Bar93]

Druskat and Wheeler [DW03] examine a leader who is responsible for the performance of a self-managed team. They discover that the leader is a boundary spanner who serves as a mediator between the organisation and the team. The authors identify 11 leadership activities which they cluster into four different broader categories. (1) building political awareness and team trust, (2) seeking information from managers, experts and peers while diagnosing team behavior and problems, (3) obtaining external support and influencing team behavior, (4) empowering teams by delegating decisions and coaching teams. Their conclusion suggests that the dedicated leader should hand over the leadership activities to the team over time. By doing so, the committed leader releases capacity and thus can take care of more teams.[DW03]

While the boundary spanner enumerates coaching as one facet of a team leader, other authors elaborate on team coaching indepth [HW05]. Coaching refers to helping a team to help itself develop further. This approach implies to observe group behavior and provide feedback on what impedes teamwork. Furthermore, the coach helps the team to critically reflect upon changing team behavior and discussing on patterns for improvement. The team must be open to coaching and therefore, the coach must carefully pick the right timing to do coaching [KGM+96]. The team coach should use specific meetings within which the team and the coach review the team's purpose, current progress and matters for improvement.

Table 4.1: Leadership Roles Empowering Self-Managed Teams

Role	**Description**
Un-Leader [MS87]	A leader who empowers a team to lead itself.
Facilitator [Bar93]	A neutral person in the group who empowers the team to develop a supportive team climate.
Stage Setter [HH02a]	A leader who sets goals and provides the framework within which teams cooperate towards a common goal.
Boundary Spanner [DW03]	A team leader who mediates between the organisational level and the team.
Team Coach [HW05]	A team leader who helps the team to develop itself further.

4.1.2 Role Theory

Role theory can be considered from a static and dynamic perspective. The static lens describes a set of roles that are shared within a team or that are embraced by one individual while dynamic role theory describes that the distribution of roles varies depending on the maturity of the team. In addition, roles are either shared between self-organising team members or a set of different roles are embodied within one individual leader.

Quinn [Qui88] uses the competing values framework to describe how a manager embodies two paradoxical roles in order to succeed. A leader must be task oriented to achieve certain goals on the one hand, and must be people oriented by taking care of the employees on the other hand. These two roles constantly challenge the inner balance of a manager. Quinn et al. [QFTM90] build on this idea and suggest eight different leadership roles (Mentor, Facilitator, Innovator, Broker, Producer, Director, Coordinator, Monitor) located in four competing quadrants (relating people, leading change, producing results, managing processes).

Zafft, Adams, and Matkin [ZAM09] examine these competing roles for self-organising teams in relation to team performance. They conclude that teams that share self-organising roles more broadly show higher performance than teams in which leadership roles are rather centred on one individual team member. Yet, they measured distribution of leadership roles at the end of the project and acknowledge that leadership roles may be distributed differently depending on the project stage. Yang [Yan96] adopts a dynamic perspective and finds that those eight roles vary according to the development stage of a team. She suggests that the roles of monitoring and coordinating may be replaced by team norms after a while.

Hoda, Noble and Marshall [HNM12b] discover six different self-organising roles (Mentor, Coordinator, Translator, Champion, Promoter, and Terminator) and describe that mature teams tend to share those roles more broadly within the team than immature teams. While immature teams rather stick to formal role keepers, the roles can be transferred to other team members with the fitting skill set in more mature teams. Barke and Prechelt [BP19] refer to the importance of role clarity among team members, e.g. shared expectations, for taking on roles. An insufficient understanding of roles in teams leads to role conflicts.

In this thesis, I build a bridge between one dedicated role keeper and role sharing while taking maturity and role conflicts into account.

4.1.3 Maturity

Team literature research differentiates between static and dynamic teamwork models. While a static perspective refers to teams that are stable and have successfully reached a constant mature stage, the dynamic approach assumes that a team undergoes different maturity stages. This study refers to dynamic teamwork models since I believe it helps us to explain the changing Scrum Master.

There are several dynamic teamwork models. For example, Gersick [Ger89] and Ginnett [Gin19] suggest the first meeting to be decisive for the further collaboration of a team. Still others suggest an iterative approach wherein the team continuously revolves between different phases [MMZ01]. Agile teams are linked to the forming-storming-norming-performing model by Tuckman [Tuc65].

An agile team transfers through the different maturity stages until it evolves into a truly agile team [GTF17]. Therefore, developers practice the agile way of working differently over time. Through situational leadership theory, Gren et al. [GTF17] suggest that a rather immature team needs structure and order and that it will not accept agile leadership behavior whereas a mature team is able to self-organise better which allows the Scrum Master to step back.

The *forming phase* suggests that team members focus on a leader who sets ground rules for further cooperation [Tuc65]. Team members are insecure about how to behave, and they search for opportunities to observe expected behavior. In this stage, agile teams are suggested to be more open towards leadership that is centred on one person [GGJ19]. The *storming phase* often involves role conflicts due to a lack in unity and security [Tuc65]. Performance often drops in this stage [KS15]. The *norming phase* helps teams to increasingly understand and agree on how to work in an agile way [GGJ19] and to build a shared understanding of roles and responsibilities [NW99]. Team performance increases in this phase [KS15]. The *performing stage* describes a high performing team in which the team members play roles flexibly according to the situation [Tuc65].

Furthermore, the Retrospective is a tool that helps the team in developing maturity [GGJ19].

Based on previous research, I differentiate between mature and immature teams and suggest that leadership activities are rather centred on the dedicated Scrum Master in an early team stage while in mature teams developers also take on leadership activities.

4.1.4 Agile Team Features

Agile teams are described along the core features iterative learning, cross-functionality and self-organisation [Con09; TN86] each containing several sub-features. To explain the characteristics, studies often refer to theories from psychology, social-psychology or organisational studies.

The first core feature is *team learning*. Team learning describes the learning behavior of agile teams by delivering feedback to each other, sharing information openly, providing support to each other, and learning from failures and experiments [Edm99; TN86]. Team learning involves the two sub-features shared mental models [LWW01] and psychological safety [Edm99]. Shared mental models means that members can place communication of fellows into the appropriate context and thus receive and interpret communication similarly. Psychological safety describes an environment within which team members feel safe to talk openly with each other. Talking openly results in shared learning experiences.

The second core characteristic is *cross-functionality*. Nonaka and Takeuchi [TN86] emphasize that cross-functional project teams are not built by experts but rather by people that have access to all kind of different knowledge of different hierarchical layers while they learn on the job. Even though the team works cross-functionally, individuals still have specializations and constantly aim to find a balance between cross-functionality and specialization [HNM12a]. Cross-fertilization describes that agile teams incorporate different specializations regarding cognition, behavior and personalities [TN86]. Even though there may be specialized skills within a team, all team members must feel responsible since agile teams are team oriented [MDD10].

The third core characteristic is *self-organisation* which implies that teams divide tasks among each other based on needs, fit and self-assignment [HNM12b], make fast decisions towards a common goal [CH01; Hig09; TN86] and monitor themselves [MDD10]. While agile teams own a high level of freedom, they feel responsible to deliver value [HNM12a].

I claim that the core goal of agile leadership is to enable the Developers to evolve the agile team features.

4.1.5 Contextual Factors

Despite self-organisation, teams still commit to local rules [VW09]. Development teams are embedded into the organisational context which influences their capability to work in an agile way [HB13].

Magpili and Pazos [MP18] provide a literature review on contextual factors influencing self-managed teams. They enumerate the following success factors: corporate culture showing facets of a learning organisation [GEG08], collectivistic national culture [NLB99], clarity on organisational goals [Wag01; Wag97] including high feedback on performance [GCD+16], team-oriented and flat organisational structure [BBCL16], all kind of training, access to resources depending on the team's need [HNM12b; ICLG02] and team rewards and social incentives [Wag97]. For a detailed description, please refer to Magpili and Pazos [MP18].

Management is predominantly in charge of setting the context, and thus, the capability to self-organise work depends heavily upon management support [HB13; HH02a; TN86]. Instead of focusing on individuals a manager is expected to set the context within which teams can work self-organised and take over responsibility [CH01; DW03; TN86]. Setting the boundary conditions involves providing money, moral support, challenging requirements and a compelling vision [TN86].

Additionally a manager is described to be a guide who inspires the team [CH01] and does not rely on formal authority [Bon10]. Moreover, the manager does not provide solutions to the team but is open towards an inspect and adapt approach by the Developers [Hod13; TN86]. This requires that the manager demonstrates a high level of trust in people to do a good job [Bec00; CH01; HH02b; Hod13; MAD12].

Despite the influence of organisational context on agile teams scarce research takes the context into consideration when examining agile team behavior [CH01; GBS+16; HN17]. I explore in which way bureaucratic features influence Developers to take on leadership roles of one dedicated Scrum Master.

4.2 Theory Building

This thesis aims at exploring changing leadership in agile teams using the example of a Scrum Master. I combine the above-described literature on team leadership, role theory, maturity and contextual factors with research from agile software development to explain changing leadership in agile teams.

4.2.1 Perspective on Leadership

Before I outline my theory on changing leadership in agile teams I specify my leadership perspective on the Scrum Master in the following.

The dedicated Scrum Master is considered to be a leadership enabler who follows the core goal to enable Developers to work in an agile way and to take on leadership roles themselves [SSne]. The 'agile way' is described by the agile team features iterative learning, cross-functionality and self-organisation. Therefore, the Scrum Master does not aim at a specific quantity or quality of output but supports a team in developing the agile team features [Bäc19]. A team that embraces the agile way of working is a high-performing and, thus, an effective team [GTF17]. However, I do not denote that certain leadership responsibilities remain with one leader but that the leadership roles shift from one person to another depending on the situation [MDK09].

Applying role theory the Scrum Master is divided into a set of nine distinct leadership roles. I use team leadership to explain that the nine roles can be played by one dedicated Scrum Master but also by the team. Therefore, I suggest that anyone in the team may take on the Scrum Master behavior [MDK09; SJ17]. The role sharing of the nine leadership roles is summarized in the 9-Factor Theory.

I refer to maturity and suggest that the Scrum Master role is located within one dedicated leader in an immature team, but shared between one individual and the Developers in a more mature team. Consequently, I argue that leadership is a fluid concept that evolves over time while the team matures. Yet, I suggest that a team will only take on leadership roles if it is embedded in a supportive organisational environment. Thus, I take the context within which leadership emerges into account [CLP10].

The conceptual model is described as follows: Firstly, I will describe the nine different roles of a Scrum Master by referring to role theory and team leadership literature. Secondly, I describe the role transfer from the Scrum Master to the Developers by referring to the maturity model by Tuckman [Tuc65]. Thirdly, I elaborate on the enablers of the role transfer on internal team environment by referring to agile team features and on hindrances of the role transfer on organisational level by referring to contextual factors. These enablers result in the Agile Matching Theory which implies a required match between the internal team environment and the contextual factors for the materialization of the role transfer.

4.2.2 9 Leadership Roles

I apply role theory and propose a set of nine leadership roles of the Scrum Master which are gradually transferred to the Developers. I summarize this proposition as the *9-Factor Theory*.

Each role aims at stimulating particular agile team features. In the following I will describe the roles in detail. I structure each role description in the following way: Firstly, I refer to features of an agile team as specified in Section 4.1.4. Secondly, I link each role description to existing research on self-managed and agile teams, while also discovering new roles, and outline how each role helps the team to develop the respective agile team features.

4.2.2.1 Method Champion

Agile methods support teams to work well together. Even though the Scrum framework may help a team to work in an agile way, Scrum is not always beneficial in every team setting. Therefore, the team needs support in finding the appropriate method for a specific context [MAD12; MDK09].

I propose the Method Champion who is responsible for the implementation and application of the Scrum Method. The role not only focuses on the Scrum Method but also brings other new methods to the team that foster collaboration. On the one hand the role keeper adapts the Scrum method to the specific team context, on the other hand he or she also realizes when the Scrum approach is not the most fitting one regarding the context.

4.2.2.2 Disciplinizer on Equal Terms

Members and managers of agile teams are peers and treat each other unrelated to title or position [HNM12b]. This encourages developers to talk on equal terms with each other and to suggest their own ideas. While receiving a high level of freedom, individuals show commitment to achieve the Sprint goal [HNM12a]. Developers focus on their respective tasks and reveal discipline [SMD11].

I suggest the Disciplinizer on Equal Terms who helps the team to focus on respective tasks and stops developers from multi-tasking. It is important to not force the team to focus by referring to hierarchical power, e.g. by asking for status reports, since the facilitator is a peer to the team members [Bar93; MS87; TN86]. The role keeper communicates on equal terms with the team members while supporting the team to create a hierarchical-free space.

4.2.2.3 Team Coach

Developers learn continuously while developing product features and team members provide feedback to each other while they mature [TN86].

Team coaching is described to be an enabler for self-managing teams [Wag01] but also for agile teams [Bäc19; MCDE15; PGN14]. The coach helps developers to constantly develop themselves further by setting triggers, such as asking the right questions at the right time [KGM+96], and by encouraging for self-criticism, observation and feedback delivery [MS87]. The Coach helps to identify what is missing in the team and provides feedback on how to develop further. Particularly an immature team that aims to become self-organised needs a supportive coach [MP18; Wag97].

4.2.2.4 Change Agent

Developers continuously challenge the status quo and find new ways of getting work done [TN86]. Furthermore, implementing agile methods is a change project [HN17]. Team members who had not been working in an agile manner before, need to learn what it means to work in an agile way. Particularly teams that are embedded in a bureaucratic environment are used to a rather traditional way of working and may have to change habits.

Change Agents are critical roles during the agile team transition [PGN14]. The Change Agent convinces teams of the agile way of working and serves as a role model. The developers learn the agile manner by observing the Change Agent and imitating this behavior. Therefore, it is especially important at the beginning when implementing agile teams. While the team matures individuals become more and more convinced of the agile manner, hence the Change Agent is played to a lesser extend.

4.2.2.5 Helicopter

Developers tackle complex projects, which implies that one individual cannot understand how all matters of the project are linked to each other. Therefore, they need to build shared mental models [LWW01] of each others competences and skills to understand in which way tasks are interdependent. This allows them to plan and divide their work among each other according to the team needs and personal fit [HNM12a]. In order to be able to self-organise and make fast high quality decisions towards a common goal [CH01; Hig09; TN86], team members need to develop a cross-functional understanding of the overall project.

Manz and Sims [MS87] refer to a facilitator who supports the team to organize itself and coordinate job assignments. I suggest the role of a Helicopter who empowers the Developers to understand how to work cross-functionally as a team and how different issues are interdependent. Team members learn who possesses which knowledge, whom to approach and whom they should hand over a task if they have finished it and need someone else to continue working on it. Moreover, they can plan their work together and feel responsible for the product as a whole.

4.2.2.6 Moderator

Agile teams work cross-functionally which implies varying cognition, behavior and personalities [TN86]. This may involve different domain languages, working habits and interests regarding technical issues and may cause a lack in understanding each other. Therefore, some authors take the capability of a cross-functional team to self-manage work into question [CB97; Yuk13]. Furthermore, self-managed teams tend to establish destructive team values [Bar93]. In order to work cross-functionally, developers need to understand each other's skills, capabilities and working habits [MDK09].

I refer to the idea of a facilitator of discussions [Bar93] and suggest the role of a Moderator who helps the team to build a bridge between different domains. The role keeper mediates between individuals, e.g. translating domain language, and between different point of views which may refer to technical issues but also to personal matters. Therefore, the Developers learn to understand each other properly and thus work cross-functionally as one team while they mature.

4.2.2.7 Networker

Even though an agile team is cross-functional, it does not contain all expertise needed within one team but has access to all competences needed [TN86]. The agile team connects to external parties depending on specific Sprint goals.

Former studies found that the team leader of self-managed teams serves as a boundary spanner between the organisation and the team [DW03] and makes sure that the team has access to all required materials and knowledge from the organisation [Bar93; Cha91]. Based on these findings I propose the Networker. This role helps developers to connect with relevant stakeholders outside the team. However, unlike Druskat and Wheeler I do not suggest that the Networker should be assigned to one individual but anyone within the team can provide the personal Network and can approach experts outside of the team - unrelated to status, title and position - within an organisation.

4.2.2.8 Knowledge Enabler

Agile projects often tackle new challenges which require an inspect and adapt approach and learning on the job [TN86]. Therefore, developers learn iteratively [TN86] and apply team learning [Edm99]. Since developers balance continuous learning and iteration pressure they take time for learning but also realize when to stop acquiring new knowledge on a topic [HNM12a].

I suggest the Knowledge Enabler who supports and reminds the Developers of team learning [Edm99] and of transparent knowledge sharing. For example, the Knowledge Enabler encourages learning sessions for sharing knowledge openly with colleagues and for learning from each other. Also the role helps teams who are used to a rather traditional way of learning, e.g. learn with the help of a book, to a more unconventional way of learning, e.g. using github or attending meet-ups.

4.2.2.9 Protector

Developers have a high level of freedom to organize their work [HNM12a] and monitor themselves [MDD10]. They need interference-free space to focus on their respective Sprint goals [SMD11]. Therefore, the team needs a Protector who shields the Developers from disrupting and unreasonable requests [HM16; SMD11]. Moreover, I suggest that the Protector creates hierarchical free space in which developers can take on the other leadership roles. I label this space *leadership gap* and will describe it further in Section 4.2.3.

4.2.2.10 Summary

This thesis aims to integrate team leadership literature into research on leadership in agile teams. I have broadened the suggestion by Moe et al. [MDD10] and investigate team leadership in agile teams by referring to more research from team leadership literature. While I explain some of the roles by referring to team leadership not all of the roles can be described by already existing concepts from team leadership literature. In the following I recapitulate the link between each leadership role and existing body of knowledge and which further roles I suggest.

I link the Networker to the Boundary Spanner [DW03] and facilitator [Bar93; MS87], the Coach to several suggestions on coaching behavior [DW03; Wag01], the Disciplinizer on Equal Terms partly to the idea of a person serving as a peer to the team [Bar93; TN86], the Helicopter partly to the facilitator [MS87] and the Moderator to the idea of a mediator and neutral observer [Bar93].

I identify further roles related to the Method Champion, Change Agent, Knowledge Enabler and Protector, which have, up to my knowledge, not yet been discovered in research related to self-managed teams. However, research on leadership in agile software development teams has already referred to the Change Agent [Bäc19; PGN14], the Protector [HM16] and implicitly to the Method Champion [MAD12; MDK09]. I suggest the Knowledge Enabler as a new role.

I therefore propose the following nine leadership roles:

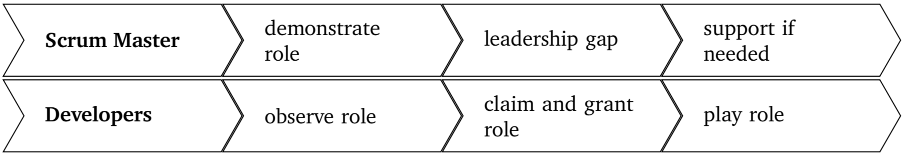
1. Method Champion: Organises meetings and get-togethers, teaches the method, supports formulating tasks and setting goals, visualises information, and discusses how to adapt the method during the Retrospective.

2. Disciplinizer on Equal Terms: Supports the team to keep to the rules, ensures that the team focuses on relevant topics and makes sure that team members attend the meetings. Discipline is accomplished via communication on a par.
3. Coach: Observes team members and uncovers which kind of behaviour is missing in a team to improve teamwork, provides feedback, and helps teams to find out what they wish to change and how to do so.
4. Change Agent: Serves as a role model, changes habits, and convinces newly established project teams of the agile way of working.
5. Helicopter: Possesses the ability to see the bigger picture, to know who possesses the right skill for a certain task, to include relevant stakeholders and to structure work.
6. Moderator: Moderates all kind of meetings and builds a bridge between perspectives and domains.
7. Networker: Connects the team with relevant stakeholders from within and outside the organisation.
8. Knowledge Enabler: Realises which kind of knowledge the team needs, supports team members to acquire that knowledge and promotes iterative learning.
9. Protector: Shelters teams from inappropriate requests from the Product Owner, managers, disciplinary leaders and other departments.

4.2.3 The Role Transfer Process

Several authors refer to a changing Scrum Master in such a way that the primary objective of a Scrum Master is to transfer leadership activities to the Developers over time [MDD10; SJ17]. Furthermore, a number of authors claim that leaders of agile teams need to demonstrate more monitoring at early stage, but can delegate tasks at a later stage of team development [Kak17; LLDL15]. Gren et al. [GTF17] states that depending on the maturity level of a team, team members tend to adapt agile working differently and

Figure 4.1: The three steps of the role transfer process.



that leadership should adapt accordingly. Yet, up to our knowledge there is no indepth research that describes how the leadership role changes and in which way leadership activities are transferred to the Developers.

This thesis builds on former studies [GGJ19; MDD10; SJ17] by referring to maturity to explain the changing Scrum Master. I use a dynamic leadership model and suggest that the nine roles of the Scrum Master are shared differently depending on the maturity stage of a team. While the above-mentioned nine leadership roles are rather centred on one dedicated Scrum Master in an immature team, the roles are rather played by the Developers in a more mature team.

I describe the changing Scrum Master along three consecutive steps which I label the *role transfer process* (figure 4.1). The core element of the role transfer process is the leadership gap which empowers the Developers to take on leadership roles themselves. The consecutive steps are as follows:

Firstly, the Scrum Master demonstrates the leadership roles while the team observes the behavior and learns its meaning. Secondly, the Scrum Master steadily steps back from leadership responsibilities and therefore provides a leadership gap. The leadership gap describes a hierarchical free space within which any team member is allowed to take on a leadership role if it makes sense in a given situation. The Developers jump right into the leadership gap by starting to claim leadership roles while the respective colleagues grant the person to be the new role keeper. In the last step, the Scrum Master performs leadership roles only where still needed whilst the Developers play most of the Scrum Master roles themselves.

After a while most of the roles are played by whoever feels committed, capable and responsible to take over the role in any given moment. Thus, I claim that leadership roles can be shared between a formally appointed Scrum Master and the Developers depending on the maturity level.

4.2.4 Factors influencing the Role Transfer

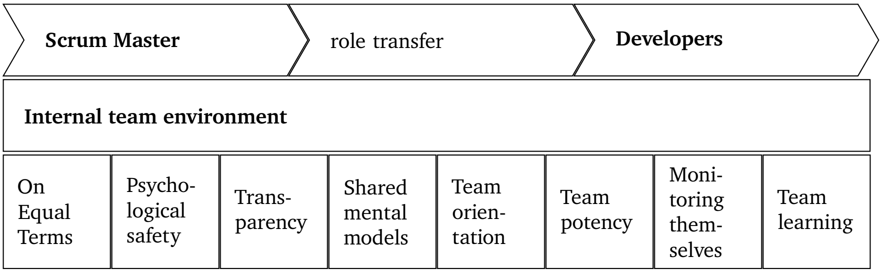
There are enablers and hindrances that influence the role transfer process. I refer to the internal team environment as an enabling factor and to the contextual factors as a hindering aspect in a rather bureaucratic setting.

4.2.4.1 Internal Team Environment

I propose that the role transfer process is fostered by the *internal team environment* which is composed of the following features: being on equal terms, psychological safety [Edm99], transparency, shared mental models [LWW01], team orientation [MDD10], team potency [GYCS93] and monitoring themselves [MDD10]. The following enumeration describes how each of the features contributes to the role transfer.

- Being on equal terms creates an atmosphere within which everyone feels equally free to take on leadership roles unrelated to hierarchical position.
- Psychological safety [Edm99] empowers developers to take on leadership activities for the first time even though they might fear to fail without having previous experience in performing the role.
- Transparency empowers teams to build a shared understanding of the roles.
- Shared mental models [LWW01] imply that the Developers have the same meaning of the different roles in mind and agree that anyone in the team can take on the respective role depending on the situation (claiming and granting).
- Team orientation is important to act according to the shared purpose of the team [MDD10], and therefore to know which leadership roles team members should take on to achieve the common goal.

Figure 4.2: Integrative model of the role transfer process.



- Team potency implies that the team believes in its capabilities [GYCS93] and therefore, to feel comfortable to take on the leadership roles.
- Monitoring themselves [MDD10] empowers the Developers to feel a sense of responsibility to take on leadership roles instead of externalizing responsibility to a formal leader.
- Team Learning [Edm99] allows the team to continuously develop themselves further regarding playing the divers roles.

4.2.4.2 Contextual Factors

Often agile teams are implemented as isolated clusters in rather traditional development companies [GL20; HB13] which can be classified as bureaucratic organisations. The core contextual factors of that company type are strong hierarchy, functional departments, rigid standards and processes and written rules [Web09].

Strong hierarchy [GBS+16; HB13; HM16; MAD12; NMM05], specialist culture [HM16; MDD09], rigid structure [NMM05] and inflexible processes [BT05; CCWP11; NMM05] were identified to hinder the agile way of working in former studies. The reasoning behind it is that strong hierarchy with a focus on decision-making by management contradicts teams that take on leadership roles [MAD12], rigid planning opposes iterative team learning [BT05] and a functional structure with rigid processes and specialists limits cross-functional collaboration of developers [NMM05].

Wagemann put it this way:

> These organisations have long histories of hierarchical decision making cemented with a work ethic based on individual achievement. Given this culture and context, team members will balk at the idea of relying on one another to get work done. [**p.50**; Wag97]

Not only team members hesitate to take on leadership but also management can destroy the self-organising nature easily by making decisions on behalf of the team or creating dependencies on other departments by implementing and sustaining rigid processes (e.g. [CH01; HB13; HM16; HN17; MAD12]).

While team coaching is said to be unsuccessful if the contextual and structural environment is not supportive towards team performance [HW05; Wag97] teams that face a supportive context are suggested to even develop coaching capabilities themselves while they mature [LBT95]. I broaden this perspective to the other leadership roles besides the Coach and suggest that contextual factors influence team members in taking on the divers leadership activities. My research focuses on agile teams that are embedded in bureaucratic companies which currently undergo an agile transformation but are not yet agile companies.

Since a bureaucratic organisation contradicts the agile way of working on team level [HB13; NMM05; SHW19b], I consider the contextual factors as hindering the role transfer. I can assume that developers struggle to take over the leadership roles if the necessary preconditions on organisational level are absent.

4.2.5 The Agile Matching Theory

Based on the role transfer process, the internal team environment and the contextual factors I build an integrative theory which I label the *Agile*

Matching Theory. This theory implies that the organisational context and the desired agile team features need to match for the role transfer to materialize.

The reasoning behind it is that people behave according to the context within which they operate [CH01]. Developers are more keen to take on leadership roles if they face the necessary preconditions on the organisational level. Team leadership can be supported with appropriate boundary conditions since agile teams nurture in an environment that supports the agile way of working. For example, a learning organisation [GEG08] in which developers feel safe to make mistakes and learn from each other increases team learning and psychological safety. Consequently developers may take on the role of a Knowledge Enabler.

Vice verse if the organisational level rather embodies bureaucratic organisational features, also an internal team environment will rather contain bureaucratic team features than the desired agile team features, and thus, the role transfer less likely occurs.

Moreover, the Scrum Master is torn in between fulfilling expectations deriving from a bureaucratic organisational culture and structure and expectations deriving from the agile manner. The agile way of working is based on shared values, believes and a common goal, a normative approach while bureaucratic organisations are based on written rules, standards, processes. This leads to a clash of expectations linked to the agile way of working (e.g. self-monitoring or handing over leadership roles) and to the bureaucratic culture and structure (e.g. traditional career models, key performance index (KPI) or reporting). Feeling the need to fulfil contradicting expectations leads to role conflicts (see also Noll et al. [NRBB17] and Stray, Sjøberg, and Dybå [SSD16]).

I speculate that if the Scrum Master decides to fulfill expectations obtained from a bureaucratic way of thinking, she or he is not capable of playing agile roles like the Protector and providing a leadership gap. Consequently, the team's opportunity for taking on leadership roles is limited.

Table 4.2 shows the contradiction by displaying features of bureaucratic organisations on the left and desired agile team features on the right. The results presented in Section 5.2.3.4 provides a detailed description of how organisational features influence agile team features.

Table 4.2: Organisational Factors and Internal Team Environment

Bureaucratic organisation	**Internal Team Environment**
(1) Hierarchical Culture [HM16; MAD12; NMM05]	Monitoring Themselves [MDD10] Transparency On Equal Terms Psychological Safety [Edm99] Team Potency [GYCS93]
(2) Specialist Culture [HM16; MDD09]	Shared Mental Models [LWW01] Transparency Team Learning [Edm99] Team Orientation [MDD10] On Equal Terms
(3) Functionally Departmentalised [Web09]	Shared Mental Models [LWW01] Team Learning [Edm99] Team Orientation [MDD10]

4.2.6 Summary of the Conceptual Model

I add to research in agile software development by integrating team leadership, role theory, maturity and contextual factors from research on self-managed teams to research on leadership in agile teams. I contribute to empirical studies on agile teams by proposing a conceptual model on the changing Scrum Master. In the following I summarize my theoretical model:

- The Scrum Master embodies a set of nine distinct leadership roles.
- In an immature team the nine leadership roles are rather played by one dedicated Scrum Master.
- While the team matures the Scrum Master transfers the roles to the Developers.
- The heart of the role transfer process involves a leadership gap provided by the Scrum Master and claiming and granting of the roles by the Developers.

- The dedicated Scrum Master does not become obsolete in a mature team. Some roles stick with the Scrum Master.
- A supportive internal team environment stimulates the role transfer.
- Bureaucratic culture and structure often lead to role conflicts and hinder the role transfer.
- The Agile Matching Theory suggests that the desired agile team features and the organisational context need to match in order for the role transfer to occur.

Chapter 5

Part I: A Grounded Theory Study

It is organized as follows: Section 5.1 explains the study design, including the research questions, the research methods, context, data collection procedure and sample. Section 5.1.6 portrays the data analysis by referring to examples derived from this study. Section 5.2 presents the results of the study which build the theory as described in Section 4.2. The findings are critically discussed in section 5.3. Finally, Section 5.4 refers to limitations of the Grounded Theory study and suggests topics for future research.

5.1 Study Design

5.1.1 Research Questions

The objective of the thesis was to explore the changing leadership using the example of a Scrum Master. I therefore formulate the following research questions for my Grounded Theory study:

1. Which roles does the Scrum Master play to support the team to work in an agile way? (RQ1)
2. In which way do developers take on the Scrum Master role overtime? (RQ2)
3. How are roles transferred from the Scrum Master to the Developers? (RQ3)
4. What is the underlying internal team environment required for the role transfer to occur? (RQ4)
5. How can the Scrum Master foster the internal team environment? (RQ5)
6. Which expectations on the Scrum Master limit the changing leadership role? (RQ6)
7. How do organisational culture and structure influence the role transfer? (RQ7)

5.1.2 Grounded Theory

Section 3.2 described that there is scarce research on leadership in agile teams. I chose Grounded Theory because this method is applied in research fields with scarce knowledge and it aims at generating new theory on social interaction between actors [GS17]. Furthermore, it is used to identify repetitive pattern of actions [Mye19].

Grounded Theory distinguishes most often between the perspective by Glaser [GS17] and by Strauss and Corbin [CS15]. While Strauss and Corbin claim that the researcher should focus on specific literature early in the research process, Glaser suggests that the researcher should read a wide range of literature while analysing data in iterative steps in order to remain open to the emergence of new theory [HC04].

Moreover, Grounded theory can be conducted by applying a positivist and constructivist view [BC07]. The positivist perspective claims that there is an objective reality, which is linked to the Glaserian point of view. The constructivist view aims to explain subjective reality of individuals that is embedded in the specific context and situation in a particular point in history [BC07]. The objective is to describe how individuals make sense of the ongoing transformation at a rather traditional development company. We aim to describe which kind of leadership individuals *believe* to be useful during an agile transformation. The constructivistic approach aims to reveal multiple perspectives on reality [Cha16]. Therefore, there may not only be one point of view reflected by the research participants. Moreover, participants may behave in a certain way due to social conventions and power relations [Cha16]. We search for underlying assumptions that foster specific behavior [Cha16].

While we apply the iterative approach of the data analysis method by Glaser and Strauss [GS17], we take a constructivist view when interpreting the data.

Grounded Theory aims at generating new theory from a repetitive comparison of emerging data. It does not intend to test existing theory. While the research initially contains a general research topic, the research question emerges during the research. Grounded Theory is an interpretative approach and the research question should be embedded in the specific context under study. [GS17]

Grounded Theory follows an iterative approach in which each step determines the next step to be taken during the research [GS17]. The different steps will be described during the following subsections while referring to data collection and analyses applied in part I. For a detailed description of each step please refer to Hoda, Noble, and Marshall [HNM12a].

5.1.3 Research Context

The research aimed at contributing to agile leadership in bureaucratic companies.

I conducted this study at Robert Bosch GmbH. The Robert Bosch GmbH employs more than 410,000 people in 60 different countries worldwide. The company history dates back to 1886. The conglomerate is active in four different business areas: mobility solutions, industrial technology, consumer goods as well as energy and building technology. Each business area consists of various subsidiaries and business divisions. Therefore, market conditions and subcultures vary wildly.

The group aims to develop from a rather bureaucratic company type to a more agile company. While the agile transformation had been a project by the headquarter until 2017, since 2018 each division has been responsible for its own agile transformation. While sometimes management decides top-down that they would like their teams to work in a more agile way, e.g. apply the Scrum framework, other project teams decide by themselves how they want to work.

Even though there exists a company-wide role description for a Scrum Master behaviour, there is neither an obligatory training nor a general rule regarding the Scrum Master. Each team can decide on its own how to train the team and Scrum Masters in the agile way of working. For example, they could book a training internally or externally of the company or not book a training at all.

5.1.4 Data Collection and Sample

Since I was an industrial PhD student at the respective company, I had direct access to the field and collected data on the topic between June 2017 and November 2018. I identified Scrum practitioners either via my personal network or via intranet and first contacted them by email. Interviews were conducted according to availability and willingness to take part.

Data was collected from 11 business divisions which have slightly different subcultures. While several divisions are already in the forefront of its agile transformation, others have started its journey just recently.

Most divisions were active in the automotive industry, while others produced domestic appliances and gardening tools.

Interviewed teams stated that they apply the Scrum framework mostly in modified form, e.g. w.r.t. the regularity of Scrum meetings. All Scrum Masters were without disciplinary power, responsible for the Scrum process and in charge of team development. Most practitioners of the company call the Scrum Master role *Agile Master*, indicating that this role should adapt to the specific team, rather than sticking to the Scrum approach by the book. Thus, I consider the sample fitting to examine maturity and the changing Scrum Master role.

Three rounds of collecting interviews were conducted. I conducted interviews with practitioners in case the interviewees were available and willing to contribute. The first round of interviews involved 22 individuals from 10 different sub-divisions. Among other roles, interviewees were software developers, Scrum Masters or coaches on organisational level. The purpose of the first round was to identify a focus topic for this research project on leadership in agile teams. Data is based on unstructured interviews and is not included in the results report in Section 5.2.

The second and third round of interviews were used for the qualitative data analysis. The data set includes 22 Scrum Masters, 8 Product Owners and 23 Developers from 14 software development and 15 non-software project teams. The size of teams ranged from 5 to 12 members and often included diverse nationalities. Since the age of teams stretched from three months up to three years, I expected the maturity of teams to vary. The second round of interviews involved predominantly Scrum Masters, while the third round of interviews approached teams as a whole (the Scrum Master, several Developers and the Product Owner). I report on the results of the 53 interviews in Section 5.2.

Observations: It is recommended to supplement semi-structured interviews by observations to understand the context of the interviews [AHK08]. I observed Scrum events, such as the daily stand-up, review, planning and retro, of sixteen Scrum teams from four different sub-companies. I observed ten of the teams between an hour and a whole day and six of the teams over a period of several months. While observing, I made notes and asked clarifying questions afterwards if necessary.

Feedback: I collected feedback from Scrum practitioners by presenting preliminary findings to observed teams and by discussing upon the results. Moreover, preliminary concepts were presented on two internal open space formats, two interactive workshops, one presentation and several one-to-one conversations with practitioners from the company. During discussions with Scrum practitioners I made notes and the participants of the workshops send their group work results to me. This material was used to refine and strengthen the developed concepts and add to validity.

Memos: During our data collection I wrote memos. I wrote down emerging questions, new ideas and relationships between codes and categories while collecting, analysing and discussing data with research colleagues from the institute and my industrial supervisor. Moreover, the memos stored ideas about links between the data and relevant literature.

5.1.5 Data Collection Procedure

Grounded Theory requires a general up-front research topic [HNM12a] which was leadership in agile teams. Initially I merely conducted a minor up-front literature study since the research problem should be allowed to emerge during the study while collecting data at the research field [GS17]. The research problem should emerge from a challenge that practitioners face [GS17].

Besides conducting qualitative interviews which were audio-taped and transcribed, I observed Scrum events, such as the daily stand-up, review, planning and retro, of various agile teams. Grounded Theory suggests theoretical sampling [GS17] in which each step determines the next step to be taken during the research based on previous results. In total, I conducted three rounds of collecting interviews while continuously comparing my findings with current literature.

First round of interviews: I first focused on leadership in agile teams in general to find out which specific research problem would be of interest. Since the purpose of the first round of interviews was to get an overview on relevant topics, this data is not included in the results section. However, I will briefly summarize the findings in the following since it lead to the focus topic of this research project. During the first round of unstructured interviews with practitioners from various business divisions and diverse organisational roles, I realized that there was a lack of clarity regarding the leadership role of one dedicated Scrum Master. Practitioners agreed that agile teams were expected to be self-organising, yet, how the Scrum Master was supposed to enable the team to do so was not entirely clear.

Comparing the data with current literature on the leadership role of a Scrum Master I found that several authors claim that the Scrum Master changes while the team matures and that parts of the Scrum Master role are transferred to the Developers over time [CH01; MDD10; SJ17]. Yet, up to our knowledge, this claim has never been empirically investigated. I decided to broaden past research and explore how the Scrum Master changes.

Second round of interviews: I conducted a second set of interviews mainly interviewing Scrum Masters besides a minority of team members. A semi-structured questionnaire focusing on the leadership role of a Scrum Master guided the interviews and allowed further questions if the answer of an interviewee seemed to offer more insights. Interviewees were asked to describe how they supported Developers and what they had learned since they had started to play the Scrum Master role. The guiding questions are available online [SHW18].

I identified nine leadership roles which the Scrum Master performs. Yet, the interviews revealed that not only the Scrum Master played leadership roles but that developers tended to take on some of the Scrum Master responsibilities, as well. Thus, I developed a second questionnaire that focused on sharing leadership responsibilities between the Scrum Master and the Developers with a specific interest in how the sharing of leadership roles had evolved over time.

Third round of interviews: The third round of interviews approached entire teams and viewed the Scrum Master role from three different angles: the Product Owner, the Scrum Master and the Developers. This helped us in understanding how the role evolved while the team matured and how leadership responsibilities were shared among developers and the dedicated Scrum Master. I became to realize that Scrum teams without a Scrum Master struggled with working in an agile way.

5.1.6 Data Analysis

I coded the collected data by applying Glaser's Grounded Theory [GS17]. While Glaser and Strauss [GS17] have a positivist view, we had a constructivist perspective for interpretation of data [Cha16]. I openly coded transcripts sentence-by-sentence. The content of the interviews was constantly compared within the same interview and across interviews. I aligned codes that appeared to be alike to one concept. I constantly reflected and compared emerging concepts critically. I aligned concepts if they appeared to be alike to build categories. Observations helped us to place the content of the interviews into context.

During data analysis we made a few sketches, quick power-point presentations and used sticky notes on whiteboards demonstrating preliminary results. We critically discussed the concepts in our researcher group and with practitioners. This led to a refinement of the concepts.

Example:

Quote: *If I am convinced of something I bring it into the world. [...] simply I bring agility with me, always on the basis of a mind-set [...], so that people can understand why it makes sense to work in that way.* *(AM)*

Key Point: *serves as role model to others*

Codes: *role model, agile mind-set, change team*

Concept: *Change Agent*

Category: *Leadership Role*

This example reveals that the Scrum Master convinced some team members of the agile manner by acting as a role model. The codes merged to the concept *Change Agent*. Overall I identified nine different concepts during the iterative data analysis. The core category is *leadership role*. One leadership role is a set of performed, connected activities that influences individuals with the objective to lead one another to the achievement of team or organisational goals or both. I chose the term *role* because it expresses a set of connected activities that is unrelated to a position or title. A leadership role can be either performed by a Developer or a Scrum Master. A leadership role emerges context-dependent.

While analysing the data I compared emerging concepts with existing research in agile software development. I had identified new roles, but also roles that were similar to findings of other studies. The names of the nine leadership roles resulted from their relation to the agile team features, e.g. the Knowledge Enabler aims at continuous learning, but also from previous research on human behavior in agile development teams, e.g. the Change Agent is named a critical role during the agile transition [PGN14].

Additionally, Scrum Masters explained that they gradually lead the team to a lesser extent and also that sometimes they would empower the Developers by doing nothing at all, which I labelled *leadership gap*. Through constant comparison [GS17] of various interviews and observations I identified nine different Scrum Master roles and developed a substantive theory [GS17] which I labelled the *role transfer process*.

I furthermore identified concepts that did not describe activities but team features which I label as the category *internal team environment*. I suggest this to be an enabler of the role transfer.

Furthermore, I uncovered challenges on organisational level that hindered the team to work in an agile way. The organisational level contained the three categories high power distance, expert culture and functionally departmentalised structure which each consisted of a bundle of concepts. For example, the category *high power distance* involved the concepts *several hierarchical layers included in decision making*, *positional legitimacy* and *management overrules*. Those categories were found to decrease the role transfer.

Through constant comparison of various interviews I identified a mismatch between organisational factors and the core features of agile teams. Based on three propositions I suggest the *Agile Matching Theory* which implies that prevalent organisational factors and the internal team environment need to fit in order for the role transfer to occur.

Theoretical saturation of data describes that data collection stops if data analysis does no longer reveal new patters [GS17]. I stopped collecting data when I felt that I received no more new insights. I started an extensive literature review [GS17] and compared my categories with existing theory. I realized that my findings were related to research on team leadership, role theory, maturity and contextual factors from research on self-managed teams. Therefore, I decided to integrate research on self-managed teams into research on agile teams to explain the changing leadership role of a Scrum Master.

5.2 Results

In this section, I illustrate the results of the interviews.

I grouped my findings on the Scrum Master activities into nine different leadership roles. While some teams described that the leadership roles were rather centred on the Scrum Master, other teams outlined that the Scrum Master role had changed over time. In the latter cases, the Developers started to take over some of the roles themselves and the Scrum Masters diminished the extent to which they played those roles.

While earlier researchers either reported on one individual who performs several leadership roles [QFTM90; Qui88] or on teams that distribute leadership roles among each other [HNM12b; ZAM09], I report on both levels of analysis. I first describe the leadership roles as they were played by the Scrum Master (RQ1) and then outline the leadership roles as they were performed by the Developers (RQ2). Thereafter I refer to the role transfer process which built the bridge between both levels of analysis (RQ3). I further describe the underlying internal team environment which serves as an enabler of the role transfer process (RQ4) and the supportive role of the Retrospective (RQ5). Afterwards I elaborate on role conflicts which limit a change in the Scrum Master role (RQ6). I conclude by reporting on contextual factors that diminished the role transfer (RQ7).

To respect participants' confidentiality, I cite them by AM (Agile Master), Dev (Developer) and PO (Product Owner).

5.2.1 9 Scrum Master Roles

I identified nine distinct leadership roles that were found to be transferred from the dedicated Scrum Master to the team while it matured. In the following I will describe each of the nine roles. Each role is described in the following order: after a short general description of the role and its aim, I describe how the Scrum Master played the role and afterwards, I describe how the Developers played the role. These descriptions answer the first and second research questions:

- Which leadership roles does the Scrum Master play? (RQ1)
- In which way do developers take on the Scrum Master role overtime? (RQ2)

5.2.1.1 Role 1: Method Champion

The aim of the Method Champion is to execute the Scrum Method to foster teams in working in an agile manner.

The role keeper organises meetings and get-together, teaches the method, such as supporting to formulate tasks and to set goals, and visualises information. Furthermore, the role involves to provide new methods and tools depending on the intended aim on team level, and to stimulate discussions on how to adapt the method to the particular team context during the Retrospective.

Scrum Master: A large majority of Scrum Masters mentioned conducting and adapting the method to be their main task when working with agile teams.

> *Well, I actually bring a broad method toolbox and expertise to the table. I've noticed it everywhere, like creative techniques. Its all about technical topics, if you can get them fast, you can create proper workshops and pick the right method. It can be a real door-opener.* (AM)

> *At the beginning it's about teaching the method. [...] Preparing the events and facilitating them. [...] It is a lot of handicrafts. [...] Well, I also do visualization such as burn up diagram, which I have to evaluate.* (AM)

Developers: In newly established agile teams, members rather waited until the Daily Stand-Up to speak about issues with each other. Over time, developers started to speak with each other right away when an issue occurred. Some teams stated that the Scrum Master had initially organised team events but after some time the Developers organised such events themselves. Also, two teams explicitly stated that the Developers visualised information on a board on their own initiative and that this was the way they learned and exchanged knowledge.

5.2.1.2 Role 2: Disciplinizer on Equal Terms

The aim of the Disciplinizer on Equal Terms is to help the team developing focus and discipline without directly monitoring developers top-down but communicating on a par. Interaction on equal terms creates non-hierarchical spaces which are important to speak openly with each other and allow team members to take on responsibility. The role keeper supports the team to stick to the rules, ensures that the team focuses on relevant topics and makes sure that team members attend the meetings.

Scrum Master: Initially, some team members were reluctant to follow the Scrum process. When the Scrum Master insisted on discipline, however, such as only talking for a certain amount of time during the daily or to follow up on measurements they had agreed on during the Retrospective, the team members started to see the benefit.

> *There are people [...] after a quarter of an hour they still talk. You do that two or three times and then you point it out carefully, it doesn't work like that. And more and more clearly and at some point I set a clock in the backlog, in the stand up and said hey, we are out of time.* (AM)

> *I consider the Scrum Master to be a person who takes action. The tools are the method and the stopwatch. If time is up, it is time to move on. That's it.* (Dev)

Developers: Due to the influence of the Scrum Master developers improved to focus and prioritise their own work. Besides reminding colleagues to stop endless discussions team members embraced the agile value focus. For example, developers reported to only do one thing at a time and not everything at the same time as they used to do in the past.

> *I also used to have this switch. That was a lot! So I tried to focus on one specific thing, on this day, for this particular amount of hours.* (Dev)

> *Usually it is a team member who says, ok we could talk about this another 40 minutes but the facts will not change.* (Dev)

5.2.1.3 Role 3: Coach

The aim of the coach is to identify which kind of team behavior is missing for executing the agile way of working and to help the team to develop this behavior. The Coach observes team members and helps the team to find ways of improvement by bringing developing conflicts to the surface and providing feedback and stimulating questions.

Scrum Master: Scrum Masters emphasized that they aimed at supporting the Developers to reflect upon their behavior and continuously improve mastering the agile way of working.

> *Scrum says that you as the team, you are the decision-makers, you have to know that. One sometimes has to do very very much, to take the road together with the people to get there. [...] That is the art of being a Scrum Master, to take the team out of their comfort zone over and over again. Without expecting too much of them but always feeding the development of the team.* (AM)

> *What you always deal with is to think and observe what went well in the team and what did not go so well. [...] As a team coach you often have to act from the background and observe strongly at first.* (AM)

Developers: Several interviewees described how they established psychological safety [Edm99] over time, e.g. during the Retrospective. Mutual trust created an atmosphere within which team members felt safe providing feedback to each other. It was no longer merely the Scrum Master who revealed personal observations.

> *The Retrospectives [...] push us to actually stand up for some opinion, to say what is wrong or to open up, and then he [the Scrum Master] unleashed the monster. I have always been very critical about lots of stuff, but now I see that everyone is critical sometimes, now I see that they [the other team members] actually care to say "look, I am not happy about this" and speaking openly had never happened before.* (Dev)

> *Well, the exchange with each other clearly is important and works very well for us. And it is also important that you talk openly and honestly with each other, what does not really work in the team and that you can talk about it without the other being offended.* (Dev)

5.2.1.4 Role 4: Change Agent

The Change Agent supports the team to change their project management habits in order to get used to the agile way of working. The role keeper serves as a role model, changes habits, and convinces newly established project teams of the agile way of working.

Scrum Master: While a large majority helped team members get used to the method step by step, others wanted to help people develop an agile way of thinking, e.g. openness towards results, rather than focusing on executing the method properly. Either way, their overall aim was to convince individuals why the agile manner was relevant and useful.

> *At the beginning, it was a bit tough to convince some team members of the agile approach. But now I think our team does not want to*

> *work in a different style anymore. There is a drive in our team that some team members even would like to go further. They have been infected with the agile virus and they want more and more.* (AM)

> *If I am convinced of something I bring it into the world. [...] simply I bring agility with me, always on the basis of a mind-set [...], so that people can understand why it makes sense to work in that way.* (AM)

> *If you do not set an example [...], you will not be able to take the team on that journey and to take that road.* (AM)

> *Then there were the new methods added, that was a big change. And they strongly questioned the meaningfullness at the beginning. 'Why do we do that? Everything used to be good and to be working.' This has died a bit over time. The people accept it now. They also see some benefits. [...] The whole thing is a long habituation process regarding the people and the processes.* (AM)

Developers: The Developers did not mention to act as Change Agents intentionally, such as prompting others to apply the method. However, several agile teams started to serve as role models for traditional project teams who apparently had raised the wish to also start working in an agile way.

> *Back then when I started with agile development, it was rather amusing. Because we felt like animals in a circus. At first, there was astonishment, then amusement, later interest and, finally, they asked whether they [our partner team] couldn't do it the same way. But this was not a process of a few days. It rather took several months.* (PO)

5.2.1.5 Role 5: Helicopter

The Helicopter aims at decreasing project complexity by understanding the broad view of a project and therefore, increasing cross-functional collaboration within teams. The role keeper owns the ability to catch the big picture of a project, to understand how topics are interrelated, to know who possesses the right skill for a certain task within or outside the team and to structure work.

Scrum Master: One Product Owner elaborated that he has chosen the architect to be the Scrum Master because he believed that this person was most suitable to identify cross-boundary links between components and team members. The team members of that team reported that the Scrum Master coached them to learn which task they may choose.

> *In our team, I don't feel like I am the boss or anything of that kind. I am just the one in my team who is best at keeping track of things and to give them a general direction.* (AM)

> *Either it is good if the team itself has the complete overview and everyone knows a little bit of everything or you need someone who kind of intellectually brings the threads together. Well, someone who spots what people get out of single issues. So to say if there was a complex heterogeneous team and you let it work independently, they may not necessarily look to the right and left what other people can do.* (AM)

Developers: Due to daily communication and visualisation, team members formed a mutual understanding [Hod11; MDD10] while they matured, so that they knew who had which knowledge or skills. Developing a Helicopter perspective helped team members to connect tasks, to know who they could approach for certain topics and to quickly transfer work to relevant experts.

> *But one can sense now that we try to exchange information with each other more and more, because we also see the complexity. It is really no longer the way it used to be that someone says: 'hey, I pick this single topic and do it all by myself.' [...] Because there are always some dependencies between topics and we try to sit together from the beginning. And this is how everyone gets to know the other topics a little bit.* (Dev)

> *In the beginning, I think you don't know who has more experience in a certain area or expertise in another area. But slowly I get to know everyone and can judge who can support me in which difficulty in the quickest possible way. [...] But in the end I know, okay, I have a problem here and who can support me.* (Dev)

5.2.1.6 Role 6: Moderator

The Moderator translates among different perspectives to help the cross-functional team develop a shared understanding. The role keeper moderates all kind of meetings and builds a bridge between perspectives and domains.

Scrum Master: The Scrum Masters mediated between individuals from different domains and helped the team to tolerate each others point of view.

> *We like to discuss a lot, thus the Scrum Master needs to moderate the conversation, to mediate between different points of view.* (PO)

> *So it is always like this, or in my understanding like this, that you train the other members a little bit through the dailies, through the communication that is supposed to take place. So that they understand what the neighbour can do to support and help each other. So that on the one hand the interfaces work, but also so that ideas are passed on when you come up with something and say 'I've seen something here, you can use it' or vice versa.* (AM)

Developers: We have not come across any team within which the Developers succeeded in playing the Moderator. One team had attended a well-prepared meeting which was moderated by a developer. While it was supposed to be a Retrospective it had ended up in a planning instead. Two teams believed that the Scrum Master needed to keep the Moderator role since it was considered to be difficult to remain neutral during a discussion while being part of the Development Team.

5.2.1.7 Role 7: Networker

The Networker ensures that the team has access to required expertise that goes beyond the current knowledge within the team. The role keeper connects the team with relevant stakeholders, e.g. managers and experts, from within and outside the organisation.

Scrum Master: The Scrum Masters reported to approach relevant experts depending on the team's demand. For example, they approached disciplinary supervisors to support the agile way of working, invited experts or trainers from inside and outside the company to explain topics, or called internal administrative support when needed.

> *For me it is very important to see and help people when they need external help. This is very important, too, when they need to go outside of the team to contact somebody else. In the organisation or even outside the project.* (AM)

> *Another one is to build up connections so you know whom to ask the question and from whom to learn. [. . .] . Or who from outside could be the one to see what other steps we have to do if a guy is missing.* (Dev)

> *In that case I knew someone from the past [...] who focused on problem-solving methods. And than I booked him for an analysis [...] and then we sat together and [...] we realized, that his way to formulate a question, and his systematic approach, that this was really good for us. After just 10 minutes he had proven to the team that he got what it takes and led it into the right direction.* (AM)

Developers: Over time, team members increased their personal network and learned whom they could approach for what. Moreover, during dailies the Developers increasingly offered personal network contacts to their team members. This increased access to knowledge and improved speed of solving tasks.

For example, that one has an information for someone, that he normally would not have access to as a planning guy. [. . .] Actually, I bring in my network from production and the developer his network and the TEF person yet another. During the open discussion at the Daily Stand-Up, I can say that I have a problem. Someone knows someone who can help me with it. (Dev)

If the expert is not within the team, we have to reach out to other people who are not part of our team. [...] For example, we had an integration task to do [...] and the team did not have this expertise or has done that activity so far and the one team member took up this task and he was supported by the team members in whatever extend they can. But since we did not have this expertise we got support from another external team and then he delivered this task. (Dev)

5.2.1.8 Role 8: Knowledge Enabler

The aim of the Knowledge Enabler is to teach the team a new approach to learning thereby team members work themselves fast into new topics and solve complex issues which had not been solved before. The role keeper recognizes which kind of knowledge the team needs, e.g. expert information or methodological skills, and supports team members to acquire that knowledge, e.g. sends them to training or conferences, and schedules knowledge exchange meetings. Furthermore, this role promotes iterative learning, e.g. learning from mistakes, fosters learning-by-doing and helps the team to become used to visualizing information, e.g. on whiteboards.

Scrum Master: Some Scrum Masters urged developers to take time for learning and to share knowledge more openly with each other. A few of them convinced managers that agile teams must sit close to each other to approach each other easily, learn from each other and build a shared understanding.

There is one person who is a specialist. It is important to me that not everything goes through this person, but that this person explains

> *to the other person how it works. So that this person can do the job him- or herself in the future. The person should get the ability to solve the task by him or herself in the future.* (AM)

> *They just do not know the whole approach and how to access it. They know classic learning like you go to a training or you study a book, but in this field, you have so many user groups, meet-ups [. . .]. And we also try to just propose a nice event. They can meet other people there and discuss with them. For example, we all went to a conference together.* (AM)

Developers: While a few team members expected the Scrum Master to own technical expertise to provide feedback, other interviewees had learned to receive feedback from their peers. They shared improvements and lessons learned and served as a sparring partner. They also described to simply approach colleagues to ask for information, to learn from each other and to work together on tasks.

> *Every now and then I also sat down with colleague A in the meeting room and simply discussed a topic where it was not so clear to me. There one has already helped each other.* (Dev)

> *But for a start, it's easier to really see everything. And then they put a component together and I did it with someone and that's how I learned it.* (Dev)

> *Today it is all very easy going. I just go over to my colleague's desk, sit down for, like, 45 minutes and work with him on a topic. Nobody says anything against that. It is very informal, but it also happens that I personally have to answer some questions.* (Dev)

5.2.1.9 Role 9: Protector

The Protector aims to maintain trouble-free working conditions as agile teams are meant to focus on their respective sprint goals only. The role keeper shelters teams from inappropriate requests from the Product Owner, managers, disciplinary leaders and other departments.

Scrum Master: Scrum Masters reported to shelter the team from re-prioritisation or excessive work demands by the Product Owner. Moreover, they protected the team from management interfering in daily business or reversing team decisions.

> *You have to be up front if there are attacks, and there are attacks. From somewhere else, from other departments or from outside. You have to be up front and protect them.* (AM)

> *Managing stakeholders. Talk to all those who are involved in the reviews, those who have had arguments from time to time, and communicate with them. That way they can keep work off the table. Talk to them, because they always asked, 'do this and do that'. Then simply say that this makes us no longer capable of acting.* (AM)

> *But then, I also pushed some things through in certain teams, [...] in which managers had taken decisions again. I had to go to the management and tell them "that is not OK, you make a mistake". Then they had to compromise and later they were really glad that they had reacted that way. Because the team gave the right hints after all. That is a situation in which one has to fight a battle on behalf of the team.* (AM)

Developers: Multiple team members reported on occasions when they encountered disturbances by stakeholders, yet, no Development Team had created a strategy how to protect themselves successfully from such disturbance that lasted for long.

I came across one approach by developers aiming at protecting themselves but did not succeed long-term:

> *Not everyone could participate, because actually we also have to do the fire fighting thing. We have a Batman for this. One of us would take over the role while the others were working. For example, someone for Monday, for Tuesday and so on. We rotate with this. [...] now we do not answer any call when we are not batman. So the team was shielded and we could actually learn in little steps.* (Dev)

In particular, teams without a regular support by a Scrum Master struggled to work in an agile manner. One team reported that it had occurred twice that management removed members temporarily while the Product Owner was absent. Likewise, a Product Owner of another team revealed that he struggled with not disturbing operational work and tended to give orders. Both Product Owners wished that there was a Scrum Master on a regular basis to defend the team.

Investigating which role the Scrum Master plays (RQ1) and in which way it changes over time (RQ2), I found that **the Scrum Master played nine different roles which are transferred to the team while it matures.** In addition, I found that some roles were more suitable for a transfer to the team members than others.

Based on the findings I suggest the following propositions:

(1) The Scrum Master involves the nine different leadership roles Method Champion, Disciplinizer on Equal Terms, Coach, Change Agent, Helicopter, Moderator, Networker, Knowledge Enabler and Protector. (2) The Developers take on the Method Champion, Disciplinizer on Equal Terms, Coach, Change Agent, Helicopter, Networker and Knowledge Enabler over time.

5.2.2 The Role Transfer Process

The third research question was: How are the leadership roles transferred from the Scrum Master to the Developers? (RQ3) I found that roles were transferred via **three steps** I labelled the **role transfer process** (shown in figure 4.1).

The first step illustrates that the Scrum Master serves as a role model by demonstrating the activities of the nine leadership roles while the Developers observe the behavior to learn it. The Scrum Master empowers the team in understanding the value and utility of the roles, e.g. during the Retrospective. They build a **shared understanding** of the roles and its meaning for the agile manner and agree to aim at distributing the leadership roles among team members.

I try to help colleagues to find their way into the roles. It is always *tricky to keep the balance between what the team should do by themselves and what should be done by the PO or SM. That is one thing that one has to reflect upon and to level out.* (AM)

The second step describes that the Scrum Master provides a leadership gap when the team is considered to be more mature. The Scrum Master plays particular roles to a lesser extend while keeping management and Product Owners from undertaking the respective role. The emerging leadership gap provides the space for the Developers to perform the roles themselves. While a team member claims to play the role the colleagues grant that person to take on the leadership role and respect the new role keeper.

As a Scrum Master I can provide strong support at the beginning to get started. But then I have to retreat gradually so that the team gets into the mode of self-organisation. Because if you do not create some free space or a vacuum, nobody will jump in. (AM)

The most exciting thing is to bear the silence until someone says something and to wait until someone else gets active. [...] Also we have to give them some free space to experiment and try out themselves. (AM)

Scrum Masters claimed that they either prepared a leadership gap intentionally by withdrawing from particular roles while waiting that Developers would take on the role, or they were not playing the role due to being absent which gave the team the chance to perform the role. Moreover, some teams experienced that management truly empowered them to take on responsibility, whereas others encountered that management, Scrum Masters or Product Owners clung to power and kept them from taking on leadership roles.

I did not have sufficient capacity to do everything myself. Therefore, some team members took over tasks, e.g. one guy arranged a timer, another one took care of the whiteboard. They were quite proactive as a team. [...] They did not tell me: "You are in the Scrum Master role, you have to make things better for us." (AM)

The third step contains that the Scrum Master only plays certain roles if the team needs support while team members perform most of the roles themselves. Yet, I found that not every role can be passed on to the Developers equally because for some roles the keeper must not be part of the Development Team, e.g. Moderator and Protector. This indicates that the dedicated Scrum Master does not disappear in a mature team but is played to a lesser extent over time.

> *It takes a lot of energy but is quite nice to experience when the team gradually walks by itself. At the same time, the time effort by the Scrum Master can be reduced.* (AM)

Based on the findings I suggest the following propositions:
(3) The Scrum Master changes in three consecutive steps over time, such that the role is rather centred on one dedicated Scrum Master in an immature team, while it is rather shared in a mature team.

5.2.3 Factors Influencing the Role Transfer Process

5.2.3.1 Internal Team Environment

The fourth research question aimed at exploring the team enablers required for the role transfer to occur: What is the underlying internal team environment required for the role transfer to occur? (RQ4) I found **eight enablers** shaping an **internal team environment** that stimulated team members to take on leadership roles. Figure 4.2 illustrates the factors describing the internal team environment.

Firstly, teams that communicated with each other **on equal terms** and seemed to refrain from hierarchical thinking appeared to share leadership roles more openly among the Developers and the Scrum Master. The Scrum Master was not considered to be a formal leader they had to please. Hierarchical free space allowed the leadership gap to occur and therefore, to claim and grant a leadership role. The overall atmosphere within the team should reflect that it is generally accepted to take on a leadership role even without being a formal leader of the group.

> *I believe that somehow this it what makes it special at this place. Even if people have different pay grades, you do not feel like this one is the one being above the other one because this one belongs to a higher salary group than the other one. Maybe also because of the occupation or because of the level of knowledge. No one makes someone else realize that there might be a difference between each other but rather it works well with each other.* (Dev)

Secondly, teams who communicated on equal terms had established **psychological safety** [Edm99; MDD10] which made them feel safe taking over the risk of playing a leadership role without previous experience in it. The Scrum Master was found to provide safety by the Scrum process. The Retrospective helped teams to build trust among each other and encouraged team members to talk openly about personal matters.

> *I think, first, one benefit is that we learned [. . .] to lose the fear of talking. So the methods forced us to bring out opinions, to give the opinions on something that was bothering.* (Dev)

Additionally, as referred to in Section 5.2.2 team members had developed a **shared mental model** [LWW01] regarding the meaning and content of the roles often via **transparency**, e.g. during the Retrospective. Furthermore, they had agreed that anyone in the team could claim and should grant performing the leadership roles.

> *It is also not that easy for the others of the team. [...] They had never been working in an agile way. And I believe to make agile work you need to embrace a specific mind-set. They need to be conscious about that they work in an agile way now. And that you have the right to say 'no' and that you have the right to say to the PO: No, I cannot handle [this workload]. And this is not that easy.* (Dev)

Moreover, **team orientation** [MDD10] was identified to be important for team members to understand how each role contributed to the shared team goal. It empowered teams to feel responsible for a particular topic and thus feel the urgency to play the respective leadership role. For example, one team struggled in iterative learning and they complained that they had no vision. They felt like the knowledge they were required to learn was useless, and they did not understand why they should learn continuously. As a consequence, the team members seldom took over the Knowledge Enabler role.

Furthermore, I found that developers who performed leadership roles increased their **team potency** which implies to believe in the team's capability to be successful [GYCS93]. Therefore, they felt motivated to continue playing leadership roles in the future. Yet, it seemed to be challenging to have the courage to get started playing the respective leadership role for the first time. One team member described it to be painful to take on a leadership role when facing the leadership gap initially.

> *What gave me at least some confidence to make decisions is that I learned what is my degree of influence. If my decisions now reach this scope and I am not crashing the project because they are working on the tool I am supposed to know. I think most of the confidence is about the environment.* (Dev)

Additionally, team members felt that **self-monitoring** [MDD10] would empower them to take on leadership roles. Teams with low level of self-monitoring stated that they externalized the sense of responsibility to a formal leader.

> *I think, maybe if the Product Owner would not tell me everyday what I have to do, maybe I would be more intrinsically motivated to do my tasks, maybe I would chose my tasks voluntarily. But like this. . . I just deliver a status report to my Product Owner every day! (Dev)*

Finally, **team learning** [Edm99] allows the team to continuously reflect upon the leadership roles, e.g. during the Retrospective, and therefore, develop themselves further regarding playing the divers roles.

Based on the results I suggest the following proposition:

(4) An internal team environment of communication on equal terms, psychological safety, transparency, shared mental models, team orientation, team potency, monitoring themselves and team learning enables that roles are transferred from the Scrum Master to the Developers.

5.2.3.2 Retrospective

Yet, some teams reported to struggle in developing the internal team environment needed for transferring the roles. The fifth research question was therefore: How can the Scrum Master foster the internal team environment? (RQ5)

The interviews revealed that the Retrospective supported agile teams to develop a supportive internal team environment. One interviewee explicitly said that the Retrospective helped the team to continuously develop itself further, while others circumscribed its positive effect on the team atmosphere. In the following I will provide different examples of how the Retrospective influenced the internal team environment.

The Retrospective helped the team to develop *shared mental models* regarding working together as a team and on individual preferences, e.g. with regard to personality or working style. A shared understanding of differences between team members also stimulated *team orientation*.

> *That one learns what makes the different personalities tick, how do I perceive them in the different rounds. Starting with the standup, through the review to the Retrospective, how do they act, do I perceive resistance, do I perceive that they have difficulties, how team-oriented are they, or how strongly do they simply push ahead without taking the team with them. I think this is the most important thing a Scrum Master should do to stimulate teamwork.* (AM)

Additionally, the Retrospective led to *transparent communication* which is one prerequisite to talk openly about the roles.

> *If communication did not work well, we normally addressed this topic during the Retrospective, or rather if it was not addressed*

> *there, we would have had a bigger problem inside the team. The climate was quite good since we always talked about everything. Also when there were coordination problems, the people treated each other well.* (AM)

Furthermore, the Retrospective established *psychological safety* which is necessary to have the courage to take on the divers roles.

> *Each time before the Retrospective I tried to explain that the purpose was not about blaming someone but to discuss constructively with each other and to develop further. I want to create an atmosphere within which failures are allowed to happen.* (AM)

The Retrospective also helped team members in developing *team potency*. Agile teams learned that they themselves were responsible to change situations for the better. Consequently, team members became aware that they were responsible to take on the divers leadership roles.

> *I believe they have developed into a real team because we always reflect upon our-selves. For example, that it is not the task of a Product Owner to set the direction but that we expect and that we also wish that they also think for themselves. They should not only be sheep in the flock, like it would be in a traditional project. Also we give them the freedom to try new things [...] to encourage them to just give it a try. If someone imagines, yes, I can do it, well than do it.* (AM)

> *You realize that the team really demands the Retrospective. Actually, every week there is something that one can improve, or at least discuss with the team. I would say doing the Retrospective is also something that we have achieved by the agile way of working [...] It also changes the way you look at things. Our first Retrospective was quite bad. There was always an external view. Such as the manager does not do anything about it, or the team does not do this or that. Always focused on individuals. At some point we managed [...] to talk about things that we could influence and could change our-selves. Even though those are not big changes, it motivates me when I see, that I can change things and that I benefit from it.* (AM)

Investigating how the Scrum Master fostered the internal team environment I found that the Retrospective promoted different facets of the internal team environment.

Based on the findings I suggest the following proposition:

(5) The Scrum Master empowers the team to develop a supportive team climate and consequently take on leadership roles by facilitating the Retrospective.

5.2.3.3 Role Conflicts

So far, I have described that the Scrum Master tends to play nine different leadership roles which are handed over to the Developers via a leadership gap, a supportive internal team environment and the Retrospective. Yet, despite a supportive internal team environment, still parts of the Scrum Master role may not be transferred to the Developers. Interviewees often described rather bureaucratic demands on the Scrum Master that led to role conflicts and to a struggle in providing the leadership gap necessary for the role transfer. The sixth research question was therefore: Which expectations on the Scrum Master limit the changing leadership role? (RQ6)

In the following I will refer to divers expectations by the Developers, managers and the Product Owner, as well as requirements due to the position of the Scrum Master in the organisational chart.

Developers

Often team members expected the Scrum Master to behave in an agile way, e.g. trusting the Developers to monitor themselves while opening the leadership gap.

> *He is a person who gives freedom to the team member to select tasks that he or she would like to do and in case of problems the Scrum Master is the one who can support and enable the team member to really work on the topic. In another way the Scrum Master is a team member but if he directs the associate how to do things this would not be a right Scrum Master.* (Dev)

Yet, some team members also expected the Scrum Master to take on typical project manager tasks, e.g. reporting, providing clear direction or taking over responsibility for team decisions. These behaviors are in contrast to maintaining a leadership gap within which the Developers take on leadership roles.

> *For us it is very difficult to find the right role as a Scrum Master where he only follows the methodology, because from an organisational point of view we actually need a project manager as a developer who tells us what to do and our Scrum Master already takes over this role. If he didn't know what was going on then it would be difficult because then we would have to manage ourselves completely and our organisation would not allow that.* (Dev)

Therefore, even though the Scrum Master would provide the leadership gap, team members would not take the opportunity to take on leadership roles.

Managers

Managers involved the disciplinary supervisor of the Scrum Master or of the Developers, internal customer and line managers. A few managers expected the Scrum Master to improve teamwork. Yet, many managers rather demanded achieving defined hard facts, e.g. improved efficiency, and keeping to formal standards. Some managers had expectations regarding the Scrum process, formal standards and leadership behavior that were in contrast to opening the leadership gap.

> *Maybe the expectation of the line managers is that the agile methods are faster for the time being. It can be faster, but I don't think this is the goal of agile project management.* (AM)

> *Of course, everyone talks about the agile way of working. But when it really comes to implementing it and you have to let go of things, as I said earlier, then it is simply a long way. [...] So one must also shape the culture from above. Of course, it is also important that*

> *the employees are also involved in some way. Someone once said in a management seminar, [...] 'my employees can do anything until they make a mistake.' And I thought, you didn't really understand much of Agile. These are the things where you still have to work hard on it.* (AM)

> *It was easy for the project leaders, the group leaders and the department leaders to say 'from now on you do Scrum'. What it was really all about often did not matter to them at first. Since then, [...] they have gone through a long valley of tears while doing a mix of Scrum and of we don't do it after all. The department heads still wanted to see their documents, still wanted to orientate themselves on the waterfall model. But overwrote it with 'Scrum'. Of course, this was a stupid hybrid for the project teams.* (AM)

While the Developers should take on leadership roles themselves and are therefore responsible for work outcomes, manifold managers expected the Scrum Master or Product Owner to be mainly responsible for outcomes.

> *This topic reporting structure, this must change colossally. This internal reporting to the next level of the hierarchy, that must no longer be. We are using a relatively large number of resources in the Group for this. That's why I believe that today's management team must radically change.* (PO)

Therefore, the Scrum Master was rather expected to follow the rules of the pre-existing bureaucratic organisational system instead of serving as a Change Agent and Coach for the agile approach. The Scrum Master was torn in between meeting traditional management requirements and embracing the agile leadership behavior, such as providing a leadership gap.

Product Owner

Some Product Owners expected the Scrum Master to serve as a counterpart with whom they could have constructive conflicts. Yet, some Product Owners were found to put high pressure on the Scrum Master or directly on the Developers. This led to the Scrum Master also putting more pressure on the

Developers instead of opening the leadership gap. Furthermore, it diminished the willingness of the Developers to take on leadership roles.

> *The PO has a lot of pressure. He controls more than he actually performs his role.* (AM)

> *That [the Product Owner position] is not really a nice position to be in in such an organisation. Because you have a team that has the pull and you are under a lot of pressure, you have the push and you have to bear it, this balancing act where you say, ok, I'm letting go of the reins for two weeks and I'm under incredible pressure. And I'm put under pressure every day. It's incredibly difficult.* (PO)

Scrum Master Position in Organisational Chart

Furthermore, some Scrum Masters said that the way their role was formally embedded in the organisational chart led to role conflicts and diminished the options to open the leadership gap. For example, some interviewees described that they hold distinct formal roles within one team that embodied contradicting interests.

> *That would also be a point of criticism that I can say, if you are also PO, you have a conflict of interest. [...] I am a technical project manager and I am already involved in the project goal setting. So my project goals are to lead the team to success by delivering our projects in time to market, cost and quality and performance. [...] As a Scrum Master I have to say, no, that is too much. It overtaxes the team, or do we need a shift as project manager or PO, I cannot say that.* (AM)

> *The disciplinary superior sits in the line. The [annual employee development dialogue], i.e. the success dialogues, I do them anyway. Depending on what kind of agreement you have there, but usually the line manager is also involved. But he [the disciplinary supervisor] hardly sees the people anymore. So it's not perfectly separated for me. [...] In the classic blueprint, someone who makes success dialogues would not be the Scrum Master, but the PO almost, in the classic theoretical picture.* (AM)

> *I am part of the team myself. I still perform my tasks in the team, in the part of the project where my responsibilities still lie. This is sometimes a difficult point, because you can't discuss the work packages from the outside in a really neutral way as a Scrum Master, but you also have a certain amount of action and interest in who does what and when.* (AM)

One interviewee was simultaneously Scrum Master and disciplinary supervisor. While the role of a Scrum Master was rather expected to provide a leadership gap in which the team took on leadership roles, the disciplinary supervisor was rather considered to make decisions him- or herself.

> *For me, I would say the tasks differ a little bit. On the one hand, as I said, I would rather see myself as a team coach and as part of the team, because that promotes the cause. On the other hand, [...] I also have other tasks. I have to do that from time to time, although we try to make a decision and see how we get it done.* (AM)

Furthermore, some Scrum Masters lack in positional power to protect the leadership gap.

> *If the environment, especially the managerial authority or something, or the delegation to these agile teams, [...] so to speak what I can't influence from the outside. If the department managers who release the people or the department managers who have the management task [...] who actually have the line responsibility, if that doesn't fit.* (AM)

> *I mean when I say there is nothing it is a volunteering thing if you say 'No, I don't want these people working there'. I just do not know the consequences. I do not know if they have the authority to say no. 'No, I am not giving anyone of my people to you', I don't know the consequences. You do it in good faith and release people to the task force.* (Dev)

We uncovered how demands by the Developers, managers and the Product Owner limited the role transfer. The Scrum Master was often expected to meet requirements deriving from a rather bureaucratic view on 'leading a project team', e.g. reporting, monitoring and decision-making. This contradicts the nature of the agile way of working which implies that the Developers take on leadership roles. The contradiction led to role conflicts between meeting the needs of the existing bureaucratic system and the new agile approach. Fulfilling the rather traditional requirements restricted opening the leadership gap, thus, limited the changing Scrum Master role.

Based on the findings I suggest the following proposition:

(6) Expectations that aim at bureaucratic behavior limit the changing Scrum Master role.

5.2.3.4 Contextual Factors

The last subsection described role conflicts deriving from expectations based on a bureaucratic company context. I will now further describe the company context that helps us understand where the expectations come from.

The last research question in my qualitative study was: How do organisational culture and structure influence the role transfer? (RQ7) Past research revealed that the context plays an important role for transforming into an agile team [HN17]. I explore how the bureaucratic context influences the role transfer. The findings describe challenges regarding organisational culture and structure and in which way each factor influenced the role transfer.

Interviewees often mentioned that **organisational culture** linked to high power distance and specialised knowledge workers reduced working in an agile way.

High power distance was exemplified by several hierarchical layers which all needed to be included in decision-making processes even though some were said to not even add value, by management or Product Owner that felt to have the right to overrule team decisions or told the Developers what they had to do and by status legitimacy. As a result, some developers reported to be frustrated, demotivated or to fear that management or Product Owner would overrule their decisions anyways. Thus, high power distance resulted in a low level of self-monitoring, of psychological safety and of team potency. Which led to a lack of willingness and capability to make team decisions.

> *But you don‘t necessarily receive decisions, you don‘t get information if you go there and say: hey, I am a team member of the agile team. If you go there as a common team member, you only get the information if you are in the proper hierarchical layer, only if you talk to a person at the same hierarchical level. That are the power plays.* (Dev)

I therefore suggest the following proposition:

(7) High power distance diminishes the role transfer (7a). This relationship is moderated by monitoring (7b), psychological safety (7c) and team potency (7d).

A **specialist culture** was described by territories and by a lack in readiness to experiment. Some managers were said to prefer teams to follow a strict plan that was thoroughly thought through in advance and were reluctant to apply an inspect and adapt approach.

Territory refers to an expert who enjoys a sovereign right to keep specific knowledge and even the respective task all to him- or herself without sharing it. Therefore, other team members did not feel responsible or allowed to learn the specific knowledge. Interviewees also referred to a lack of willingness to learn things unrelated to the personal field of expertise or to a lack of discipline to not dig too deep into an expert topic.

> *Well, there are clearly defined areas in this team. Territories which are well hidden. You need some time to recognize them. [...] certain people are in certain territories which is simply inflexible. You realize this when there is one topic dropped, it is not considered. And than there are the people waiting and have nothing to do, even though there actually needs quite a lot to be done. Because there are the territories that you are not allowed to enter and than one does not work together.* (SM)

While some team members appeared to own high specialization regarding tasks, they had a low willingness to think cross-functionally. Thus, a specialist culture resulted in weak team learning, shared mental models and team orientation, thus, developers did not perform roles like the Knowledge Enabler. I suggest the following proposition:

(8) A specialist culture diminishes the role transfer (8a). This relationship is moderated by shared mental models (8b), team learning (8c) and team orientation (8d).

Some interviewees described the **organisational structure** as functionally departmentalised. It involved the categories departmental silos, department-oriented goals, geographical distribution and rigid processes.

Developers reported that they had to rely on external know-how due to processes which slowed them down. Some teams that depended on external support had difficulties in receiving a timely response or even at all.

> *If a team is not both at the same time - a self-organised team and the company itself - you always need interfaces, stuff from other people. And especially big companies are often divided into silos, which makes it very very difficult. You always need to wait for stuff. You ask for something and then you don‘t get it. And you do something, you show it to someone and then they don‘t respond. [...] or because they simply don’t have time to respond.* (PO)

Sometimes, different sprint goals or contrasting departmental goals resulted in slow decision-making. For example, purchasing would try to reduce costs while developers searched for the technically best solution which would be more expensive. Some agile teams reported to clash with traditional teams that followed a classic project plan in contrast to iterative learning. Those neighbouring project teams were considered to be slow and inflexible in relation to agile teams, which made it difficult to synchronise project goals and milestones. Interviewees also said that it was a challenge to include external experts for the up-coming sprint in advance. Furthermore, geographical distribution limited knowledge exchange, synchronisation of progress, visualisation, discussions about critical topics and decision making.

Therefore, a functionally departmentalised structure resulted in weak shared understanding, team learning and team orientation. This reduced the presence of leadership roles in the team. I suggest the following proposition:

(9) A functional departmentalised structure diminishes the role transfer (9a). This relationship is moderated by shared mental models (9b), team learning (9c) and team orientation (9d).

5.3 Discussion

Despite a steadily increasing number of attempts to implement agile teams in industrial setting and a growing body of papers pointing at challenges such teams face, scarce empirical research explores how teams can actually learn to work in an agile manner. My research objective was to explore how the leadership role of a Scrum Master empowers a team to work in an agile way in real organisational settings with a particular focus on the changing Scrum Master. I identified nine leadership roles of a Scrum Master which are transferred to the Development Team while it matures. The core element of the role transfer process was found to be the leadership gap: a lack of leadership which offers the opportunity for the Developers to step up and take on leadership roles which were previously performed by the Scrum Master. Yet, roles were not always transferred to the Developers. I found that role conflicts limited the role transfer. Role conflicts emerged from a mismatch between features of the agile way of working and formal requirements, as well as expectations deriving from informal organisational characteristics, e.g. a bureaucratic culture or authoritarian management style.

While some researcher consider a self-organised team to perform at its best when leadership roles are shared within the team [HNM12b; ZAM09] other researchers examine competing leadership roles within one individual [QFTM90; Qui88]. I put a new perspective to role theory and suggest that a

Scrum Master first involves multiple leadership roles which are transferred to the Developers while they mature. Consequently, roles are shared between the Scrum Master and the Developers.

My suggestion that developers can learn to take on parts of the Scrum Master role is in line with the findings of other researchers [Bäc19; MDD10]. Furthermore, just as Yang [Yan96] who implies that the roles Monitor and Coordinator may turn into team values, I suggest that certain roles may turn into team values in a more mature team. The Disciplinizer on Equal Terms may turn into the team values of focus and working on a par.

Yet, I do not believe that all roles can be shared equally among the Developers and the Scrum Master but instead claim that the Scrum Master keeps the Protector and the Moderator role in a mature team. Therefore, suggest that some roles should always be played by the Scrum Master, which is a similar result as the findings by Hoda et al. [HNM12b] who discovered that in the absence of specific formal role keepers some aspects of agile working lost the team's attention, such as the Retrospective. In line with Gren et al. [GGJ19] I found the Retrospective to be an important enabler for continuously fostering an internal team climate that stimulates the role transfer. I am thus convinced that a Scrum Master will not become obsolete in a mature team but is continuously needed to empower developers to work in an agile way.

The role transfer process reflects several elements of the forming-storming-norming-performing model by Tuckman [Tuc65]. The forming phase by Tuckman [Tuc65] suggests that team members focus on a leader who sets ground rules for further cooperation. Team members are insecure how to behave and search for opportunities to observe expected behavior. It is therefore decisive for a Scrum Master to demonstrate the agile leadership roles from the beginning onward and to agree with the team that they aim towards sharing those roles.

While some interviewees described that the Scrum Master provided a leadership gap others had experienced that the Scrum Master, the Product Owner or the managers struggled with transferring leadership roles to the team. This is also described by role conflicts during the storming phase by

Tuckman.

Over time agile teams establish a shared understanding of roles and responsibilities which is expressed by the norming phase by Tuckman. Team members who are used to a rather traditional way of working become to understand how to work in an agile manner. They start to claim and grant leadership roles: team members learn to play parts of the Scrum Master role themselves while being allowed to do so by team members, the Scrum Master, the Product Owner and the managers.

The performing stage allows individuals to take on roles whenever it makes sense. Yet, it is important to emphasize that I have not come across many teams during my research in which leadership roles were truly shared among the Scrum Master and the team members. This matches with earlier empirical observations that the performing phase is seldom reached in organisational context [MMZ01].

I suggest that a contradiction between expectations deriving from a rather bureaucratic organisation and from the agile way of working is one of the reasons why truly agile teams are rare in company settings.

Similar to previous studies [BP19; NRBB17; SSD16] I identified role conflicts. The Scrum Master was torn between meeting demands of rather bureaucratic organisations and of the agile way of working. Those role conflicts limited the role transfer from the Scrum Master to the Developers. Therefore, to enable Developers to take on leadership roles, the organisation should adapt its processes and requirements to the agile approach. An alternative option would be to add project management activities officially to the Scrum Master role. Yet, than organisations should not call their teams self-organised but semi-autonomous teams [SHW19b] since otherwise teams may have wrong expectations of the agile way of working which results in frustration. Therefore it is important for organisations to decide to which degree they want to provide leadership roles to the agile team and communicate clearly what is expected in the specific company setting.

Furthermore, while researchers either consider the Scrum Master as part of a method or as a leadership role, I combine both point of views and suggest the Method Champion to be one leadership role of the Scrum Master.

The Method Champion facilitates the Scrum process, e.g. Retrospectives, which sparks an internal team environment of psychological safety and shared understanding within which team members feel empowered to take on leadership roles.

Command-and-control behavior by Scrum Masters, Product Owners or management was found to weaken self-organisation of teams [Hod13; MDD10]. Bäcklander [Bäc19] advises Scrum Masters to be absent from time to time to improve teamwork, which was already found to increase information sharing among team members in earlier studies [MDD10]. I believe that my finding of displaying a leadership gap that empowers developers to play leadership roles fits well with those previous empirical findings.

5.4 Limitations and Future Work

In the following I will critically reflect upon the research limitations and suggest topics for follow-up studies.

Grounded Theory does neither claim to test hypotheses nor to be universally applicable but to build new theory grounded in the specific context under research which can be tested quantitatively in proceeding research [GS17]. Therefore, my results cannot be generalized.

Yet, to increase **construct validity**, I draw data from different organisations from one conglomerate and used multiple sources of evidence by capturing the Scrum Master from three different angles involving Scrum Masters, Product Owners and Developers. The researchers discussed the extracted results and built concepts and theories. Additionally, emerging results were frequently reflected critically with various agile practitioners from the company and the main author observed multiple agile teams at the company site over a period of 1.5 years. The final results were supported by the observations and discussions with practitioners.

All participants work at the same corporation, mostly in the automotive industry. To increase **external validity**, I aimed at approaching an equal number of project teams at each division. Despite their slightly similar overall working culture, the 11 business divisions embrace different subcultures. Yet, I do not claim my results to be universally applicable and acknowledge that they might be limited to the specific context. Nevertheless, I did conduct a research cooperation with another conglomerates active in the automotive industry which will be briefly described in Section 7.4. Further studies should compare my findings on the changing Scrum Master role with data drawn from other companies from varying industries.

Reliability: An open-ended semi-structured questionnaire guided the interviews and therefore, each interview followed a similar structure. Yet, I asked participants about past activities based on memory. Since memories of individuals tend to change retrospectively my interviews are difficult to replicate. Furthermore, qualitative studies cannot proof hypotheses. In order to examine whether a larger majority of Scrum Masters and developers play the nine leadership roles I have identified, a quantitative follow-up study is needed to increase reliability of my study.

Social Response Bias: I only asked participant who were willing to take part in the interview. At the beginning of each interview, participants were informed about the purpose of this study and assured of confidentiality, so as to receive open and honest responses.

The majority of participants spoke openly, also about their personal concerns what was not working well in their organisation or in their agile team. There were three people from three different teams whose overly positive statements did not match with the comments on the same topic from other interviewees of the very same team. Moreover, after the official interviews had terminated, the three interviewees made statements that contradicted the content of the previous interviews during small-talk with the interviewer. Since the authors could not be sure whether social response bias applied, those three participants were excluded from the sample.

Chapter 6

Part II: A Quantitative Exploration

So far I have investigated how leadership in agile teams changed over time based on observation and retrospective narratives of interviewees. I have built a substantive theory on the changing leadership role. I have not yet quantitatively explored the findings on the presence and change of the nine leadership roles. In a follow-up study, I aimed to quantitatively explore the presence and distribution of the nine leadership roles in relation to maturity.

Part II reports on the quantitative exploration of the 9-Factor Theory. This chapter is predominantly based on the paper *A Quantitative Exploration of the 9-Factor Theory: Distribution of Leadership Roles between Scrum Master and Agile Team* [SGHW20a] which was published in the proceedings of XP'20. I slightly modified the original article to make if fit with the rest of this thesis.

Section 6.1 recaps the key findings related to the literature review and Grounded Theory study, while Section 6.2 describes the study design. Section 6.3 reports on the findings of the quantitative exploration by referring to the nine leadership roles of a Scrum Master and the sharing of the distinct roles between Scrum Master and the Developers while taking maturity into account.

Section 6.4 discusses the findings critically. Practical implications for the Scrum Master description are suggested in Section 6.5. Finally, limitations and suggestions for future work close part II.

6.1 Concept

6.1.1 Motivation

Studies have found that leadership influences the ability of a team to work in an agile manner [Bäc19; GTF17; MDD10; SJ17]. Different qualitative studies suggest that the leadership role of a Scrum Master changes while the team matures and that some aspects of it are transferred to team members [MDD10; SJ17]. The Grounded Theory study in part I presented similar results.

While some studies suggest that the Scrum Master is entirely transferred to Developers in more mature teams [Bäc19; SJ17], other studies find that one dedicated Scrum Master plays the role differently in more mature teams [GGJ19; MDD10]. For example, the Scrum Master is assumed to evolve from command-and-control behavior to a coach [GGJ19; MDD10].

Part I of this thesis examined the activities of a Scrum Master by applying Grounded Theory and identified nine leadership roles which change while the team matures. Based on qualitative semi-structured interviews I speculated that the Protector and the Moderator remain with one dedicated Scrum Master, while the other seven roles are gradually transferred to the Developers. The results of a Grounded Theory study are a new theory for future quantitative work [GS17]. Part II aimed at a quantitative support for a mature team predominantly playing the Scrum Master activities. Moreover, part II aims to contribute to understanding agile teams by providing a survey to quantitatively examine leadership in agile teams.

6.1.2 Changing Leadership in Agile Teams

To place part II into the research context I shortly recapitulate the findings of my literature review in Section 3.2.2.

Leadership is often characterised to change depending on the maturity of the team [Bäc19; GGJ19; MDD10; SJ17]. Moe et al. [MDD10] report on teamwork challenges of a newly implemented Scrum team over a period of nine month. They observe that initially team leadership was rather centred on the Product Owner and the Scrum Master. The Scrum Master even started to control team members which diminished team leadership and led to less motivation and trust of the Developers. While the team matured, the authors observed that team leadership advanced, such that team members started to take on more responsibility.

Even though several studies find similar results [Bäc19; GGJ19; SJ17], researchers do not agree on the extent to which the team plays the leadership activities of a Scrum Master over time. While some authors speculate that only some of the Scrum Master activities are transferred to the Developers [GGJ19; MDD10], other authors suggest that the dedicated Scrum Master becomes obsolete in more mature teams [Bäc19; SJ17]. While a study by Bäcklander [Bäc19] describes that often developers grow into the Scrum Master role over time, Moe et al. [MDD10] discover that team members rarely take over responsibility. Srivastava and Jain [SJ17] conclude that all team members should be able to take on the Scrum Master role in more mature teams.

Part I of this thesis suggests a set of 9 leadership roles of which 7 are gradually transferred to the Developers, while 2 of the roles remain with one dedicated Scrum Master. I discovered the roles Method Champion, Disciplinizer on Equal Terms, Change Agent, Helicopter, Moderator, Networker, Knowledge Enabler and Protector, which I summarize once again in Section 6.2.3 but are explained to a greater extent in Section 4.2.2. I name the nine leadership roles of a Scrum Master the 9-Factor Theory.

Since part I is a Grounded Theory based theory grounded in empirical qualitative data, the 9 leadership roles of a Scrum Master and how the role distribution unfolds in an immature as compared to a mature team has not yet been quantitatively analyzed. Based on previous research, I explore the 9-Factor Theory, and find that the Scrum Master varies with regard to the presence of the distinct roles and to the extent to which leadership is shared in different maturity stages.

6.2 Study Design

This section recaps the research question and company context and portrays the participants, the measurement, data collection and analysis of my study.

6.2.1 Research Questions

I build on the findings related to research question RQ1 and RQ2 of part I and suggest the following research questions:

- Which leadership roles does the Scrum Master play? (RQ8)
- Which leadership roles do the Developers play? (RQ9)
- Are leadership roles distributed between a Scrum Master and the Developers, and if so, is the role more often shared in mature as compared to immature teams? (RQ10)

6.2.2 Company Context and Participants

The data derived from the same corporation as the data used in part I. Yet, I did not approach the same people but different respondents in order to add reliability to my findings. Data was collected from the multi-national conglomerate Robert Bosch GmbH with more than 20 different sub-companies producing automotive, electrical and consumer industry goods. Scrum teams have the roles Product Owner, Scrum Master and Developers. Depending on the setting teams may have additional roles like a project manager, business owner, group leader or release train engineer. Yet, there is no company-wide standard.

The Scrum Master is a job title at the Robert Bosch GmbH. The person playing the committed Scrum Master varies among teams. For example, the role keeper can be a former developer or a former group leader. Often, the Scrum Master is called 'Agile Master' indicating that the role keeper should rather focus on team dynamics than on the Scrum method. Scrum Masters at the Robert Bosch GmbH are usually not disciplinary supervisors of agile team members, and were probably without authoritative power in the sample.

In total, *67 participants* took part in the study. 46 were from software development projects, 3 from software and hardware development, 4 from software development and IT and the remaining 14 from other topics (e.g. mechanical engineering, purchasing, human resources). 56.7% of the participants had been working more than 11 months with their colleagues.

The sample contained *37 Scrum Masters* of which 20 had at least 10 months of experience in the Scrum Master role. The remaining *30 participants were team members*. 14 team members stated that they were 9 or more members in their team. I did not measure this item for the Scrum Masters.

Due to confidentiality reasons, providing the team name was optional. 37 participants opted to enter their team name and related to *19 different teams from nine different business divisions* at the Robert Bosch GmbH. Since not all respondents inserted their team name, I could not map responses to teams and were only able to compare individual responses.

6.2.3 Measurement

The research questions guiding this study required a quantitative exploration of the 9-Factor theory. Each of the nine factors describes a leadership role. Besides evaluating the existence of different leadership roles, part II aimed at providing evidence that leadership roles are shared between a Scrum Master and the Developers and that the leadership roles are distributed differently depending on the maturity of an agile team.

I now briefly describe the 9 Factors. A deeper description is offered in

Section 4.2.2.

Factor MC (Method Champion): The role contains organizing meetings, teaching the method, support formulating tasks and setting goals, and discusses how to adapt the method during the Retrospective.

Factor DE (Disciplinizer on Equal Terms): Supports the team to keep to the rules, ensures that the team focuses on relevant topics and makes sure that team members attend the meetings. Discipline is accomplished via communication on a par.

Factor CO (Coach): Observes team members and uncovers which kind of behaviour is missing in a team to improve teamwork, provides feedback, and helps teams to find out what they wish to change and how to do so.

Factor CA (Change Agent): Serves as a role model, changes habits, and convinces newly established project teams of the agile way of working.

Factor HEL (Helicopter): Possesses the ability to see the bigger picture, to know who possesses the right skill for a certain task, to include relevant stakeholders and to structure work.

Factor MO (Moderator): Moderates all kind of meetings and builds a bridge between perspectives and domains.

Factor NET (Networker): Connects the team with relevant stakeholders from within and outside the organisation.

Factor KE (Knowledge Enabler): Realises which kind of knowledge the team needs, supports team members to acquire that knowledge and promotes iterative learning.

Factor PRO (Protector): Shelters teams from inappropriate requests from the Product Owner, managers, disciplinary leaders and other departments.

Items for Measuring the 9 Factors Based on the description of the Scrum Master roles as described in Section 5.2.1, I initially built a set of 67 items. Based on techniques rooted in pool items and item review [Rus09], after two revisions I reduced the initial set to 55 items, each connected to one activity of the nine different roles.

Each factor was covered by 4 to 9 different items. For example, the Disciplinizer on Equal Terms contained the following four items: *Supports*

Table 6.1: Maturity

Months	Team Member (N=29)	Scrum Master (N=36)
0-2	0	2
3-5	5	8
6-8	6	3
9-11	4	5
More than 11	14	18

team to keep to the rules. Helps team to focus on relevant topics. Makes sure members attend meetings. Communicates on a par. Yet, items are not grouped in the questionnaires, s.t. participants are blind to the existence of the factors. This helps avoid bias that could artificially form clusters.

Maturity To test maturity, I asked how many months the team had been working in an agile manner. The choice is inspired by Wheelan et al. [WDT03]. They found a significant correlation between the average number of months a team had been working together and the four group development stages [WH96], in which a mature team was perceived to be meeting 5.2 months or more on average (Stage 3=5.2 months on average; Stage 4=8.5 months on average). Based on previous results the question *How many months has your team been working in an agile manner?* provided five choices (0-2 months, 3-5 months, 6-8 months, 9-11 months, more than 11 months).

Self-Assessment and External Assessment Since teams and formal leaders often rate leadership behavior differently [CLP10], I conducted a self-assessment and an external assessment for evaluation of each item (leadership activity). Therefore, each item contained two Likert items: the self-assessment and the external assessment. More specifically, the Scrum Master conducted a self-assessment of the leadership behavior he or she believed to perform and an external assessment of the leadership activities he or she believed the Developers performed, and the Developers vice verse rated themselves and the Scrum Master.

Therefore, the participants answered each item twice (2*55): one to rate the Scrum Master and one to rate the Developers. The participants rated their perception of leadership activities displayed by the Scrum Master and the Development Team using a five-point Likert item with 1=strong disagreement that the activity was done by the respective party, 5=high agreement, and an additional option = Don't know/Not applicable. Questions were randomly ordered.

6.2.4 Data Collection

To assess the 9-Factor Theory I used a web-based survey tool provided by the Robert Bosch GmbH, as part of the agreement to run the study with them.

To invite Scrum practitioners to take part in my survey, I used my personal network within the Robert Bosch GmbH and a internal social business platform provided by the company. An invitation letter contained the link to the online survey and introduced the broader topic of the research and informed that data would be treated anonymously and that participation was voluntarily. Besides treating personal data confidentially on my side, participants had the opportunity to voluntarily insert their team name and their email address to receive their aggregated team results. This personal data was used for the respective team Retrospective only and for no scientific or management purpose, which was also emphasized in the invitation letter. Filling out the survey took approximately 15 minutes. With the exception of the personal data all questions were compulsory. The full questionnaire is available online [SGHW20b]. Due to confidentiality requirements by the Robert Bosch GmbH, the raw data cannot be provided openly.

6.2.5 Pilot study

Eight individuals filled out a pilot of the online survey and provided feedback on understanding the content of the items and the convenience to answer the survey.

Some participants had stated to be annoyed when they had to read one item twice on consecutive pages separately for the Scrum Master and the team members and the company had urged to build a questionnaire that would not take longer than 15 minutes to be filled out. Rating each item for both parties at the same time and on one page was considered to save time and to be more convenient.

Even though I had used the feedback for modification, drop out rate was 60% after launching the survey officially. Several participants delivered the feedback that reading all the items on one page was inconvenient. Therefore, I modified the questionnaire once again, and put the 55 items on three consecutive pages each containing an equal number of items.

This modification led to a loss of data, which I could not plan for with the tool supplied by the company, in 8 already fully filled-out responses. The modified survey accomplished 121 responses, of which 68 were completed while 53 did not reach the last item. I opted to retain only fully completed questionnaires rather than adding partial data. 16 respondents stopped after they had filled out the first block of items, while 22 respondents dropped out when reaching the first block of items and 15 individuals just opened the link without answering any of the questions. Once again, I received the feedback by participants, that the questionnaire was inconvenient to be read.

Due to the above-mentioned constraints I still kept the questionnaire the way it was designed. Also I cannot say with certainty why so many individuals decided to stop filling out the questionnaire. It may also be that they did not feel comfortable with rating Scrum Master and team members separately.

I removed the responses of one individual who rated every item with "agree," likely indicating a lack of motivation to participate in the study. This led to a total sample of 67 (55.37%) respondents.

6.2.6 Analysis

For each of the 9 factors I built a mean value by the related items for the Scrum Master and the Development Team separately. To avoid including individuals that had only answered a few items related to one factor, I included responses in the calculation of the mean value when individuals had at least answered n-1 items per role. That means, if a factor had 4 items, I only included individuals that had answered at least 3 of the items.

To assess whether leadership roles were shared between the Scrum Master and the Development Team I applied a similar approach as Zafft, Adams and Smith's [ZAM09] approach to measuring leadership distribution in self-managed teams. Applying a 5-point Likert scale (1=strongly disagree, 5=strongly agree), they suggest a leadership behavior to be present when someone scores higher than 4.0 [ZAM09]. In the analysis, I considered a factor to be embodied by the Scrum Master or the Development Team if the respective party rated 4.0 or higher. If one participant rated both, Scrum Master and Development Team, in one factor higher than 4.0, the respective role was considered to be distributed between both parties within one team.

If at least five of the nine factors were found to be shared within the same team, I considered the Scrum Master role to be shared between the Development Team and the dedicated Scrum Master.

6.3 Results

The results are structured as follows: After referring to external and self-assessment, I will answer the three research questions in consecutive order.

External and Self-Assessment The average mean for the nine factors revealed that the Scrum Master tended to rate herself higher than the Development Team rated the respective Scrum Master, while the Scrum Master tended to rate the Development Team lower. One exception was the Networker which the Scrum Master rated slightly higher than the Development Team rated itself. Likewise, I found that the team members tended to rate themselves higher than the Scrum Master rated them, while they tended to rate the activities performed by the Scrum Master lower.

Table 6.2: Descriptive Statistics for the 9 Factors (Scrum Master)

	Scrum Master				
Factor	N	Mean	Std. deviation	n*	h**
MC	67	4.15	.56	47	70.15%
DE	67	4.18	.55	49	73.13%
CO	66	4.09	.73	46	69.69%
CA	61	3.95	.65	37	60.66%
HEL	64	3.73	.68	28	43.75%
MO	67	4.07	.63	49	73.13%
NET	65	3.70	.86	30	46.15%
KE	63	3.62	.76	22	34.92%
PRO	62	3.70	.88	32	51.61%

*n describes the absolute frequency of a factor rating higher than 4.0.
**h describes the relative frequency (n/N per row).
Note: Each column contains summarized results and refers to answers by Scrum Masters and the agile team taken together.

6.3.1 Scrum Master

The first research question is: *Which leadership roles does the Scrum Master play?* (RQ8)

To be able to give evidence on the Scrum Master performing one of the nine leadership roles, the mean value of a factor has to be higher than 4.0 (explained in Section 6.2.6). The mean value for four factors is higher than 4.0, namely Factor MC, DE, CO and MO, and more than two third of the Scrum Masters score high on them. Factor CA, HEL, NET and PRO are linked to about half of the Scrum Masters. Only about one third have a mean value higher than 4.0 regarding Factor KE. More information in Table 6.2.

Therefore, I answer RQ8 and find that a majority of the Scrum Masters play the Method Champion, Disciplinizer on Equal Terms, Coach and Moderator, while the Change Agent, Helicopter, Networker and Protector is played by merely about half of the Scrum Masters and the Knowledge Enabler is performed by only about one third.

Table 6.3: Descriptive Statistics for the 9 Factors (Development Team)

	Development Team				
Factor	N	Mean	Std. deviation	n*	h**
MC	60	3.19	.67	7	11.67%
DE	65	3.83	.52	32	49.23%
CO	64	3.58	.59	17	26.56%
CA	56	3.56	.52	16	28.57%
HEL	62	3.72	.54	24	38.71%
MO	62	3.72	.48	22	35.48%
NET	62	3.44	.81	21	33.87%
KE	58	3.58	.66	20	34.48%
PRO	53	3.10	.84	10	18.86%

*n describes the absolute frequency of a factor rating higher than 4.0.
**h describes the relative frequency (n/N per row).
Note: Each column contains summarized results and refers to answers by Scrum Masters and the Development Team taken together.

6.3.2 Developers

My second research question is: *Which leadership roles do the Developers play?* (RQ9)

To be able to give evidence on the Developers playing one of the nine roles, the mean value of a factor has to be higher than 4.0 (explained in Section 6.2.6). Table 6.3 illustrates that all mean values of the nine factors related to the Development Team are lower than 4.0. Therefore, one could claim that team members tend to not play the leadership roles. Yet, almost 50% of the Development Teams score higher than 4.0 for Factor DE. Between 30% and 40% perform Factor HEL, MO, NET and KE. Factor MC and PRO are rarely aligned to the Developers.

Based on my results, I answer RQ9 and find that the Development Team tends to not play the leadership roles. About half of the team members perform the Disciplinizer on Equal Terms, while only about one third perform the Helicopter, Moderator, Networker and Knowledge Enabler. The Method

Table 6.4: Distribution of the 9 Factors

Factor	Shared	Only Scrum Master	Only Devs	No one	N	Total %
DE	43.30%	29.90%	4.50%	22.40%	67	100.00%
MO	31.30%	41.80%	1.50%	25.40%	67	100.00%
HEL	28.40%	13.40%	7.50%	50.70%	67	100.00%
CO	25.40%	43.30%	0.00%	31.30%	67	100.00%
NET	22.40%	22.40%	9.00%	46.30%	67	100.00%
CA	19.40%	35.80%	4.50%	40.30%	67	100.00%
KE	16.40%	16.40%	13.40%	53.70%	67	100.00%
PRO	14.90%	32.80%	0.00%	52.20%	67	100.00%
MC	10.40%	59.70%	0.00%	29.90%	67	100.00%

Note: Each column contains summarized results and refers to answers by Scrum Masters and the Development Team taken together.

Champion, Coach and Protector are performed least often by the teams.

6.3.3 Distribution of the 9 Factors between Scrum Master and Development Team

The third research question is: *Are leadership roles distributed between a Scrum Master and the Development Team, and if so, is the role more often shared in mature as compared to immature teams?* (RQ10)

If a participant scores a factor for both the Scrum Master and the Developers higher than a mean value of 4.0, the factor is considered to be distributed between the Scrum Master and the Development Team. While Factor DE, HEL and MO are distributed in 30% to 40% of the teams, Factors MC, CA, KE and PRO are distributed in 10% to 20% of the teams. Table 6.4 shows an overview on the distribution for each of the nine factors, starting with the most frequently shared Factor DE to the least frequently shared Factor MC.

If a respondent scores a mean value higher than 4.0 for at least five of the factors for both, Scrum Master and Development Team, the Scrum Master role is considered to be distributed between the Developers and the dedicated

Scrum Master. 20.90% of the respondents share the Scrum Master role.

38.5% of the teams that had been working 3-5 months in an agile manner shared the Scrum Master role, 11.11% of the teams rating 6-11 months shared it and 18.8% of the teams rating more than 11 months shared the role. Therefore, teams that had been working for 3-5 months tended to share the role by 20 percentage points more than teams that had been working for 11 months or more, and by 27.39 percentage points more than teams that had been working in an agile way between 6-11 months.

Furthermore, I check if some Development Teams perform the Scrum Master predominantly, such that the Development Team scored for 5 factors higher than 4.0, while the Scrum Master scored for less than 5 factors higher than 4.0. I did not find such a case in the data.

Based on these results I answer RQ10 and claim that leadership roles can be shared, yet, some roles are shared more often than others. While I find that the Disciplinizer on Equal Terms is most often shared between the Developers and the Scrum Master, I find that the Method Champion, Coach and Protector are rather centred on one dedicated Scrum Master.

Furthermore, the distribution of the Scrum Master role varies in different maturity stages. I find that teams who share the role had most often been working in an agile way between 3 to 5 months. Therefore, the role was rather shared in immature teams. Furthermore, I did not find a single team in which the Scrum Master role was centred on the Developers.

6.4 Discussion

My study aimed at exploring the presence of and the change in the 9-Factor Theory as described in part I. Based on descriptive statistics, I found that the nine different roles are performed to a varying extent:

While the Scrum Master rates highest in the Method Champion, Disciplinizer on Equal Terms, Coach and Moderator, the Development Team scores highest in the Disciplinizer on Equal Terms, Helicopter, Moderator, Knowledge Enabler and Networker. Both, Scrum Master and Development

Table 6.5: 3 proposed Clusters of the 9-Factor Theory

Cluster	Leadership Role (Factor)	More important to
Psychological Team Factors	Method Champion (ME) Coach (CO) Moderator (MO)	Scrum Master
Product-Related Factors	Disciplinizer on Equal Terms (DE) Helicopter (HEL) Knowledge Enabler (KE)	Development Team
Organisational Factors	Change Agent (CA) Networker (NET) Protector (PRO)	It depends

Team, tend to perform the Protector less often than the other roles.

Based on this result, I suggest to broaden the 9-Factor Theory. My results indicate that the nine factors can be further grouped into three clusters: psychological team factors, product-related factors and organisational factors. I will now elaborate on this idea based on empirical results.

Factor MC, CO and MO rather focus on internal socio-psychological team mechanisms, while Factor CA, NET and PRO involve an external focus towards the organisation. Factor DE, HEL and KE are rather product-related and aim at continuous learning and knowledge sharing. The Scrum Master scores higher in roles related to psychological team factors (e.g. Method Champion and Coach). The Development Team scores higher in product-related factors (e.g. Helicopter and Knowledge Enabler). Roles that bridge the organisation with the team were played more often by the Development Team regarding the Networker, but less often regarding the Protector.

Moreover, about half of the Development Teams did not play the Protector, the Change Agent or the Networker which are linked to the organisational factors. In rather bureaucratic organisations, as in our case, it might be more difficult to perform the roles related to bridging the organisation and the team. A traditional environment rather focuses on hierarchy as opposed to protect the team from management and on departmentalised structure as opposed to network with each other independent from formal structures [NMM05].

I speculate that if a Scrum Master played the Protector to a larger extent, the team members would take over the leadership roles more often. The Protector provides hierarchical free space within which developers feel safe to take on the divers roles [SHW19a].

Furthermore, 53% of the teams did not perform the Knowledge Enabler and about 51% the Helicopter. A possible explanation for our results would be that either the Scrum Master considers product-related roles to not be part of the job description since the Developers are expected to self-organise their work, or it is more difficult to play the respective roles in a bureaucratic context since that company type is build on experts with specialized skills as opposed to cross-functional knowledge sharing [NMM05]. This may be supported by the Developers scoring equally low on this factor.

This study also aimed at exploring the 9-Factor Theory in relation to maturity. The 20% of the teams that did share the Scrum Master role, provided support for the suggestion that the Scrum Master role is distributed differently in different maturity stages. Teams that had been working in the agile manner for 3 to 5 months and more than 11 months shared the role most often.

This finding fits with the maturity model by Tuckman [Tuc65]: Teams after 3 to 5 months tend to be in the storming phase, within which teams are not sure about who plays which role within the team. Therefore, both, Development Team and Scrum Master, perform the nine leadership roles. Teams working in an agile way for more than 11 months could already have reached the performing phase within which roles are played according to the situation and less linked to one dedicated role keeper.

Yet, I did not find any Development Team that played the Scrum Master role to a larger extent than the dedicated Scrum Master. Therefore, my results do not point at the direction that the formal role keeper steps back from the role as suggested by several studies [Bäc19; SJ17]. This finding also fits with earlier claims that teams in organisational settings rarely develop into high performing teams that take on roles spontaneously [MMZ01]. I therefore propose that in most of the teams the dedicated Scrum Master does not become obsolete over time but rather changes the primary role during the different phases of team development.

Another explanation of the results could be that neither Scrum Master nor Development Team but someone else took over the role. As described in Section 6.2.2 agile teams and the aligned agile roles vary among different settings at the Robert Bosch GmbH. It might be that some of the nine leadership roles are also played by the Product Owner or disciplinary supervisor. However, those roles were neglected in our quantitative exploration. The last paragraph of the practical implications provide suggestions on how to deal with this in company settings.

6.5 Practical Implications

I found that the leadership roles were rather centred on the Scrum Master. In the following I thus suggest how to develop the Scrum Master description in company settings. Section 6.4 proposed to group the Scrum Master description into three clusters: psychological team factors, organisational factors and product-related factors.

While some practitioners suggest that the Scrum Master should play product-related roles, others state that interference on a technical level hinders self-organising teams. I suggest that every team should discuss on its own, to which degree it needs product-related support by a Scrum Master. Yet, a Scrum Master who performs product-related roles builds an understanding of the respective product, thus, can also more easily bridge the agile team with the processes, requirements, tools and standards of a rather bureaucratic surrounding.

For example, the Scrum Master can be a mouthpiece of the team to discuss with the management which processes and requirements of rather traditional project management are still needed despite the team working in an agile way, and which ones are rather unnecessary and hinder the progress of the team. The Scrum Master can argue which tools and processes the team needs to work in a more agile way. Also, taking over product-related roles improves understanding when to protect the team, e.g. from re-prioritization, and when to give in and allow to re-arrange planning due to changes in requirements on organisational level.

Thus, the Scrum Master supports the organisation to gradually evolve into a more agile place. Yet, I acknowledge the balancing act of a Scrum Master to support the team in product-related matters and to serve as a coach at the same time. The Scrum Master continuously needs to serve as a coach and support the Developers to learn how to take on the divers roles.

Agile teams in a traditional industrial corporation may not be used to take on leadership activities as a whole team. Yet, if the leadership gap is not filled by the Developers, there is the risk of a leadership vacuum, in which no one takes over leadership roles. This may lead to less performance.

Nevertheless, I found that the Scrum Master and the Development Team tend to play organisational factors to a lesser extent, and encourage managers even further to build an agile friendly surrounding within which the organisational factors can be performed. These factors are necessary to integrate the agile team into the organisational setting, such as having access to relevant stakeholders and information, reducing interfaces and efforts for alignment and building trust between agile teams and traditional structures. Consequently, motivation and progress of agile teams will increase even further. Yet, organisations also need to understand and accept that sometimes developers do not want to take on leadership roles.

Therefore, companies should use my questionnaire to reflect upon the role distribution in their specific industry background and organisational environment relevant to their team. There might be roles beyond the Scrum Master and the Development Team that take on the leadership roles. Thus, I suggest to not only focus on leadership sharing among the Scrum Master and the Development Team but to broaden the perspective. I propose to use the leadership roles and aligned activities to determine if they are covered by any 'job title' in the setting, which might be the Product Owner or the disciplinary supervisor. After all, the agile way of working is not about establishing a standard regarding which job title plays which leadership role but about making sure that the needs of an agile team are covered in any given situation. Since teams mature and agile settings vary, teams need to find a context-dependent equilibrium of leadership sharing. Therefore, each team has to discuss on its own how to divide leadership activities among each other. Furthermore, since context changes, teams need to discuss regularly upon who takes on which leadership role in a given situation. Practitioners will understand respective leadership needs, learn to balance and evolve them, and thus, improve teamwork.

6.6 Limitations and Future Work

In the following I will suggest future topics for research while referring to limitations of this study.

Objectivity: since I conducted an online survey, I assume a low level of social response bias. Yet, respondents were allowed to insert their email address for receiving their team results. This could lead to a social response bias in such a way that respondents wanted to rate high in the Scrum Master activities.

With 67 participants our sample is limited in size and prevented us to perform a psychometric evaluation of the tool, limiting our confidence in the tool validity. A psychometric evaluation of the tool would not be a familiar step in software engineering studies, so we see this as a missed opportunity rather than a limitation. Future studies should aim for a bigger sample size that allows to perform an exploratory factor analysis, thus quantitatively clustering the factors. As the theorized 9 factors might be difficult to test psychometrically, I suggest future studies to test the three suggested clusters in Section 6.4, thus, allowing for testing agile team behavior along three variables instead of nine.

Since the drop-out rate for this study was quite high, for future studies I suggest, to rate the Scrum Master and the Development Team each on separate consecutive pages. Therefore, participants will have to answer six different pages of questions. This will take more time, yet, may lead to a more convenient experience to fill out the questionnaire, and thus, increase the number of responses.

Moreover, even though each business division operates within a different sub-culture and industry context, still all teams were from the same conglomerate. Even if this study is clearly placed as an exploratory one, I want to highlight that we cannot claim the results to be universally applicable. I suggest a larger sample drawn from different companies with different industry backgrounds to extend this study in the future.

Moreover, almost 50% of the team members stated to be 9 or more persons in their team. I was not able to control for this variable since I had not asked the Scrum Master on their number of team members. Larger groups are found to be less likely to evolve into a mature team [WDT03]. Future testing should take this into account.

Data points at an evolving Scrum Master role in relation to maturity. However, maturity was rated by the number of months each team had been working in an agile way. I cannot claim with certainty that the time a team has been working in an agile manner is related to maturity stages. Furthermore, I have not conducted a longitudinal study but compared different teams which had been working a varying amount of time in the agile manner.

Future testing should refer to the maturity stage by Wheelan [WDT03] to examine the 9-Factor Theory for a valid measurement of group maturity, investigate time and group development in relation to varied company types and sizes, as well as in a longitudinal study.

Chapter 7

Discussion and Conclusion

This chapter summarizes the findings and discusses them critically while referring to existing evidence. I highlight my contribution to research and suggest practical implications.

Section 7.1 summarizes the results of part I and part II while referring to the research questions RQ1 - RQ10. The discussion in Section 7.2 elaborates on the value of one individual performing the Scrum Master, the challenges of shared leadership in established companies and suggests a new role description of the *Agile Master*. This discussion provides groundwork for the practical implications for the Development Team, Scrum Master, Product Owner, Management and organisation in Section 7.3. Section 7.4 refers to limitations of the study and Section 7.5 suggests future research which has not yet been referred to in part I and part II.

7.1 Summary of the Research Findings

Empirical studies have suggested that the role of leadership evolves while the team matures [GTF17; MDD10; SJ17]. Some authors state that team members in a more mature team also take on leadership roles [MDD10;

SJ17]. Yet, up to my knowledge, no scientific study has examined how the leadership role changes in an agile team over time. This thesis explored how leadership enables the Developers to take on leadership roles and adapt to the agile way of working.

I focus on the leadership role of a Scrum Master and explain how the team can evolve from one dedicated leader to self-organisation in established organisations.

Through a Grounded Theory based study (part I) and a quantitative exploration (part II) this thesis provides empirical support for a change in leadership using the example of the Scrum Master. Furthermore, I advance the theoretical underpinning of agile leadership by integrating more research on role theory and team leadership in self-managed teams into research on leadership in agile software development teams. Based on theory and empirical findings I build a substantive theory and a quantitative test on changing leadership in agile teams. I therefore contribute to understanding leadership in agile teams.

In the following I summarize my research findings. I briefly recapitulate part I and part II while referring to the respective research questions in consecutive order.

Part I reported on the results of a Grounded Theory study. Based on observations and qualitative interviews with 75 Scrum practitioners from 11 different business divisions of the Robert Bosch GmbH I answered research questions RQ1 to RQ7.

Which roles does the Scrum Master play to support the team to work in an agile way? (RQ1)

I identified the following leadership roles: Method Champion, Disciplinizer on Equal Terms, Coach, Change Agent, Helicopter, Moderator, Networker, Knowledge Enabler and Protector. I label the set of nine leadership roles the *9-Factor Theory*. A description of each role can be found in Section 4.2.2.10.

In which way do the Developers take on the Scrum Master role over-time? (RQ2)

The semi-structured interviews with Developers, Scrum Masters and Product Owners provided examples for the Developers taking over 7 of the 9 leadership roles over time: the Method Champion, Disciplinizer on Equal Terms, Coach, Change Agent, Helicopter, Networker and Knowledge Enabler. I did not come across examples in which the Developers played the Protector and the Moderator.

How are roles transferred from the Scrum Master to the Developers? (RQ3)

The leadership roles are transferred in three consecutive steps which I name the *role transfer process*. First the Scrum Master is a role model and demonstrates the nine leadership roles while the Development Team is observing the behavior. Secondly, the Scrum Master provides a leadership gap, namely a hierarchical-free space within which neither the dedicated Scrum Master, managers, the Product Owner nor any formal leader role interferes. This allows the Developers to take on leadership roles themselves. Team members claim and grant leadership roles depending on the situation. In the last step, the Scrum Master only plays the leadership roles when still needed, while the Development Team performs most of the nine roles.

What is the underlying internal team environment required for the role transfer to occur? (RQ4)

The role transfer requires an internal team environment of communication on equal terms, psychological safety, transparency, shared mental models, team orientation, team potency, self monitoring and team learning. These team features help the Development Team to understand the leadership roles and agree upon sharing the roles. How each of these factors influences the role transfer is described in Section 5.2.3.1.

How can the Scrum Master foster the internal team environment? (RQ5)

By facilitating the Retrospective the Scrum Master helps the Developers to create a supportive internal team environment. The Retrospective contributes to building a shared understanding on leadership in agile teams and empowers to take action.

Which expectations on the Scrum Master limit the changing leadership role? (RQ6)

The Scrum Master is torn between fulfilling expectations deriving from a bureaucratic way of thinking and the agile way of thinking which led to role conflicts. For example, sometimes managers expect the Scrum Master to do monitoring, even though the agile way of working describes that the Development Team monitors itself. As a consequence of the role conflicts, the Scrum Master is torn between clinging to the leadership roles and handing them over to the Developers. If the Scrum Master decides to fulfill expectations deriving from a bureaucratic system, the role transfer is diminished.

How do organisational culture and structure influence the role transfer? (RQ7)

Organisational culture linked to high power distance (e.g. command-and-control behavior), specialised culture (e.g. very knowledgeable experts) and functional departmentalised structure (e.g. departmental silos) reduces the role transfer.

Based on the results of part I I suggest the *Agile Matching Theory* which implies that an internal team environment and organisational context need to match for the role transfer to occur. Section 4.2.5 provides a detailed description of the theory.

Part II reported on a quantitative exploration with 67 participants from more than 19 different Scrum teams of the Robert Bosch GmbH. This study aimed to test the presence and change of the nine leadership roles. **Which leadership roles does the Scrum Master play? (RQ8)**

Our data revealed that a majority of the Scrum Masters play the leadership roles of the Method Champion, Disciplinizer on Equal Terms, Coach and Moderator, while the Change Agent, Helicopter, Networker and Protector is played by merely about half of the Scrum Masters and the Knowledge Enabler is performed by only about one third.

Which leadership roles does the Development Team play? (RQ9)

Our data displayed that the Development Teams rarely take on the leadership roles. About half of the teams perform the role of Disciplinizer on Equal Terms, while only about one third perform the roles of Helicopter, Moderator, Networker and Knowledge Enabler. The Method Champion, Coach and Protector are performed least often by the Developers.

Are leadership roles distributed between a Scrum Master and the Development Team, and if so, is the role more often shared in mature as compared to immature teams? (RQ10)

The quantitative exploration revealed that a few teams share the leadership role and that the distribution of the leadership roles varies in different maturity stages. I find that teams who share the Scrum Master role had most often been working in an agile way between 3 to 5 months. Therefore, the role was rather shared in immature teams. Furthermore, I did not find a single team in which the Scrum Master was centred on the Development Team.

Since part II finds that different roles are performed to a varying extend I suggest to divide the 9-Factor Theory into the three clusters: 1) psychological team factors, comprised of the leadership roles Method Champion, Coach and Moderator 2) product-related factors, comprised of the roles Disciplinizer of Equal Terms, Helicopter and Knowledge Enabler 3) organisational factors, comprised of the roles Change Agent, Networker and Protector. I suggest this clustering as we find that organisational factors are rather rare in Development Teams, product-related factors are more often performed in selected teams, while psychological team factors are rarely transferred from the Scrum Master to developers.

7.2 Discussion

7.2.1 The Value of a dedicated Scrum Master

One recurring question is who should take on the dedicated Scrum Master role. Some researchers argue in favor of a rotational Scrum Master in mature teams [SJ17; SSD16] which implies that any developer in the team can take over the role. Other studies reveal that some sort of dedicated neutral facilitator is needed to establish a healthy team environment [Bar93]. Our data revealed that only a few teams shared the Scrum Master role and no individual Development Team predominantly performed the leadership roles. I therefore argue in favor of one dedicated person taking on the leadership role of a Scrum Master and do not support the claim of a rotational Scrum Master. Even though I suggest a role transfer process, I propose that one dedicated leader is always needed in an agile team. Yet, I do not agree that one person keeps all leadership roles [QFTM90] but suggest that the Scrum Master should rather be shared between one dedicated role keeper and the Developers.

Based on the quantitative exploration I propose that some roles are more suitable to be shared than others. The psychological team factors (e.g. Method Champion and Coach) and the Protector rather remain with the dedicated Scrum Master over time, while the product-related factors (e.g. Helicopter and Knowledge Enabler) and the Networker are gradually shared between the committed Scrum Master and the Developers. Whoever feels willing and competent in a given situation takes over a required role [HNM12b].

Some of the roles, such as the Helicopter, were reported to stick with the same team member after a while. Teams stated that the person who was best in a specific role would always play this role. Yet, sometimes clinging to roles created territories in which only that specific person was allowed to play that role and other team members lacked the opportunity to perform each role to an equal amount (Section 5.2.3.4). Some studies reveal that self-managed teams who share roles perform better [ZAM09], and therefore, one can speculate that committed role keepers are not always beneficial for the team. However, role sharing also varies in different maturity stages [Yan96] and role distribution needs to fit with the respective context and maturity stage. I therefore highlight that the Scrum Master should reflect regularly with the team who takes upon which role and whether or not team members would prefer to share specific leadership roles. Role clarity is important for establishing shared expectations [BP19] on how to distribute leadership roles.

The Retrospective helps to reflect upon expectations. Despite its importance for team development, the Retrospective was found to be neglected in the absence of the Scrum Master [HNM12b]. An agile team needs a committed Method Champion and Coach who facilitates the Retrospective and contributes to a healthy internal team environment during the Retrospective. A supportive internal team environment encourages teams to take on leadership roles and therefore work in a more agile way.

Furthermore, teams need a Protector who shelters the team from inappropriate disturbances by Product Owner or management [Hod13]. This is especially important in a rather bureaucratic environment that is used to command-and-control mode. Team members in such an environment may lack in the legitimate right to play the Protector which may lead to less focus and consequently diminished work progress. Therefore, a dedicated Scrum Master is needed who has the legitimate power to perform the Protector.

The quantitative exploration revealed that not all of the leadership roles were performed to an equal amount by the committed Scrum Master, while also the Development Team did not take over the roles which were neglected by the Scrum Master. For example, the Protector was only performed by about half of the Scrum Masters and by no developer. While I acknowledge that sometimes the context does not require certain roles, I still think teams would benefit from a dedicated role keeper playing the distinct leadership roles to a larger extend. All of the leadership roles aim at improving the agile manner and the agile way of working helps teams to develop high-value products [CH01].

Moreover, team mechanisms and organisational environments are complex and understanding them requires a lot of experience and practice. One committed person should continuously develop coaching and method skills further to play the Method Champion and Coach and increase knowledge about the organisational context in order to play the Protector and Change Agent. The committed Scrum Master should be trained in psychological team mechanisms and organisational factors to support the team in an appropriate manner. Continuous learning and development requires time and cannot be done by all developers to an equal amount. However, this would be required if the Scrum Master role was rotated among team members.

A dedicated leadership enabler is a neutral person who fosters a supportive internal team environment and leadership sharing, knows the organisational setting, links the team to it and develops it further, protects the team and continuously takes time and effort to enhance these skills and capabilities further.

I would like to emphasize that it is not only important that one dedicated Scrum Master encourages developers in taking on leadership roles actively, but also needs to step back by providing a leadership gap. While much research emphasizes the need to intentionally 'motivate' followers (e.g. [PHP15]), I would like to support the claim by Charles C. Manz [Cha91] that teams are intrinsically motivated, and do not need to be motivated. A more mature Development Team which is built upon a proper team design [GL20] and embedded in a fitting environment with a compelling vision will not hesitate to take on leadership roles when facing the leadership gap.

7.2.2 Leadership in Agile Teams in Established Companies

The Grounded Theory study discovered nine leadership roles that are shared between one dedicated Scrum Master and the Developers. The descriptive exploration indicated that some roles were shared less often than others and that not all roles were performed by developers. One explanation emerging from the qualitative interviews is that role conflicts deriving from a mismatch between bureaucratic culture and the agile manner diminished the role transfer. Thus, a bureaucratic environment is a major challenge for evolving leadership in agile teams.

Several studies indicate that a lack of external support limits leadership in teams [HB13; HN17; MDD09]. For example, management can diminish self-organisation easily by micro-management (e.g. [HB13; HM16; HN17; MAD12]). Project managers that are responsible for results struggle with transferring leadership to the team and experience tension [Tay16]. Similarly, I found that the Scrum Master felt the need to protect the team from disturbances, while simultaneously coping with rather traditional requests by Product Owner, manager or other stakeholders, e.g. reporting. This resulted in role conflicts and restricted team leadership.

Moreover, our data revealed that some settings did not change the organisational structure. Instead of replacing traditional roles, organisations add agile roles on top of existing traditional roles. Therefore, even more roles approached the Developers. Consequently, agile teams handled many requests at once which made them lose focus and therefore less productive. Moreover, in settings where group leaders and project managers still existed, the Scrum Master had less power, because the role keeper was sometimes merely considered to be a personal assistant rather than a leadership enabler. This diminished the Protector and Change Agent role. Also a study by Gren [GL20] concludes that existing structures in bureaucratic companies are sometimes renamed by agile terms, yet, teams in such an environment do not develop the agile way of working.

Therefore, even though companies aim at implementing agile teams, team leadership will remain low if a company does not simultaneously abandon bureaucratic organisational culture and structure. To which extent the organisational context must change to allow for more leadership on team level is an ongoing discussion among researchers and practitioners alike.

Simply replacing the traditional positions by agile roles is considered not to be the appropriate solution to develop the agile manner [GGJ19; HB13; HN17; Pri10]. The sample predominantly derived from companies that are active in the automotive industry which develops products for a safety critical environment. Due to legal requirements, it appears to be difficult to change the organisation towards a more agile way. However, newly established companies in the same industry do have more flexible structures and a learning culture. For example, they think more in features and continuous delivery instead of delivering a finished product. To compete with emerging companies and rapid product development, a fundamental change of the way established companies develop products is inevitable.

Research on agile transformation tends to argue in favor of a gradual process to adapting to the agile manner [GL20; HN17]. Grigg [Gri10a] describes a gradual process by referring to the transformation from a command-and-control organisation to self-organised teams in three different stages over a period of several years. The author claims that it is very difficult to make employees take over more responsibility and they need to get used to it step-by-step. The author suggests to gradually provide more rights to the team while attending supportive training, e.g. how to use certain tools. Rather than exchanging traditional methods with agile methods, organisations are advised to harmonize agile and traditional development depending on a given situation and the context [BT05; GL20; HN17].

The Agile Matching Theory describes that the transfer of leadership roles depends on the fit between contextual factors and agile team features. I claim that distribution of leadership between the dedicated Scrum Master and the Developers is not about whether or not transferring leadership roles but about finding an equilibrium of role sharing while taking the organisational context into account.

However, this does not mean that the context should remain bureaucratic since culture and structure in established companies need to evolve to ensure competitiveness. Instead the context should be examined critically to search for opportunities for gradual change. Designing the context of the self-organised team right in the first place was identified to be a key leadership responsibility [GL20; HH02a; Wag97]. organisations have to make sure that the agile team is implemented in an agile-friendly environment, such that team members feel encouraged to take over leadership roles.

I suggest that instead of over-emphasizing changing Scrum Master behavior and developer behavior, leaders should critically examine the context within which agile teams operate and gradually adjust it such that teams can work in a more agile way. Organisational context and team behavior are interdependent and have to evolve together in order to diminish role conflicts and therefore, empower the team to take on more leadership roles. This equilibrium has to be re-examined and adapted iteratively since the team and the organisation gradually evolve.

To make agile working a success, not only developers need to change behavior but also management [HB13; MDD09]. Change on management level includes the way resources are allocated [HB13], adjusting the backlog to the strategy on organisational level [MDD10], handing over decision-making power to the team [Hod13] and a *minimum* of required processes. The vision of a project should be clear and access to required resources should be guaranteed [Wag97].

Moreover, a learning environment is found to increase acceptance of the agile mode of working [CT09]. Therefore, management should encourage a learning organisation [GEG08], which involves continuous integration, an inspect and adapt approach, learning from failures and flat hierarchies within which team members feel motivated to take on leadership roles [Hod13]. Furthermore, management should not jump into the leadership gap when provided by the Scrum Master but allow the Developers to drive.

7.2.3 New Role Description: Agile Master

Bäcklander [Bäc19] describes that the Scrum Master has evolved from facilitating the Scrum process to caring for team dynamics since its implementation in the company setting. I found similar results. The Scrum Master at our company setting focused on supporting context-dependent adaption of the agile manner instead of facilitating the Scrum method according to the book.

Even though many teams had left the Scrum process behind, I found that the team still applied agile methods, such as ScrumBan or Kanban, or had developed team-specific adaptations of agile methods. Those teams still had a Scrum Master who helped the Developers to collaborate and to apply a suitable agile method in their specific context. Often the teams did not call the role keeper Scrum Master but Agile Master.

Focusing on human behavior is more important than keeping a certain agile method [CH01]. Our results indicate that the Scrum Master in a company setting actually does more than facilitating agile methods and also evolves and cannot be considered a static concept. I acknowledge that applying a fitting agile method in a given team context is important and label the respective role the *Method Champion*. Yet, I discovered eight more roles that enable the team to work in an agile way. I summarize the set of 9 leadership roles the *9-Factor Theory*.

Based on my results I suggest to give a new name to the term Scrum Master. I suggest to label the role Agile Master. By doing so, I hope that practitioners will perceive the role as someone who serves as a leadership enabler to the team and the organisation more than someone who focuses on facilitating the Scrum process. I will provide a description of the *Agile Master* in the following.

The Agile Master is a peer to the Developers but does not take part in operational activities of the respective team. The dedicated role keeper is a leadership enabler who enables the team to adapt the agile manner while taking the context into account. The Agile Master plays a set of nine leadership roles (9-Factor Theory), all supporting varying features of agile teams which relate to self-organisation, cross-functionality and iterative learning.

The 9-Factor Theory can be clustered into psychological team factors, product related factors and organisational factors. While the Agile Master rather takes care of psychological team factors and organisational factors, the product related factors are rather performed by the Developers.

A team aims to share the leadership roles. However, how the Agile Master evolves and in which way the role is shared depends on the maturity, the internal team environment, the structure and the culture of a given setting. Furthermore, the Agile Master needs to provide a leadership gap within which developers feel capable and willing to take on leadership roles. In a more mature team, the dedicated Agile Master plays the role to a lesser extent but does not become obsolete.

The Agile Master facilitates the Retrospective which generates discussions on current challenges, leadership needs and the preferred extent of role distribution. Team external leaders can also be invited to the discussions on leadership needs and expectations. As a result, the team builds a shared understanding on leadership in a given context. Yet, the aim is not to share all roles to an equal amount but to find an equilibrium of role sharing according to the unique needs and context of the given team. Simultaneously all people involved continuously and critically examine the context for opportunities of changing towards a more agile place.

Moreover, role sharing between the Agile Master and the Developers requires support by the Product Owner, disciplinary supervisor and other stakeholders. The person who is in touch with the customer needs to provide money, moral support and a clear vision and strategic goal, while also offering a high level of freedom and trust to do the work and to reach the common goal.

7.3 Practical Implications

The following chapter provides practical implications of our findings. I provide suggestions for the Developers, the Scrum Master, the Product Owner, the management and the organisational culture and structure.

7.3.1 Implications for the Developers

When individuals from rather traditional development companies start working in agile teams they have to learn a new way of leadership in teams, which will lead to slower delivery of work products at the beginning. Management should grant sufficient time to teams to enable them to regularly reflect upon the leadership roles during the Retrospective. Team members must become convenient with interacting with their team and managers in a different way, and to develop the courage to bridge the leadership gap when provided. Teams need time to try the roles and learn them, possibly by failure. Just like any newbie in a formal leadership position needs time and is given time to learn the role, Development Teams need time to learn the leadership roles of Scrum.

I suggest the practice of the Retrospective to be a core element to enable the role transfer. The reasoning behind it is that the Retrospective establishes a beneficial internal team environment. For example, when talking about personal matters during the Retrospective, team members establish psychological safety within which they feel safe to take on leadership roles. The Retrospective serves as a learning opportunity within which team members discuss about the various leadership roles, the leadership gap, the opportunity to take on leadership roles and what would need to change to develop towards more shared leadership. Furthermore, the agile team should occasionally invite stakeholders and managers to the Retrospective. This allows them to build a shared understanding of working together and a shared leadership culture.

I also found that not aligning responsibilities at all leads to insecurity about roles and responsibilities. This will lead to a leadership vacuum within which no one takes on responsibility, at worst. Therefore, I urge practitioners to talk openly about roles and responsibilities and to agree on having specified owners for some topics and on having shared ownership for other topics.

Last but not least, Development Teams need to continuously discuss about what they would need to perform all roles. I have found that taking on leadership is not only about being capable and willing to fill the leadership gap but also about having access to the respective rights to fill the gap. For example, conglomerates sometimes only grant specific team members access to information on certain type of data. Therefore, knowledge is centred on that specific person which may become a bottleneck in case of absence of that individual person. Consequently, this dependency on expert knowledge slows the team progress down and limits playing the Knowledge Enabler. While in some cases the law requires that only a person with a specific educational background has access to certain data, in other cases access to data could actually be shared openly with other team members. Therefore, Scrum teams and respective managers have to talk openly about rights and duties, and whether or not expectations can be met in reality while taking the organisational context into account.

7.3.2 Implications for the Scrum Master

The Scrum Master has to be granted sufficient legitimacy to shelter the team while it matures and to preserve the leadership gap as a major enabler for the team's transformation. Especially in traditional development companies that are rather hierarchical in its nature, developers tend to struggle in receiving the leadership gap. Manifold, the Product Owner or managers would jump into the leadership gap instead of allowing the team to do so. Therefore, Developers need a Scrum Master to protect the team and shelter it while it matures.

The Scrum Master must encourage the team to take the time to regularly reflect upon the leadership roles during the Retrospective, learn their meaning and content, build a mutual understanding and figure out how and to what extent to take on leadership roles. Simultaneously, the Scrum Master must be patient and wait until team members take on responsibility when they face a lack of leadership.

While immature teams rather focus on internal issues, more mature teams have learned self-regulating behavior and are ready to focus on adapting to changes in external demands [HW05]. The Scrum Master steps back and may be needed less over time, and hence, can start focusing on other issues. Either a Scrum Master could become responsible for other Development Teams since the respective team requires less time or the Scrum Master could become an organisational coach and focus even more on solving organisational impediments and aligning divers agile teams. For example, the Scrum Master could increase coaching service for management and Product Owner. Also the Scrum Master could take even more care of linking the Developers with relevant stakeholders.

Furthermore, I have uncovered that some developers demand their Scrum Master to also do project management activities. Thus, some Scrum Masters operating in rather traditional development organisations may need to embody additional thoroughly selected project management activities. Hence, implementing the agile way of working means to continuously walk the tightrope between keeping purposeful bureaucratic facets and changing towards more agile characteristics.

My findings have several implications for the training of Scrum Masters. Many consultancy firms offer a certified 2-day Scrum Master training in which they explain the basics of the Scrum method. I claim that an effective Scrum Master needs a more comprehensive learning opportunity than a 2-day training.

My results demonstrate that the Scrum Master is a very comprehensive role, dealing with complex team mechanisms and organisational features. I propose that a long-term training for the Scrum Master is needed since a longer period of time is required to understand the impact of the nine leadership roles on teams and organisations. The Scrum Master has to learn how to apply the role more context-dependent instead of following a given process.

The Scrum Master needs regular knowledge exchange meetings to continuously develop the skill to train teams in the agile manner. For example, a community of Scrum Masters can meet regularly for peer consulting to learn from each other and support each other. Team mechanisms are highly complex and many situations in daily life of a Scrum Master are unique. It is therefore important for a Scrum Master to continuously discuss with peers about practical examples and to do peer coaching.

7.3.3 Implications for the Product Owner

Many teams cannot adapt the agile manner due to hindrances on organisational level. Those challenges on organisational level can seldom be solved by team members or the Scrum Master alone because of lack of legitimate power. In such organisational environment, it is the task of the Product Owner and management to provide a framework within which developers receive the necessary freedom to act.

The Product Owner is an important mediator between the organisational surrounding and the Development Team. The Product Owner is supposed to point at the overall direction, as agreed to with the customer, while also protecting the team from organisational pressure. Yet, some Product Owners were found to be torn in between delivery pressure and providing freedom to the Developers. Some Scrum Masters were limited in their scope of action due to the Product Owner putting pressure on them. Therefore, the agile teams seldom performed the leadership roles. I found that in rather traditional development organisations Product Owners struggled in bridging the expectations by the organisation with the expectations and needs of the Scrum Master and the Developers.

I therefore suggest to take action to diminish the organisational pressure. For example, representatives from the Development Teams could meet regularly and increase alignment between different agile teams in terms of budget and resources. Also, the Product Owner should believe even more in the power of providing freedom to the Developers instead of asking for regular status reports.

7.3.4 Implications for the Management

Many practitioners on the management level have set the agile transformation of their organisations as one of their top priorities. One method to reach this goal is the framework Scrum which promises that when implementing agile projects their teams are instantly “doing twice the work in half the time” [SS14]. Yet, the Scrum Guide describes an ideality missing a link to team mechanisms such as maturity and a detailed description of how to comfort project teams on their journey to become an agile team. Thus, few have recognized and welcomed the time required for the team development process.

Even though management expects employees to change and take on more responsibility, some managers are reluctant to grant leadership roles to the teams. Due to the bureaucratic past being deeply imprinted in parts of the organisational memory, traditional viewpoints on hierarchy tend to remain in some areas. Therefore it is easy to hold on to traditional sources of power. External pressure, top-down changed targets, shifted priorities and rewards for traditional behavior as well as frequent changes of the team setup destroy the sheltered space within which agile teams can grow.

When implementing agile teams in rather traditional development companies, often organisations develop hybrid models: they try new approaches to collaboration while keeping to traditional processes and sources of power, e.g. in terms of reporting. This leads to even more complexity and higher efforts for alignment and communication. For example, agile teams may have a review and a status report separately. This does not only take more time but may also result in controversial decision-making processes. Even though managers expect developers to take on leadership roles, not everyone is aware of the consequences. While some managers steadily transfer more and more responsibility to the team, others still cling to traditional decision-making rituals. This might lead to de-motivation of agile teams since they were promised more leadership responsibility when launching agile methods.

I suggest that managers try even harder to change their mind-set towards more agility and provide the appropriate boundary condition for the agile way of working. Despite much effort to provide autonomy to teams, many agile development projects still need more rights and less processes to fulfill their tasks successfully. I therefore suggest to critically examine the existing structure and processes and agree upon decision-making responsibilities. Moreover, I propose to avoid introducing even more formal meetings when implementing Scrum teams but instead trust agile teams to take on leadership roles in a more informal way. This will lead to improved team maturity, and therefore to high performing teams. Management not only needs to provide the leadership gap but also to provide help to fill the gap. For example, grant access to resources such as required level of rights for decision-making.

Additionally, managers should provide opportunities for developers to be actively part of the needed structural change towards a more agile organisation. For example, managers should provide the necessary freedom to the team to design their own working conditions. Furthermore, I urge managers to provide even more transparency and opportunities to voice the opinion by a regular Backlog Grooming and implementing openly accessible activity boards. Including developers in strategic decisions will lead to even more motivation and commitment to take on leadership roles.

7.3.5 Implications for the Organisation

Besides high power distance, I found that specialist culture and a functional departmentalised structure decreased team leadership. Therefore, established companies that have already started its agile transformation by implementing agile teams have to increase its efforts even further to empower teams to take on leadership. They have to put even more effort into changing the organisational structure and culture.

Even though the Scrum method suggests dedicated full-time team members that own all competences needed to fulfill a given task, this is often not possible in reality since companies fulfill different customer's needs simultaneously.

Consequently, teams have to call in experts according to their sprint goals which is often difficult in rather bureaucratic organisations due departmentalised structure and fixed staffing on projects or departments. Therefore, the organisational structure has to change more to an **open organisation** with easy access to different competences and skills. Companies can also consider more **open source projects** and approach experts from the outside. This makes staffing of projects more flexible and generates new knowledge. It also includes the advantage of external workers being independent from company career systems and norms, therefore, making them unbiased actors which allows for more entrepreneurial spirit.

Furthermore, established companies should foster a learning organisation [GEG08] that encourages employees to continuously share knowledge openly and to learn from each other unrelated to their position or functional structure. I suggest that a learning organisation and a cross-boundary friendly structure are prerequisites for team learning and cross-functional collaboration. This requires tools and communities to allow for transparency, e.g. social business platforms that are easily accessible. Management needs to foster knowledge management tools and experiments, and appreciate those that actively share or promote knowledge and voice their opinion. Furthermore, managers should provide space and work time for communities of practice and open space meetings, give developers the right to speak on conferences and grant developers a personal training budget which they can manage themselves.

The opportunity to contribute and to be acknowledged for knowledge sharing among team members and communities intrinsically motivates individuals and fosters team learning and creativity [TN86]. This leads to an open organisation with easy access to different competences and skills. Besides, knowledge sharing saves time in the long-term since it impedes re-inventing the wheel each time a new team faces a challenge another team had already solved in the past.

I have observed several promising initiatives in established companies to start an agile transformation by setting up agile teams in protected yet even isolated clusters. Providing a guarded environment to let teams, management and the surrounding structures try, experiment, learn and accept new ways of collaboration for too long brings the risk that those "islands" will only exist as such. Established companies that have made first experience with working in an agile way and that have created valuable insights must take the next challenging step to introduce their individual learning episodes to a broader organisational level and to expand their activities to those structures that seemed not ready yet, by providing similar values, believes as well as goal setting and to nourish from the success of their protected test teams.

7.4 Limitations

We have already referred to limitations in part I and part II. In the following further limitations of the thesis are described.

The Grounded Theory is embedded in the context of the subjects under study and the semi-structured interviews are based on personal interpretation and retrospective memories of interviewees. To increase objectivity of the qualitative interviews and observations (part I), I conducted a quantitative survey (part II). Yet, the disadvantage of the quantitative exploration was that I could not capture the context within which data was collected. For example, I did not collect data on culture, structure or additional formal roles since this would have added numerous more items to the survey which probably had increased the drop-out rate of respondents. Due to a lack of information on the context my explanation of the findings of the quantitative exploration is rather of speculative nature. I therefore would like to emphasize that the quantitative data in part II should be interpreted while considering the context as described in part I. For future studies I recommend to do more mixed-studies of qualitative and quantitative methods. For example, a study with another focus topic on leadership in agile teams could conduct a longitudinal case study based on observation, qualitative interviews and several iterations of the quantitative survey.

Even though part I and part II were based on two different samples, data was collected from one conglomerate only. While working on this thesis, I conducted a research cooperation with another large company active in the automotive industry. Due to legal requirements I do not report on the other company in this thesis and just provide a brief insight in the following. For detailed information please refer to the respective paper by Söderqvist and Spiegler [SSne].

The cooperation was a shared project on the leadership enabler [SSne]. Even though the Developers at the other company did not apply Scrum, product development teams had implemented a role similar to the Scrum Master that was responsible to help the team to self-organise itself while developing products. This role rather asked critical questions and helped the team to reflect upon teamwork than telling the team what to do. I therefore believe, that my findings on the Scrum Master as a leadership enabler can be broadened to other company contexts. Moreover, it supports the proposal of the 'Agile Master' which is unrelated to a specific method like Scrum. Since the other company was a Swedish corporation, it broadens the research implications to another culture. Nevertheless, the group is also operating in the automotive domain which limits our findings to a specific industry.

Even though I collected data from teams operating in different industries, this thesis neglects which kind of product was developed in the distinct projects. Distinct products require distinct project methods [Rup10]. Therefore, different business cases may also require different kind of leadership sharing. Even though we scratch on the market conditions and product requirements in our Spiegler, Heinecke, and Wagner [SHW19b] paper we have not considered in detail in which way the business model and product influences the way leadership is shared. Future studies should take the market environment, business model and product into account while examining leadership sharing.

While many companies report on success stories on agile teams this research also takes the challenges into consideration. I believe that suitable leadership in agile teams requires effort and time on many different levels. This study is one further step to help management understand how culture and structure limit and enhance leadership in agile teams. More research is needed to understand supporting and hindering factors and to how they apply in reality [GBS+16].

7.5 Future Research Directions

The research contribution and limitations pave the way for a number of follow-up research options. In the following, I describe the possible avenues for research that build on this thesis.

Future research could create a valid assessment of the 9-Factor Theory. A psychometric evaluation of the 9-Factor theory can design a valid test of the 9 leadership roles in agile teams. The resulting questionnaire can be used to test distinct hypotheses. For example, a quantitative test can examine the relationship between agile leadership and performance. It can examine if teams that share the nine leadership roles to a higher extend tend to build more effective products. Likewise, such a valid test can show if a particular distribution of the roles among team members leads to more effective product development. For example, it could evaluate whether or not teams are more effective if product related factors are more centered on developers and psychological factors are more centred on one dedicated Scrum Master. Knowing this could help practitioners provide appropriate training for Scrum Masters and developers, and it would also help them to hire the right fit for a role. If teams with a Scrum Master who takes on product-related factors tend to develop more effectively, companies should hire Scrum Masters with a technical background. If teams develop more effectively with a Scrum Master who rates high in psychological factors, companies should rather hire Scrum Masters with a background in psychology.

Moreover, a longitudinal study could assess if product development becomes more effective if the Scrum teams are trained in the 9 leadership roles and in the role transfer process. The research method could be either an experiment or action research. The researchers could train some groups in the nine leadership roles, while they would not train control groups. Comparing the trained groups with the control groups could provide valuable insights on the effectiveness of the nine leadership roles on product development. Moreover, it would be interesting to examine if the role transfer process in the trained groups increases and if leadership roles are shared to a larger extend in these groups than in the control groups.

Furthermore, research could compare the presence and distribution of the nine roles in teams operating in different company types. For example, research could compare organisations with flat hierarchical structure and strong hierarchical structure or expert culture and open learning culture. The results would show if some roles are more present in some company contexts than in others. Assuming that a team with a strong presence of the nine leadership roles tends to work in a more agile way than teams with a weak presence of the nine roles, the results would provide insights into which company context foster teams working in a more agile way. The findings would support change agents to consider necessary changes in the organisational context to allow teams to work in a more agile way. Moreover, the findings would help practitioners decide how to train Scrum Masters and developers in different company contexts.

Last but not least, we have found that management and Product Owner need to provide freedom but also have to take the individual's specific need of freedom into account. Some teams may need more freedom than others, just as some individuals need more freedom than others. Hoda and Murugesan [HM16] suggest a relationship between Hofstede's cultural model [MH11] and individual's willingness to act autonomously. I found that individuals perceive autonomy and control differently. Future research could test the relationship between the individual's capability of self-leadership [MS87] and the willingness to take on leadership roles. I suggest that depending on the level of self-leadership, team members share leadership roles differently. If individuals with a low level of self-leadership tend to take on the leadership roles less often, practitioners could train individuals in self-leadership in order to empower them to take on leadership roles in the future.

Chapter 8
Conclusion

An increasing number of companies aim to develop software in an agile way. The agile way of working promises high-value, feedback-oriented products in a fast-pacing, complex and uncertain environment [SS20; WC03]. While some companies narrate success stories, others still struggle. How to establish agile teams successfully and how to accompany them on their agile journey is still left unexplained. Even though agile methods increase in popularity, the agile way of working still lacks in theoretical underpinning [DD08; DNBM12]. Cockburn [CH01] suggests that a team learns over time how to work in an agile way. One way to support developers evolving into a truly agile team is fitting leadership. What kind of leadership a team needs that is by definition self-organised is not yet clear [SMD11].

One leadership role that helps the Developers to work in an agile way is the Scrum Master. A few studies suggest a changing leadership role while the team evolves into a truly agile self-organised team. This thesis replies to the calls for more research on leadership in agile teams [SMD11] by explaining evolving leadership and by providing a theoretical foundation to leadership in agile teams. This thesis is the first research project to thoroughly examine leadership of evolving agile teams using the example of a Scrum Master. I contribute to understanding leadership roles in agile teams by integrating research on team leadership from self-managed teams into research on agile teams. While research often relies on single case studies, I approached more than 48 agile teams in a real company setting and found support for the claim of a changing leadership role.

I first conducted a qualitative study applying Grounded Theory. The data derived from observations and interviews with 53 Scrum practitioners from 29 teams. I continuously compared emerging data with existing theory from agile software development but also from other research fields that focus on human behavior. I built a new theory by integrating research on team leadership in self-managed teams into research on leadership in agile teams. The result is a theoretical explanation for the changing leadership role. I later on built on the findings and conducted a quantitative survey on the 9 leadership roles with 67 individuals from more than 19 different Scrum teams.

I discovered that the Scrum Master embodies 9 distinct leadership roles that empower the team to work in an agile way. While the team matures those roles are transferred to the Developers via the role transfer process. The leadership gap is the core enabler. It describes a hierarchical- and leader-free space which empowers developers to take on leadership roles themselves. Yet, I did not find that the dedicated leader becomes obsolete in a mature team, but that the committed Scrum Master plays the role to a lesser extend.

Moreover, I found that leadership in agile teams is highly context-dependent and changes according to the internal team environment, contextual factors including leadership culture, organisational culture and functional, departmentalised structure. I suggest the Agile Matching Theory which implies a mandatory fit between contextual factors and the internal team environment for the role transfer to occur. I suggest that a contradiction between expectations deriving from a rather bureaucratic organisation and from the agile way of working is one of the reasons why truly agile teams are rare in bureaucratic company settings.

The findings aim to support companies in their agile transformation. I believe that by focusing more on understanding the underlying human behavior and context of agile methods rather than understanding how to conduct an agile method according to the book, will boost the agile way of working.

Leadership in an agile setting is highly context-dependent. Each company has to find its own leadership model while taking the current situation into account. Finding appropriate leadership that supports agile teams during the agile transformation requires regular reflection upon the current maturity, and expectations deriving from culture and structure. To think that there is an agile ’final’ stage and a road-map to follow is a non-agile thought in itself. Leadership is about continuously finding an equilibrium within a given context - taking culture, structure and maturity into account.

Future studies should build on my theoretical foundation and empirical findings to conduct more research on leadership in agile software development teams.

Bibliography

[AHK08] S. Adolph, W. Hall, P. Kruchten. “A methodological leg to stand on: lessons learned using grounded theory to study software development.” In: *Proceedings of the 2008 conference of the center for advanced studies on collaborative research: meeting of minds*. 2008, pp. 166–178 (cit. on p. 85).

[Bäc19] G. Bäcklander. “Doing complexity leadership theory: How agile coaches at Spotify practise enabling leadership.” In: *Creativity and Innovation Management* 28.1 (2019), pp. 42–60 (cit. on pp. 22, 23, 42, 44, 48, 49, 51, 54, 55, 66, 68, 72, 118, 120, 124, 125, 138, 153).

[Bar93] J. R. Barker. “Tightening the iron cage: Concertive control in self-managing teams.” In: *Administrative science quarterly* (1993), pp. 408–437 (cit. on pp. 22, 59–61, 68, 70–72, 147).

[BBCL16] E. Bernstein, J. Bunch, N. Canner, M. Lee. “Beyond the holacracy hype.” In: *Harvard business review* 94.7 (2016), p. 8 (cit. on p. 65).

[BC07] A. Bryant, K. Charmaz. *Grounded theory*. London, United Kingdom: SAGE Publications Ltd, 2007 (cit. on pp. 82, 83).

[Bec00] K. Beck. *Extreme programming explained: Embrace change*. Boston, MA: Addison-Wesley, 2000 (cit. on pp. 39, 66).

[BLPS01] R. Baskerville, L. Levine, J. Pries-Heje, S. Slaughter. “How internet software companies negotiate quality.” In: *Computer* 34.5 (2001), pp. 51–57 (cit. on pp. 37, 38).

[BO88] N. D. Birrell, M. A. Ould. *A practical handbook for software development*. Cambridge: Cambridge University Press, 1988 (cit. on p. 36).

[Boe06] B. Boehm. “A view of 20th and 21st century software engineering.” In: *Proceedings of the 28th international conference on Software engineering*. 2006, pp. 12–29 (cit. on pp. 33–37).

[Boe88] B. W. Boehm. “A spiral model of software development and enhancement.” In: *Computer* 21.5 (1988), pp. 61–72 (cit. on pp. 34–36).

[Bon10] N. A. Bonner. “Predicting leadership success in agile environments: An inquiring systems approach.” In: *Journal of Management Information and Decision Sciences* 13.2 (2010), pp. 83–103 (cit. on pp. 43, 44, 50, 65).

[BP19] H. Barke, L. Prechelt. “Role clarity deficiencies can wreck agile teams.” In: *PeerJ Computer Science* 5 (2019), e241 (cit. on pp. 62, 119, 148).

[BT05] B. Boehm, R. Turner. “Management challenges to implementing agile processes in traditional development organizations.” In: *IEEE Software* 22.5 (2005), pp. 30–39 (cit. on pp. 23, 43, 76, 152).

[BTPH17] M. Beaumont, B. Thuriaux-Alemán, P. Prasad, C. Hatton. “Using agile approaches for breakthrough product innovation.” In: *Strategy & Leadership* (2017), pp. 26–37 (cit. on p. 50).

[CB97] S. G. Cohen, D. E. Bailey. “What makes teams work: Group effectiveness research from the shop floor to the executive suite.” In: *Journal of management* 23.3 (1997), pp. 239–290 (cit. on p. 70).

[CCWP11] K. Conboy, S. Coyle, X. Wang, M. Pikkarainen. “People over process: key people challenges in agile development.” In: *IEEE Software* 28.4 (2011), pp. 47–57 (cit. on p. 76).

[CH01] A. Cockburn, J. Highsmith. “Agile software development: The people factor.” In: *Computer* 11 (2001), pp. 131–133 (cit. on pp. 22, 23, 38, 43–46, 48, 53, 65, 66, 69, 77, 78, 87, 149, 153, 167).

[Cha16] K. Charmaz. “Shifting the Grounds: Constructivist Grounded Theory Methods for the 21st Century.” In: *Developing grounded theory: The second generation*. Ed. by J. M. Morse, P. N. Stern, J. Corbin, B. Bowers, K. Charmaz, A. E. Clarke. Vol. 3. Routledge, 2016, pp. 127–193 (cit. on pp. 83, 88).

[Cha91] H. P. S. Charles C. Manz. “Super Leadership: Beyond the myth of heroic leadership.” In: *Organizational Dynamics* 19.4 (1991), pp. 18–35 (cit. on pp. 48, 59, 71, 150).

[CLP10] L. Crevani, M. Lindgren, J. Packendorff. “Leadership, not leaders: On the study of leadership as practices and interactions.” In: *Scandinavian journal of management* 26.1 (2010), pp. 77–86 (cit. on pp. 47, 49, 55, 67, 129).

[Col04] G. A. Cole. *Management theory and practice*. 6th ed. London: Cengage Learning EMEA, 2004 (cit. on p. 48).

[Con09] K. Conboy. “Agility from first principles: Reconstructing the concept of agility in information systems development.” In: *Information systems research* 20.3 (2009), pp. 329–354 (cit. on pp. 21, 43, 64).

[Cor20] P. Corry. *The Evolution of the Scrum Guide— ‘10 to ‘19*. 2010 (accessed June 10, 2020). URL: `https://medium.com/serious-scrum/the-evolution-of-the-scrum-guide-10-to-19-f3ac4d82cfcb` (cit. on p. 40).

[CS15] J. Corbin, A. Strauss. *Basics of qualitative research: Techniques and procedures for developing grounded theory*. 4. ed. Los Angeles, Calif.: Sage publications, 2015 (cit. on p. 82).

[CT09] F. K. Chan, J. Y. Thong. “Acceptance of agile methodologies: A critical review and conceptual framework.” In: *Decision support systems* 46.4 (2009), pp. 803–814 (cit. on p. 153).

[DD08] T. Dybå, T. Dingsøyr. “Empirical studies of agile software development: A systematic review.” In: *Information and software technology* 50.9-10 (2008), pp. 833–859 (cit. on p. 167).

[Den15] S. Denning. “Agile: it’s time to put it to use to manage business complexity.” In: *Strategy & Leadership* 43.5 (2015), pp. 10–17 (cit. on p. 53).

[DNBM12] T. Dingsøyr, S. Nerur, V. Balijepally, N. B. Moe. “A decade of agile methodologies: Towards explaining agile software development.” In: *Journal of Systems and Software* 85.6 (2012), pp. 1213–1221 (cit. on p. 167).

[DW03] V. U. Druskat, J. V. Wheeler. “Managing from the boundary: The effective leadership of self-managing work teams.” In: *Academy of Management Journal* 46.4 (2003), pp. 435–457 (cit. on pp. 45, 59–61, 65, 71, 72).

[Edm99] A. Edmondson. “Psychological safety and learning behavior in work teams.” In: *Administrative science quarterly* 44.2 (1999), pp. 350–383 (cit. on pp. 52, 64, 71, 75, 76, 79, 94, 105, 106).

[Fie67] F. E. Fiedler. *A theory of leadership effectiveness*. New York: McGraw-Hill, 1967 (cit. on p. 52).

[Fis19] M. Fischer. “Exploring External Leadership in Agile Software Development Teams and its Influence on Team Empowerment.” In: International Research Workshop on IT Project Management 2019. 2019 (cit. on p. 50).

[FM91] K. Forsberg, H. Mooz. “The relationship of system engineering to the project cycle.” In: *INCOSE International Symposium*. Vol. 1. 1. Wiley Online Library. 1991, pp. 57–65 (cit. on p. 36).

[GBS+16] P. Gregory, L. Barroca, H. Sharp, A. Deshpande, K. Taylor. “The challenges that challenge: Engaging with agile practitioners’ concerns.” In: *Information and Software Technology* 77 (2016), pp. 92–104 (cit. on pp. 21, 55, 66, 76, 164).

[GCD+16] E. Gonzalez-Mulé, S. H. Courtright, D. DeGeest, J.-Y. Seong, D.-S. Hong. “Channeled autonomy: The joint effects of autonomy and feedback on team performance through organizational goal clarity.” In: *Journal of Management* 42.7 (2016), pp. 2018–2033 (cit. on p. 65).

[GEG08] D. A. Garvin, A. C. Edmondson, F. Gino. “Is yours a learning organization?” In: *Harvard business review* 86.3 (2008), pp. 1–10 (cit. on pp. 65, 78, 153, 161).

[Ger89] C. J. Gersick. “Marking time: Predictable transitions in task groups.” In: *Academy of Management journal* 32.2 (1989), pp. 274–309 (cit. on p. 63).

[GGJ19] L. Gren, A. Goldman, C. Jacobsson. “Agile ways of working: A team maturity perspective.” In: *Journal of Software: Evolution and Process* (2019), e2244 (cit. on pp. 21, 63, 64, 74, 118, 124, 125, 151).

[Gin19] R. C. Ginnett. “Crews as groups: Their formation and their leadership.” In: *Crew resource management*. Elsevier, 2019, pp. 73–102 (cit. on p. 63).

[GL20] L. Gren, M. Lindman. “What an Agile Leader Does: The Group Dynamics Perspective.” In: *International Conference on Agile Software Development*. Springer, 2020, pp. 178–194 (cit. on pp. 23, 50–52, 54, 55, 76, 150–152).

[Gra91] J. W. Graham. “Servant-leadership in organizations: Inspirational and moral.” In: *The Leadership Quarterly* 2.2 (1991), pp. 105–119 (cit. on p. 52).

[Gri10a] A. Grigg. “Employee empowerment is the main ingredient in a baking company’s competitive strategy.” In: *Global Business and Organizational Excellence* 29.2 (2010), pp. 6–18 (cit. on p. 151).

[Gri10b] K. Grint. *Leadership: A very short introduction*. Oxford University Press, 2010 (cit. on pp. 47, 49).

[GS17] B. G. Glaser, A. L. Strauss. *Discovery of grounded theory: Strategies for qualitative research*. New York: Routledge, 2017 (cit. on pp. 26, 82, 83, 86, 88, 89, 120, 124).

[GTF17] L. Gren, R. Torkar, R. Feldt. “Group development and group maturity when building agile teams: A qualitative and quantitative investigation at eight large companies.” In: *Journal of Systems and Software* 124 (2017), pp. 104–119 (cit. on pp. 23, 45, 46, 49, 51, 52, 54, 55, 63, 66, 73, 124, 143).

[GYCS93] R. A. Guzzo, P. R. Yost, R. J. Campbell, G. P. Shea. “Potency in groups: Articulating a construct.” In: *British journal of social psychology* 32.1 (1993), pp. 87–106 (cit. on pp. 75, 76, 79, 106).

[HB13] D. Hodgson, L. Briand. “Controlling the uncontrollable:‘Agile’teams and illusions of autonomy in creative work.” In: *Work, employment and society* 27.2 (2013), pp. 308–325 (cit. on pp. 65, 76, 77, 150–152).

[HBJ07] P. Hersey, K. H. Blanchard, D. E. Johnson. *Management of organizational behavior*. Upper Saddle River: Prentice Hall, 2007 (cit. on p. 52).

[HC04] H. Heath, S. Cowley. “Developing a grounded theory approach: a comparison of Glaser and Strauss.” In: *International journal of nursing studies* 41.2 (2004), pp. 141–150 (cit. on p. 82).

[HH02a] J. R. Hackman, R. J. Hackman. *Leading teams: Setting the stage for great performances*. Harvard Business Press, 2002 (cit. on pp. 45, 59, 61, 65, 152).

[HH02b] J. A. Highsmith, J. Highsmith. *Agile software development ecosystems*. Boston: Addison-Wesley, 2002 (cit. on pp. 39, 66).

[Hig09] J. R. Highsmith. *Agile project management: creating innovative products*. Boston: Pearson Education, 2009 (cit. on pp. 65, 69).

[Hig20] J. Highsmith. *History: The Agile Manifesto*. 2001 (accessed June 10, 2020). URL: https://agilemanifesto.org/history.html (cit. on p. 39).

[HK18] N. Holtzhausen, J. J. de Klerk. “Servant leadership and the Scrum team’s effectiveness.” In: *Leadership & Organization Development Journal* 39.7 (2018), pp. 873–882 (cit. on pp. 50, 52, 54).

[HM16] R. Hoda, L. K. Murugesan. “Multi-level agile project management challenges: A self-organizing team perspective.” In: *Journal of Systems and Software* 117 (2016), pp. 245–257 (cit. on pp. 21, 45, 53, 55, 71, 72, 76, 77, 79, 150, 165).

[HN17] R. Hoda, J. Noble. “Becoming agile: a grounded theory of agile transitions in practice.” In: *Proceedings of the 39th International Conference on Software Engineering*. IEEE Press. 2017, pp. 141–151 (cit. on pp. 21, 23, 38, 45, 46, 53, 66, 69, 77, 114, 150–152).

[HNM12a] R. Hoda, J. Noble, S. Marshall. “Developing a grounded theory to explain the practices of self-organizing Agile teams.” In: *Empirical Software Engineering* 17.6 (2012), pp. 609–639 (cit. on pp. 44, 50, 64, 65, 68, 69, 71, 83, 86).

[HNM12b] R. Hoda, J. Noble, S. Marshall. “Self-organizing roles on agile software development teams.” In: *IEEE Transactions on Software Engineering* 39.3 (2012), pp. 422–444 (cit. on pp. 22, 62, 65, 68, 90, 117, 118, 148).

[Hod11] R. Hoda. “Self-organizing agile teams: A grounded theory.” PhD thesis. Victoria University of Wellington, 2011 (cit. on p. 96).

[Hod13] R. Hoda. “Power to the People.” In: *IEEE software* 30.2 (2013), pp. 92–92 (cit. on pp. 22, 23, 66, 120, 149, 152, 153).

[HP06] M. Hoegl, P. Parboteeah. “Autonomy and teamwork in innovative projects.” In: *Human Resource Management* 45.1 (2006), pp. 67–79 (cit. on pp. 44, 45).

[HS14] M. Hammarberg, J. Sunden. *Kanban in action*. Shelter Island, New York: Manning Publications Co., 2014 (cit. on p. 39).

[HW05] J. R. Hackman, R. Wageman. “A theory of team coaching.” In: *Academy of management review* 30.2 (2005), pp. 269–287 (cit. on pp. 61, 77, 157).

[ICLG02] Z. Irani, J. Choudrie, P. E. Love, A. Gunasekaran. “Sustaining TQM through self-directed work teams.” In: *International Journal of Quality & Reliability Management* 19.5 (2002), pp. 596–609 (cit. on p. 65).

[Kak17] A. K. Kakar. “Investigating the prevalence and performance correlates of vertical versus shared leadership in emergent software development teams.” In: *Information Systems Management* 34.2 (2017), pp. 172–184 (cit. on pp. 50, 73).

[KGM+96] S. W. Kozlowski, S. Gully, P. McHugh, E. Salas, J. Cannon-Bowers. “A dynamic theory of leadership and team effectiveness: Developmental and task contingent leader roles.” In: *Research in personnel and human resources management* 14 (1996), pp. 253–306 (cit. on pp. 61, 69).

[KMC16] S. W. Kozlowski, S. Mak, G. T. Chao. “Team-centric leadership: An integrative review.” In: *Annual Review of Organizational Psychology and Organizational Behavior* 3 (2016), pp. 21–54 (cit. on p. 49).

[KS15] J. R. Katzenbach, D. K. Smith. *The wisdom of teams: Creating the high-performance organization*. Boston: Harvard Business Review Press, 2015 (cit. on p. 63).

[KW88] B. Keys, J. Wolfe. “Management education and development: Current issues and emerging trends.” In: *Journal of Management* 14.2 (1988), pp. 205–229 (cit. on p. 48).

[LBT95] D. H. Lindsley, D. J. Brass, J. B. Thomas. “Efficacy-performing spirals: A multilevel perspective.” In: *Academy of management review* 20.3 (1995), pp. 645–678 (cit. on p. 77).

[LLDL15] M.-L. Liu, N.-T. Liu, C. G. Ding, C.-P. Lin. “Exploring team performance in high-tech industries: Future trends of building up teamwork.” In: *Technological Forecasting and Social Change* 91 (2015), pp. 295–310 (cit. on pp. 50, 73).

[LSA11] M. Laanti, O. Salo, P. Abrahamsson. “Agile methods rapidly replacing traditional methods at Nokia: A survey of opinions on agile transformation.” In: *Information and Software Technology* 53.3 (2011), pp. 276–290 (cit. on pp. 38, 39).

[LWRJ05] D. J. Leach, T. D. Wall, S. G. Rogelberg, P. R. Jackson. “Team autonomy, performance, and member job strain: Uncovering the teamwork KSA link.” In: *Applied Psychology* 54.1 (2005), pp. 1–24 (cit. on pp. 44, 45).

[LWW01] L. L. Levesque, J. M. Wilson, D. R. Wholey. “Cognitive divergence and shared mental models in software development project teams.” In: *Journal of Organizational Behavior* 22.2 (2001), pp. 135–144 (cit. on pp. 64, 69, 75, 79, 105).

[Mad07] S. Madsen. “Conceptualising the causes and consequences of uncertainty in IS development organisations and projects.” In: *European Conference on Information Systems*. 2007 (cit. on p. 38).

[MAD12] N. B. Moe, A. Aurum, T. Dybå. “Challenges of shared decision-making: A multiple case study of agile software development.” In: *Information and Software Technology* 54.8 (2012), pp. 853–865 (cit. on pp. 21, 23, 38, 44, 53, 55, 66, 68, 72, 76, 77, 79, 150).

[May10] P. Mayring. “Qualitative inhaltsanalyse.” In: *Handbuch qualitative Forschung in der Psychologie*. Springer, 2010, pp. 601–613 (cit. on p. 25).

[MCDE15] N. B. Moe, D. S. Cruzes, T. Dybå, E. Engebretsen. “Coaching a global agile virtual team.” In: *10th International Conference on Global Software Engineering*. IEEE. 2015, pp. 33–37 (cit. on pp. 44, 68).

[MDD09] N. B. Moe, T. Dingsøyr, T. Dybå. “Overcoming barriers to self-management in software teams.” In: *IEEE Software* 26.6 (2009), pp. 20–26 (cit. on pp. 21, 59, 76, 79, 150, 152).

[MDD10] N. B. Moe, T. Dingsøyr, T. Dybå. “A teamwork model for understanding an agile team: A case study of a Scrum project.” In: *Information and Software Technology* 52.5 (2010), pp. 480–491 (cit. on pp. 22, 23, 42, 44, 51, 53–55, 58, 59, 65, 71–76, 79, 87, 96, 105, 106, 118, 120, 124, 125, 143, 152).

[MDK09] N. B. Moe, T. Dingsyr, O. Kvangardsnes. “Understanding shared leadership in agile development: A case study.” In: *42nd Hawaii International Conference on System Sciences*. IEEE. 2009, pp. 1–10 (cit. on pp. 21, 48, 51, 53, 54, 66–68, 70, 72).

[MDK10] F. P. Morgeson, D. S. DeRue, E. P. Karam. “Leadership in teams: A functional approach to understanding leadership structures and processes.” In: *Journal of management* 36.1 (2010), pp. 5–39 (cit. on p. 48).

[MH11] M. Minkov, G. Hofstede. “The evolution of Hofstede’s doctrine.” In: *Cross cultural management: An international journal* 18.1 (2011), pp. 10–20 (cit. on p. 165).

[MMRG08] J. Mathieu, M. T. Maynard, T. Rapp, L. Gilson. “Team effectiveness 1997-2007: A review of recent advancements and a glimpse into the future.” In: *Journal of management* 34.3 (2008), pp. 410–476 (cit. on p. 52).

[MMZ01] M. A. Marks, J. E. Mathieu, S. J. Zaccaro. “A temporally based framework and taxonomy of team processes.” In: *Academy of management review* 26.3 (2001), pp. 356–376 (cit. on pp. 63, 119, 139).

[MP18] N. C. Magpili, P. Pazos. “Self-managing team performance: A systematic review of multilevel input factors.” In: *Small Group Research* 49.1 (2018), pp. 3–33 (cit. on pp. 44, 45, 55, 65, 69).

[MS87] C. C. Manz, H. P. Sims Jr. “Leading workers to lead themselves: The external leadership of self-managing work teams.” In: *Administrative science quarterly* (1987), pp. 106–129 (cit. on pp. 22, 23, 53, 58–61, 68–70, 72, 165).

[Mye19] M. D. Myers. *Qualitative research in business and management*. 3rd ed. Los Angeles: Sage Publications Limited, 2019 (cit. on pp. 25, 82).

[NLB99] C. E. Nicholls, H. W. Lane, M. B. Brechu. “Taking self-managed teams to Mexico.” In: *Academy of Management Perspectives* 13.3 (1999), pp. 15–25 (cit. on p. 65).

[NMM05] S. Nerur, R. K. Mahapatra, G. Mangalaraj. “Challenges of migrating to agile methodologies.” In: *Communications of the ACM* 48.5 (2005), pp. 72–78 (cit. on pp. 21, 23, 36–38, 43, 48, 55, 76, 77, 79, 138).

[NRBB17] J. Noll, M. A. Razzak, J. M. Bass, S. Beecham. “A study of the Scrum Master’s role.” In: *International Conference on Product-Focused Software Process Improvement*. Springer. 2017, pp. 307–323 (cit. on pp. 54, 78, 119).

[NW99] G. A. Neuman, J. Wright. “Team effectiveness: beyond skills and cognitive ability.” In: *Journal of Applied psychology* 84.3 (1999), p. 376 (cit. on p. 63).

[OB16] H. H. Olsson, J. Bosch. “No More Bosses?” In: *International Conference on Product-Focused Software Process Improvement*. Springer. 2016, pp. 86–101 (cit. on p. 45).

[PF02] S. R. Palmer, M. Felsing. *A practical guide to feature-driven development*. Upper Saddle River: Prentice Hall, 2002 (cit. on p. 39).

[PGN14] R. M. Parizi, T. J. Gandomani, M. Z. Nafchi. “Hidden facilitators of agile transition: Agile coaches and agile champions.” In: *8th. Malaysian Software Engineering Conference (MySEC)*. IEEE. 2014, pp. 246–250 (cit. on pp. 68, 69, 72, 89).

[PHP15] D. W. Parker, M. Holesgrove, R. Pathak. “Improving productivity with self-organised teams and agile leadership.” In: *International Journal of Productivity and Performance Management* 64.1 (2015), pp. 112–128 (cit. on p. 150).

[Pop07] M. Poppendieck. “Lean software development.” In: *Companion to the proceedings of the 29th International Conference on Software Engineering*. IEEE Computer Society. 2007, pp. 165–166 (cit. on p. 39).

[Pri10] M. Prifling. “Exploring Leadership Styles in Software Development Projects.” In: *PACIS*. 2010, p. 72 (cit. on pp. 50, 151).

[QFTM90] R. E. Quinn, S. R. Faerman, M. P. Thompson, M. R. McGrath. *Becoming a master manager: A competency framework*. New York: Wiley, 1990 (cit. on pp. 62, 90, 117, 148).

[Qui88] R. E. Quinn. *Beyond rational management: Mastering the paradoxes and competing demands of high performance.* San Francisco: Jossey-Bass, 1988 (cit. on pp. 62, 90, 117).

[Roy87] W. W. Royce. “Managing the development of large software systems: concepts and techniques.” In: *Proceedings of the 9th international conference on Software Engineering*. 1987, pp. 328–338 (cit. on pp. 34–37).

[Rup10] N. B. Ruparelia. “Software development lifecycle models.” In: *ACM SIGSOFT Software Engineering Notes* 35.3 (2010), pp. 8–13 (cit. on pp. 33, 35, 163).

[Rus09] J. Rust. *Modern psychometrics : the science of psychological assessment*. New York: Routledge, 2009 (cit. on p. 128).

[SB02] K. Schwaber, M. Beedle. *Agile software development with Scrum*. Vol. 1. Upper Saddle River: Prentice Hall, 2002 (cit. on pp. 22, 39).

[Sch04] K. Schwaber. *Agile project management with Scrum*. Sebastopol: Microsoft press, 2004 (cit. on p. 43).

[Sch16] C. Schmidt. *Agile software development teams*. Springer, 2016 (cit. on p. 43).

[Sch97] K. Schwaber. “Scrum development process.” In: *Business object design and implementation*. Springer, 1997, pp. 117–134 (cit. on pp. 22, 38, 40).

[SGHW20a] S. V. Spiegler, D. Graziotin, C. Heinecke, S. Wagner. “A Quantitative Exploration of the 9-Factor Theory: Distribution of Leadership Roles Between Scrum Master and Agile Team.” In: *International Conference on Agile Software Development*. Springer. 2020, pp. 162–177 (cit. on p. 123).

[SGHW20b] S. V. Spiegler, D. Graziotin, C. Heinecke, S. Wagner. *A Quantitative Exploration of the 9-Factor Theory: Distribution of Leadership Roles between the Scrum Master and the Agile Team*. http://doi.org/10.5281/zenodo.3634046. 2020 (cit. on p. 130).

[SHW18] S. V. Spiegler, C. Heinecke, S. Wagner. *Interview Guidelines for "Leadership Gap in Agile Teams: How Teams and Scrum Masters Mature"*. http://doi.org/10.5281/zenodo.2243113. 2018 (cit. on p. 87).

[SHW19a] S. V. Spiegler, C. Heinecke, S. Wagner. "Leadership Gap in Agile Teams: How Teams and Scrum Masters Mature." In: *International Conference on Agile Software Development*. Springer. 2019, pp. 37–52 (cit. on p. 138).

[SHW19b] S. V. Spiegler, C. Heinecke, S. Wagner. "The influence of culture and structure on autonomous teams in established companies." In: *International Conference on Agile Software Development*. Springer. 2019, pp. 46–54 (cit. on pp. 77, 119, 163).

[SJ17] P. Srivastava, S. Jain. "A leadership framework for distributed self-organized scrum teams." In: *Team Performance Management: An International Journal* 23.5/6 (2017), pp. 293–314 (cit. on pp. 22, 23, 42, 50, 51, 53–55, 58, 67, 73, 74, 87, 124, 125, 138, 143, 144, 147).

[SLKC01] R. Schmidt, K. Lyytinen, M. Keil, P. Cule. "Identifying software project risks: An international Delphi study." In: *Journal of management information systems* 17.4 (2001), pp. 5–36 (cit. on p. 37).

[SMD11] V. G. Stray, N. B. Moe, T. Dingsøyr. "Challenges to teamwork: a multiple case study of two agile teams." In: *International conference on agile software development*. Springer. 2011, pp. 146–161 (cit. on pp. 23, 54, 56, 68, 71, 167).

[SNM01] S. Sircar, S. P. Nerur, R. Mahapatra. "Revolution or evolution? A comparison of object-oriented and structured systems development methods." In: *MIS Quarterly* (2001), pp. 457–471 (cit. on p. 38).

[Som04] I. Sommerville. *Software Engineering. Harlow, UK*. 2004 (cit. on p. 34).

[SS14] J. Sutherland, J. Sutherland. *Scrum: the art of doing twice the work in half the time*. Currency, 2014 (cit. on p. 159).

[SS20] J. Sutherland, K. Schwaber. *The Scrum guide: The definitive guide to Scrum: The rules of the game*. http://scrumguides.org/. 2020 (cit. on pp. 22, 40, 41, 167).

[SSD16] V. Stray, D. I. Sjøberg, T. Dybå. “The daily stand-up meeting: A grounded theory study.” In: *Journal of Systems and Software* 114 (2016), pp. 101–124 (cit. on pp. 78, 119, 147).

[SSne] J. B. Söderqvist, S. Spiegler. “How to enable leadership among self-organizing developers.” R&D Management Conference. Paris, 2019, June (cit. on pp. 66, 163).

[Sta14] S. Stavru. “A critical examination of recent industrial surveys on agile method usage.” In: *Journal of Systems and Software* 94 (2014), pp. 87–97 (cit. on pp. 21, 38).

[Sto74] R. M. Stogdill. *Handbook of leadership: A survey of theory and research.* Free Press, 1974 (cit. on pp. 47, 49).

[Tay16] K. J. Taylor. “Adopting Agile software development: the project manager experience.” In: *Information Technology & People* (2016) (cit. on pp. 44, 150).

[TB51] E. L. Trist, K. W. Bamforth. “Some social and psychological consequences of the longwall method of coal-getting: An examination of the psychological situation and defences of a work group in relation to the social structure and technological content of the work system.” In: *Human relations* 4.1 (1951), pp. 3–38 (cit. on pp. 22, 58).

[TH04] J. E. Tomayko, O. Hazzan. *Human aspects of software engineering*. Charles River Media, 2004 (cit. on pp. 36, 37).

[TN86] H. Takeuchi, I. Nonaka. “The new new product development game.” In: *Harvard business review* 64.1 (1986), pp. 137–146 (cit. on pp. 22, 38, 44, 50, 64–66, 68–72, 161).

[Tuc65] B. W. Tuckman. “Developmental sequence in small groups.” In: *Psychological Bulletin* 63.6 (1965), pp. 384–399 (cit. on pp. 63, 64, 67, 118, 138).

[UMM07] M. Uhl-Bien, R. Marion, B. McKelvey. “Complexity leadership theory: Shifting leadership from the industrial age to the knowledge era.” In: *The leadership quarterly* 18.4 (2007), pp. 298–318 (cit. on pp. 49, 51).

[VW09] R. Vidgen, X. Wang. “Coevolving systems and the organization of agile software development.” In: *Information Systems Research* 20.3 (2009), pp. 355–376 (cit. on pp. 43, 44, 50, 65).

[Wag01] R. Wageman. “How leaders foster self-managing team effectiveness: Design choices versus hands-on coaching.” In: *Organization Science* 12.5 (2001), pp. 559–577 (cit. on pp. 65, 68, 72).

[Wag97] R. Wageman. “Critical success factors for creating superb self-managing teams.” In: *Organizational dynamics* 26.1 (1997), pp. 49–61 (cit. on pp. 65, 69, 77, 152, 153).

[WC03] L. Williams, A. Cockburn. “Agile software development: it’s about feedback and change.” In: *IEEE computer* 36.6 (2003), pp. 39–43 (cit. on pp. 38, 167).

[WDT03] S. A. Wheelan, B. Davidson, F. Tilin. “Group development across time: Reality or illusion?” In: *Small group research* 34.2 (2003), pp. 223–245 (cit. on pp. 129, 142).

[Web09] M. Weber. *The theory of social and economic organization*. New York: The Free Press, 2009 (cit. on pp. 76, 79).

[WH96] S. A. Wheelan, J. M. Hochberger. “Validation studies of the group development questionnaire.” In: *Small group research* 27.1 (1996), pp. 143–170 (cit. on p. 129).

[WM18] K. Werder, A. Maedche. “Explaining the emergence of team agility: A complex adaptive systems perspective.” In: *Information Technology & People* 31.3 (2018), pp. 819–844 (cit. on p. 23).

[Yan96] O. Yang. “Shared leadership in self-managed teams: A competing values approach.” In: *Total Quality Management* 7.5 (1996), pp. 521–534 (cit. on pp. 22, 55, 62, 118, 148).

[Yuk13] G. Yukl. *Leadership in Organizations*. 8th ed. Boston Munich: Pearson, 2013 (cit. on pp. 47–49, 70).

[ZAM09] C. R. Zafft, S. G. Adams, G. S. Matkin. “Measuring leadership in self-managed teams using the competing values framework.” In: *Journal of Engineering Education* 98.3 (2009), pp. 273–282 (cit. on pp. 22, 55, 62, 90, 117, 132, 148).

List of Figures

List of Tables