Empirical Software Engineering (2021) 26: 41
https://doi.org/10.1007/s10664-021-09949-5

An empirical study on changing leadership in agile teams

Simone V. Spiegler[1,2] · Christoph Heinecke[3] · Stefan Wagner[1]

Accepted: 4 February 2021 / Published online: 22 March 2021

Abstract
An increasing number of companies aim to enable their development teams to work in an agile manner. When introducing agile teams, companies face several challenges. This paper explores the kind of leadership needed to support teams to work in an agile way. One theoretical agile leadership concept describes a Scrum Master who is supposed to empower the team to lead itself. Empirical findings on such a leadership role are controversial. We still have not understood how leadership unfolds in a team that is by definition self-organizing. Further exploration is needed to better understand leadership in agile teams. Our goal is to explore how leadership changes while the team matures using the example of the Scrum Master. Through a grounded theory study containing 75 practitioners from 11 divisions at the Robert Bosch GmbH we identified a set of nine leadership roles that are transferred from the Scrum Master to the Development Team while it matures. We uncovered that a leadership gap and a supportive internal team climate are enablers of the role transfer process, whereas role conflicts may diminish the role transfer. To make the Scrum Master change in a mature team, team members need to receive trust and freedom to take on a leadership role which was previously filled by the Scrum Master. We conclude with practical implications for managers, Product Owners, Development Teams and Scrum Masters which they can apply in real settings.

Keywords Agile working · Development team · Scrum master · Maturity · Leadership

1 Introduction

Recently, more and more organizations aim at developing their products in a more agile way (Stavru 2014). Agile teams are said to work cross-functionally and self-organized in iterative

Communicated by: Kelly Blincoe

✉ Simone V. Spiegler
simone.spiegler@iste.uni-stuttgart.de

1 Institute of Software Engineering, University of Stuttgart, Stuttgart, Germany

2 Robert Bosch Automotive Steering GmbH, Schwäbisch Gmünd, Germany

3 Robert Bosch GmbH, Stuttgart, Germany

learning loops towards a common goal (Conboy 2009). Even though an increasing number of organizations strive to implement agile teams, it is not entirely clear how teams can adopt the agile way of working (Moe et al. 2010; Nerur et al. 2005). Especially rather bureaucratic companies seem to struggle in their agile transformation (Moe et al. 2009; Nerur et al. 2005). Fitting leadership behavior is found to be one key success factor for evolving into an agile self-organized team (Gren et al. 2019). However, which kind of leadership agile teams need is not yet clear.

The successful implementation of self-organized 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 (Manz and Sims 1987; Srivastava and Jain 2017; Trist and Bamforth 1951), knowledge management (Takeuchi and Nonaka 1986), complexity theory (Bäcklander 2019; Schwaber 1997), role theory (Hoda et al. 2013; Yang 1996) and agile project management (Sutherland and Schwaber 2017). One recurring topic across the various streams of research is the role of leadership in a team that is by definition self-organized.

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. Takeuchi and Nonaka (1986)) and an autonomous team that self-organizes its operational work (e.g. Hoda et al. (2013)). While some researchers suggest a facilitator who serves as a peer to team members (Takeuchi and Nonaka 1986) or a leader who empowers the team to lead itself (Manz and Sims 1987), other researchers do not consider a formal leader *of* the team but instead emphasize self-organizing roles *within* the team (Hoda et al. 2013).

At present, the most widely known agile approach is Scrum (Schwaber 1997) which divides leadership between three different prescriptive roles: the Product Owner, the Scrum Master and the Development Team. 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 team to work cross-functionally and self-organized, protects the Development Team from external disruption and helps the organization to adapt to the agile way of working (Sutherland and Schwaber 2017). The Development Team self-assigns tasks towards a shared goal (Cockburn and Highsmith 2001).

It is not entirely clear how the Scrum Master and the Development Team share leadership. While some empirical studies reveal that leadership is rather focused on one dedicated Scrum Master (Moe et al. 2010), other studies suggest that the whole team takes on leadership (Srivastava and Jain 2017).

The aim of the dedicated Scrum Master is shared leadership. In this paper, we use the understanding by Pearce and Conger who "define shared leadership as a dynamic, interactive influence process among individuals in groups for which the objective is to lead one another to the achievement of group or organizational goals or both. This influence process often involves peer, or lateral, influence and at other times involves upward or downward hierarchical influence." (Pearce and Conger, 2002, p. 1) This paper focuses on the link between one dedicated person who takes on leadership and sharing leadership within the team.

We examine leadership in agile teams using the example of the Scrum Master. We consider the dedicated Scrum Master to be a leadership enabler who follows the goal to empower a team to work in an agile way and to share leadership. A leadership enabler transforms followers into leaders while the team matures. The Scrum Master does not aim at a specific quantity or quality of output (Bäcklander 2019). We will focus on leadership of the Scrum Master and the Development Team, because we want to understand how the Scrum Master and the developers share leadership over time (Moe et al. 2010).

Empirical research on Scrum teams revealed that the Scrum Master does not always split leadership openly with the team but occasionally becomes an obstacle for distribution of leadership within teams in early stages. This is because the Scrum Master tends to stick to a command-and-control mode in newly established teams (Moe et al. 2010), while the Scrum Master in a more mature team enables the Development Team to lead itself and to share the leadership role among team members (Srivastava and Jain 2017).

These findings could be explained by changes in the maturity of agile teams. Cockburn (2002) hints at the Japanese philosophy of Shu-Ha-Ri to explain that an agile team matures while they practice Scrum. The team learns how to work in an agile way over time and therefore needs more guidance by a Scrum Master in a less mature stage (Gren et al. 2017), while the Scrum Master is even suggested to be shared among developers in mature teams (Hoda et al. 2013).

Despite a few qualitative studies that specifically focus on exploring leadership in agile teams (Srivastava and Jain 2017; Bäcklander 2019), the empirical research on this topic is still underrepresented in the software development community. Besides the high popularity of the method Scrum, the current description of the Scrum method does not provide sufficient information on leadership in agile teams. While the Scrum Guide describes three prescriptive roles, this paper explores descriptive roles. In this article, a descriptive role describes a set of performed, connected activities. Descriptive roles are informal and transient, which implies that they are not linked to a specific position or status (Hoda et al. 2013).

To be able to support organisations in an agile transformation, our research objective is to explore how leadership changes while the team matures using the example of the Scrum Master. We believe investigating changing leadership of the Scrum Master will provide valuable insights into how teams can learn to work in an agile way.

Our study focuses on implementing agile teams in bureaucratic companies that are in the middle of their agile transformation. This type of organisation opposes the agile way of working (Nerur et al. 2005). It is used to rely on a hierarchy in contrast to self-organizing teams (Moe et al. 2012), to rigid planning instead of iterative learning (Boehm and Turner 2005) and to divide work according to functional departments instead of cross-functional teams (Nerur et al. 2005). In this paper, we use the terms bureaucratic company and traditional development company interchangeably.

We present findings from a study at 11 business divisions of the conglomerate Robert Bosch GmbH, primarily operating in the automotive industry.

We applied grounded theory (Charmaz 2016) and conducted three different rounds of interviews. While the first round contained 22 *unstructured* interviews with practitioners from various roles, including agile coaches and Scrum Masters, the second and third round included 53 qualitative *semi-structured* interviews with practitioners from 29 different software and non-software project teams that had applied Scrum over a period of three months up to three years. The interviews were supplemented by observations of Scrum meetings and of visiting team rooms. We analyzed derived data and received feedback by agile practitioners on our results to check whether our findings were consistent.

We identified a set of nine leadership roles: Method Champion, Disciplinizer on Equal Terms, Coach, Change Agent, Helicopter, Moderator, Networker, Knowledge Enabler and Protector. We have found that those roles are rather centred on one dedicated Scrum Master in early stage while the roles are rather distributed in more mature teams. We further identified the role transfer process which involves a supportive internal team climate and a leadership gap that enables team members to take on leadership roles.

The main contribution of this paper is to provide a deep insight into the agile transformation at a conglomerate that widely uses the Scrum method to empower self-organized teams. This paper extends our previous conference paper on the changing Scrum Master (Spiegler et al. 2019) by the following parts: we add more content on the theoretical background in Section 2. We provide additional data on the nine leadership roles in Section 4.1 and new data on the internal team environment in Section 4.3. Moreover, we provide evidence on the impact of the retrospective on the internal team environment in Section 4.4. We further describe limitations of the role transfer by referring to role conflicts in Section 4.5.

Our results help practitioners to understand how teams change from a rather traditional way of working to an agile approach by focusing on leadership. This also helps managers to learn how they can support developers in their agile way of working.

The rest of the paper is organized as follows: In Section 2, we refer to related work on the Scrum Master and describe how maturity helps us in understanding changing leadership in agile teams. In Section 3, we outline our study design. In the following Section 4, we present the results of our qualitative interviews including the description of the nine leadership roles and the role transfer process. We discuss how our findings relate to other research in Section 5. In Section 6, we provide practical implications, and in Section 7 we refer to limitations and suggest future research.

2 Related Work

We aim at exploring the change in leadership using the example of the Scrum Master. We will first refer to research related to maturity in agile teams, and afterwards elaborate on research on leadership in agile teams.

2.1 Maturity

Team literature research differentiates between dynamic and static teamwork models. While static teamwork models refer to teams that are stable and have successfully reached a constant mature stage, dynamic models assume that a team undergoes different maturity stages. This study focuses on dynamic models because we believe the differentiation between mature and immature teams may help us to explain a change in leadership.

Since the first empirical study on group development phases by Bales and Strodtbeck (1951), different authors suggested conceptual models on group development phases among which the probably most famous one is the forming-storming-norming-performing model by Tuckman (1965). Tuckman differentiates between several phases:

The forming phase sets ground rules for interaction between a leader and the team as well as between team members. The storming phase often involves conflicts due to absence of unity and insecurity. The norming phase helps teams to build a shared understanding of roles and responsibilities (Neuman and Wright 1999). The performing stage describes that team members play roles flexibly according to what makes most sense in a given situation (Tuckman 1965).

While early researchers suggest successive phases on group development, later on authors criticized this approach and suggested that each group develops differently.

For example, Gersick (1989) and Ginnett (2019) 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 (Marks et al. 2001).

The maturity model by Tuckman was shown to be suitable to explain different development stages of agile teams. Gren et al. (2017) found a relationship between the group development stages and team agility and speculate that depending on the maturity level of a team, developers practice the agile way of working differently. For example, they 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-organize better which allows the Scrum Master to step back. They furthermore identify the retrospective to be a tool that helps the team in developing maturity (Gren et al. 2019).

This paper explores how leadership changes while the team matures.

2.2 Leadership in Agile Teams

We investigate leadership in agile teams using the example of the Scrum Master. Most papers on the Scrum Master are based on personal experience and common sense and lack in theory and structured approach to data collection. The leadership perspective on the Scrum Master varies among researchers. Some authors elaborate on research that focuses on individuals as leaders, while other researchers assume that leadership in agile teams can also be conducted by several individuals.

Bäcklander (2019) refers to complexity leadership behavior and considers the Scrum Master to be an enabling leader without disciplinary power. Often developers grow into Scrum Masters over time. Since its application to a software company the role description has evolved from specifically focusing on the Scrum Method to caring for team dynamics. Furthermore, the Scrum Master continuously balances providing a structure, e.g. with the help of the retrospective, and not providing a structure, e.g. by being absent. Bäcklander (2019) concludes that leadership literature and team literature are tightly linked together, since enabling leadership fosters team processes, such as team learning (Edmondson 1999). She suggests further research on how team members can take on leadership.

A study on group development of agile teams (Gren et al. 2017) refers to *situational leadership theory* which derives from the contingency stream. Contingency theory suggests that the situation influences the effectiveness of a leadership style (Fiedler 1967). Situational leadership describes that the leader should change behavior if the followers develop (Hersey et al. 2007). Gren et al. (2017) speculate that a rather immature team needs structure and order to organize itself, while a mature team is able to self-organize which allows the Scrum Master to step back (Gren et al. 2017). Moreover, leaders adapt their behavior to company specific culture and structure (Gren and Lindman 2020).

In contrast to the aforementioned perspective on the Scrum Master being linked to one individual leader, other authors refer to leadership sharing between one dedicated individual and the agile team (Moe et al. 2010; Srivastava and Jain 2017).

Moe et al. (2010) observe teamwork challenges of a newly established Scrum team over a period of nine month. They refer to *team leadership* which means in their context to provide direction, structure, and support for other team members, such as active listening, and can be shared among Development Teams, Product Owner and Scrum Master. Moe et al. (2010) reveal that the Product Owner and the Scrum Master rather reduced team leadership by starting to control team members and not listening to their concerns. Consequently, developers demonstrated a lack in trust and willingness to reveal their impediments openly. However, the authors acknowledge that while the team matured, team members started to take on more responsibility and therefore team leadership improved over time. Moreover, the authors state that teams need management support and resources to grow into self-organization.

Srivastava and Jain (2017) present a leadership framework of the Scrum Master in locally distributed teams that had been working in an agile way for three years on average. They refer to *super leadership* (Manz and Sims 1987), which describes that a leader enables a team to lead itself. They conclude by suggesting rotational leadership, which implies that the Scrum Master is shared among team members depending on the situation. Moreover, they refer to Carson et al. (2007) and acknowledge that the development of team leadership needs to be accompanied by shared purpose, social support and voice.

2.3 Discussion of the Literature

This paper focuses on leadership in agile teams using the example of the Scrum Master. The scarce empirical findings reveal contradicting results on the Scrum Master. While some authors refer to the Scrum Master as an individual leader (Bäcklander 2019; Gren et al. 2019; Gren et al. 2017), others refer to shared leadership (Moe et al. 2010; Srivastava and Jain 2017). Some authors describe that a Scrum Master struggles with command-and-control behavior, while team members neglect to take on leadership responsibilities (Noll et al. 2017; Stray et al. 2011). Researchers still struggle with the inherent paradox of leadership in a team that is supposed to be self-organizing. Thus, leadership in agile teams is not yet sufficiently understood.

We aim to contribute to research on agile teams by focusing on changing leadership. So far, researchers have not built a bridge between one dedicated leader in an immature team and leadership sharing in a mature team.

The Scrum Master either had been found to be centered on one dedicated person in an immature team (Moe et al. 2010) or has been suggested to be played by multiple group members in mature teams (Srivastava and Jain 2017). Moreover, the company context (Gren and Lindman 2020) and the team climate (Carson et al. 2007) influence leadership. Our aim is to explore and portray the characteristics of a Scrum Master in immature as compared to mature teams and to describe how leadership changes. To the best of our knowledge, the theoretical suggestion of changing leadership while the team matures (Gren et al. 2017) has never been empirically investigated.

This paper aims to explore how leadership evolves while the team matures, in such a way that leadership is shared between the Scrum Master and the Development Team over time. Our results indicate that the Scrum Master plays diverse leadership roles which can be transferred from one individual to distinct team members during the maturity journey.

While empirical data on the Scrum Master is often missing or merely based on single case studies, we provide qualitative data from multiple teams operating at different companies of one conglomerate.

3 Study Design

3.1 Research Method

We aim to explore changing leadership in agile teams. Research on human interaction in software development teams is still scarce. We chose grounded theory because this method is applied in research fields with scarce knowledge (Glaser and Strauss 2017).

This research method requires to start collecting empirical data without a specific research problem and research question in mind, since this is expected to emerge during the research project at the field. Following grounded theory we collected data in an iterative manner based on emerging patterns and themes, while constantly comparing emerging data (Glaser and Strauss 2017).

Grounded theory can be conducted by applying a positivist and constructivist view (Bryant and Charmaz 2007). 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 (Bryant and Charmaz 2007). Our 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 (Charmaz 2016). 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 (Charmaz 2016). We search for underlying assumptions that foster specific behavior (Charmaz 2016).

While we apply the iterative approach of the data analysis method by Glaser and Strauss (2017), we take a constructivist view when interpreting the data.

3.2 Company Context

We conducted this study at Robert Bosch GmbH which employs more than 410,000 people in 60 different countries worldwide. The company history dates back to 1886 and can therefore be classified as established company. 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.

While the agile transformation started as a headquarter project that supported divisions in their agile journey, now each division is responsible for its own agile transformation. One wildly used agile approach across various sub-companies is the framework Scrum. Some sub-companies implement Scrum without changing the organisational context (e.g. structure). Other sub-companies implement the Scrum framework while also transforming the whole organisation. Even though a corporate role description of and training for a Scrum Master exists, there is neither an obligatory training nor mandatory leadership responsibility regarding the role.

Internal trainers offer corporate training on diverse topics of the agile way of working, yet, each team can decide on its own how to educate the team and the Scrum Master. For example, a team could book a training internally or externally of the company or not book a training at all.

With scarce exception Scrum Masters are without disciplinary power, responsible for the Scrum process and in charge of team development. Moreover, most practitioners of the company call the Scrum Master *Agile Master*, indicating that this role should adapt to the specific team, rather than sticking to the Scrum approach by the book. Interviewed teams stated that they apply the Scrum method mostly in modified form, e.g. w.r.t. the regularity of Scrum meetings.

3.3 Data Collection

Since the first and second author are employees of the company, we had direct access to the field. We collected data by conducting interviews and observations, and collected feedback on emerging results from practitioners. We identified Scrum practitioners either via our personal network or via the company's social media platform and first contacted them by email. We collected data from 11 business divisions which have slightly different subcultures.

Due to confidentiality requirements by the Robert Bosch GmbH, the raw data cannot be provided openly.

Interviews Between June 2017 and November 2018 we conducted three different turns of interview collection based on previous findings. Most interviews were conducted in German and later on translated into English. Some interviews were conducted in English.

The first round of interviews included 22 individuals from 10 different sub-divisions. The sample contained all kind of organizational roles including but not limited to Scrum Masters, developers and organizational coaches. The second and third round of interviews includes 22 Scrum Masters, 8 Product Owners and 23 team members from 14 software development and 15 non-software project teams. Some but not all Scrum Masters were certified Scrum Masters. Likewise, some but not all Scrum Masters visited meet-ups and internal conferences to exchange knowledge with other Scrum Masters. 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, we expected the maturity of teams to vary.

We conducted most qualitative interviews face-to-face while very few interviews were conducted via skype. We scheduled the meetings according to availability and willingness to take part. One interview lasted approximately 45 minutes on average. While the first set of interviews was based on hand-written notes, the second and third round of interviews were audio-taped and transcribed.

Observations It is recommended to supplement semi-structured interviews by observations to understand the context of the interviews (Adolph et al. 2008). The main author observed Scrum events, such as the daily stand-up, review, planning and retro, of sixteen Scrum teams from four different sub-companies. She 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, the author made nodes and asked clarifying questions afterwards if necessary.

Feedback We collected feedback from Scrum practitioners by presenting preliminary findings to observed teams and 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 the main author made nodes and the participants of the workshops send their group work results to the main author. This material was used to refine and strengthen the developed concepts and add to validity.

Memos During our data collection the first author wrote memos. She wrote down emerging questions, new ideas and relationships between codes and categories while collecting, analyzing and discussing data. Moreover, the memos stored ideas about links between the data and relevant literature.

3.4 Data Collection Procedure

Grounded theory suggests theoretical sampling (Glaser and Strauss 2017) in which each step determines the next step to be taken during the research based on previous results. The iterative approach (Glaser and Strauss 2017) lead us to the above enumerated three rounds of data collection which will be described thoroughly in the following.

First Round Grounded theory requires a general up-front research topic (Hoda et al. 2012), which was leadership in agile teams.

The research problem should emerge from a challenge that practitioners face (Glaser and Strauss 2017). During the first round of unstructured interviews with practitioners from various business divisions and diverse organizational roles, we realized that there was a lack of clarity regarding the Scrum Master. Practitioners agreed that a Development Team was expected to be self-organizing, yet, how the Scrum Master was supposed to enable the team to do so was not entirely clear. Moreover, while some practitioners referred to the Scrum Master as a leader, others opposed the idea that the Scrum Master should be labeled a "leader". Comparing our results with the Scrum Guide and with scientific literature we found that there was not yet sufficient information to fully explain leadership of the Scrum Master as it is practiced in this organizational setting.

The purpose of the first round was to get an overview on relevant topics and derived data is not included in the results section.

Second Round Thereafter we conducted a second set of interviews mainly interviewing Scrum Masters besides a minority of developers. A semi-structured questionnaire focusing on the development of leadership behavior of a Scrum Master guided the interviews and allowed further questions when the answer of an interviewee appeared to offer more insights. Interviewees were asked to describe what they did to support agile teams and what they had learned since they had started to play the Scrum Master. The guiding questions are available online (Spiegler et al. 2018).

We identified nine leadership roles which the Scrum Master performs. Yet, the interviews revealed that not only the Scrum Master conducted leadership but that developers tended to take on some of the Scrum Master activities, as well.

Third Round We developed a second questionnaire that focused on sharing leadership between the Scrum Master and the Development Team with a specific interest in how the sharing of leadership had developed over time. Moreover, we asked questions regarding supporting and hindering factors of taking on leadership.

Our third round of interviews approached entire teams and viewed the Scrum Master from three different angles: the Product Owner, the Scrum Master and the Development Team.

This helped us in understanding how the Scrum Master evolved while the team matured and how leadership activities were shared among developers and the dedicated Scrum Master. We came to realize that Scrum teams without a Scrum Master struggled with working in an agile way.

Theoretical saturation of data describes that data collection stops if data analysis does no longer reveal new patterns (Charmaz 2016). We stopped collecting data when we perceived that the content of the interviews was repetitive and data analysis revealed no more new insights.

During the research process the following research questions emerged

RQ1: How does the Scrum Master perform leadership in an agile team?

RQ2: In which way do team members take on leadership over time?

RQ3: How is leadership transferred from the Scrum Master to the Development Team?

RQ4: What is the underlying internal team environment required for the transfer of leadership?

RQ5: How can the Scrum Master foster the internal team environment?

RQ6: Which expectations on the Scrum Master limit the changing leadership?

3.5 Data Analysis

We followed the instructions for Glaser's open coding as described in the dedicated paper by Hoda et al. (2012). While Glaser and Strauss (2017) have a positivist view, we had a constructivist perspective for interpretation of data (Charmaz 2016).

We openly coded transcripts sentence-by-sentence. We aligned codes that appeared to be alike to one concept. The content of the interviews was constantly compared within the same interview and across interviews. We constantly reflected and compared emerging concepts critically. Observations helped us to place the content of the interviews into context.

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 we identified nine different concepts during our iterative data analysis. Our 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 organizational goals or both. We 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.

Scrum Masters explained that they gradually lead the team less and also that sometimes they empower the team by keeping back, which we labelled *leadership gap*. Through constant comparison (Glaser and Strauss 2017) of various interviews and observations we identified nine different leadership roles and developed a substantive theory (Glaser and Strauss 2017) which we termed the *role transfer process*.

While analysing our data we compared emerging concepts with existing research in agile software development. We 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 (Parizi et al. 2014).

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.

3.6 Validity Procedure

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 part of the interview had finished, the interviewees made contradicting statements to what they had claimed during the interviews. Since the authors could not be sure whether social response bias applied, those three participants were excluded from the sample.

The aim of a constructivist study is to reveal the reality how research participants perceive it (Bryant and Charmaz 2007). To ensure that we captured the perception of the research participants accurately, we received feedback from practitioners as described in Section 3.4, and from the research community during presentations and informal discussions. Moreover, we did not only conduct interviews but enriched our data by observations and further feedback sessions such as workshops. Despite these measures, our data analysis involves subjective interpretation of the researchers which is a well-known issue in grounded theory (Charmaz 2016).

4 Results

In this section, we illustrate the findings of our interviews.

We grouped our findings on leadership in agile teams into the following nine different leadership roles that empower a team to work in an agile way: Method Champion, Disciplinizer on Equal Terms, Coach, Change Agent, Helicopter, Moderator, Knowledge Enabler, Networker and Protector. While some teams described that the leadership roles were rather centred on the Scrum Master, other teams outlined that leadership of the Scrum Master had changed over time. In the latter cases, developers started to take over some of the leadership roles themselves and the Scrum Masters diminished the extent to which they played those roles.

We 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 team (RQ2). Thereafter, we refer to the role transfer process which build the bridge between both levels of analysis (RQ3). We further describe the underlying internal team environment which serves as an enabler of transferring leadership (RQ4) and the supportive role of the retrospective (RQ5). We conclude by reporting on role conflicts which limit a change in the Scrum Master (RQ6).

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

4.1 Nine Leadership Roles

4.1.1 Role 1: Method Champion

The interviews revealed that teams often adapted the Scrum Method to their specific context. Even though the Scrum framework may help a team to work in an agile way, Scrum was not always perceived to be beneficial in every team setting. Often the team needs support in finding the appropriate method for a specific context (Moe et al. 2009; 2010).

Well, I actually bring a broad method toolbox and expertise to the table. I've noticed it everywhere, like creative techniques. It's 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)

While some Scrum Masters insisted on doing Scrum according to the book, a large majority of Scrum Masters mentioned conducting and adapting the method to be their main task when working with agile teams. We label this role *Method Champion* which describes activities related to the implementation and application of the Scrum Method. On the one hand the Method Champion adapts the Scrum method to the specific team context, on the other hand the role aims at realizing when the Scrum approach is not the most fitting one regarding the context. Some Scrum Masters described that they introduced other methods besides Scrum, e.g. XP or Kanban.

Also two Development 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. Moreover, some teams stated that the Scrum Master had initially organised team events but after some time the team members organised such events themselves.

4.1.2 Role 2: Disciplinizer on Equal Terms

Initially, some team members were reluctant to follow the Scrum process. Some Scrum Masters reported that when they insisted on discipline, 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. We call this role *Disciplinizer on Equal Terms* which describes to help the team to focus on respective tasks and stops developers from multi-tasking.

> *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)

However, there were also developers who did not appreciate this role and rather felt they were forced to behave in a certain way. While some developers welcomed timekeeping, others felt that the Disciplinizer was more interested in following a process than caring for people. Since one agile principle involves *people over processes*, we are not entirely convinced that the Disciplinizer can be considered an agile role. Therefore, we add the *on equal terms* to the name. In agile settings, individuals treat each other unrelated to title or position (Hoda et al. 2012).

The aim of the role is to help the team developing focus and discipline without directly monitoring teams top-down but communicating on a par. During different Scrum events we observed, that the Disciplinizer on Equal Terms did rather not involve forcing the team to focus by referring to hierarchical power, e.g. by asking for status reports.

We assume that interaction on equal terms creates non-hierarchical spaces which are important to speak openly with each other, suggest new ideas and allow every team member to take on responsibility.

While at the beginning the Scrum Master was more concerned on discipline, also the developers started to emphasize on the agile value *focus*. Some improved focusing on their operational tasks.

> *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)

Whereas other developers reminded their fellows to stop endless discussions.

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

We also observed in several teams that after 2 to 3 months developers started to tell the Scrum Master that he or she should stick to agreed rules, e.g. not being late for meetings or only talking for a certain amount of time.

4.1.3 Role 3: Coach

Often interviewees claimed that they observed team behavior to identify which kind of team behavior was missing for executing the agile way of working and to help the team to develop this behavior. Some interviewees also said that they brought developing conflicts to the surface. We label this role *Coach*. A coach helps developers to constantly develop themselves further by setting triggers, such as asking the right questions at the right time (Kozlowski et al. 2009), and by encouraging for self-criticism, observation and feedback delivery (Manz and Sims 1987). Several interviewees described these activities.

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

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)

Therefore, this interviewee did not act as a visible leader, but rather supported cautiously from the background. Previous research has described that an immature team that aims to become self-organized needs a supportive coach (Wageman 1997; Magpili and Pazos 2018). Yet, we found that the *Coach* was not linked to a specific person like the Scrum Master, but that anyone in the team could act as a Coach. For example, one developer described how the team established psychological safety over time during the retrospective.

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)

This developer expressed how mutual trust created an atmosphere within which team members felt safe providing feedback to each other. When trust among team members increased, it was no longer merely the Scrum Master who revealed personal observations.

While the Method Champion knows distinct project management methods like Scrum, Kanban or XP, and can consult the team on the different methods, a Coach helps the team to understand which way of team behavior needs improvement. Someone who acts as a Coach may be good at asking the team the right questions, but may not necessarily know anything about project methods.

4.1.4 Role 4: Change Agent

The *Change Agent* relates to two different facets: convincing stakeholders of the agile way of working and serving as a role model.

Some individuals would persuade managers or other stakeholders to behave in a more agile way. The overall aim is to convince individuals why agile values and principles are relevant and useful.

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

This quote allows to speculate that dedicated leaders who expect the team to behave in a more agile way, but do not embrace agile values and behavior themselves, will most likely not inspire developers to work in an agile manner.

Some Scrum Masters described to serve as a role model by believing in the agile values.

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)

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 team members become more and more convinced of the agile manner, hence the Change Agent is played to a lesser extend.

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)

This example demonstrates that 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.

It may make sense to start an agile transformation with a team that embraces several team members that naturally bring the agile values with them. Those will than inspire others.

While the Coach helps the team to develop itself further by asking the right questions, and the Method Champion focuses on implementing the right project methods, the Change Agent serves as a role model to others.

4.1.5 Role 5: Helicopter

Teams have a cross-functional understanding of the overall project, in order to be able to self-organize their work and to make fast high quality decisions towards a common goal (Cockburn and Highsmith 2001; Takeuchi and Nonaka 1986).

During our interviews we identified a role we call *Helicopter* which implies the ability to catch the big picture of a project, to understand how topics are interrelated and to know who possess the right skill for a certain task within the team.

During the observation of one team over several months, we observed that a Scrum Master constantly took on the Helicopter during the daily. Yet, this limited the degree to which the developers acted as Helicopter. The Scrum Master complained, that the Development Team did not work in a more cross-functional way, and that he had to be the mediator of the diverse threads. After this conversation during the Retrospective, the developers started to pay more attention to each other's skills and competences.

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)

Also in other heterogeneous teams the developers learned to have the bigger picture in mind. Due to daily communication and visualisation, team members formed a mutual understanding (Moe et al. 2010; Hoda 2011) while they matured, so that they knew who had which knowledge or skills, and whom they should hand over a task if they have finished it and needed someone else to continue working on it.

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)

We speculate that the reason for the Helicopter is that 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 (Levesque et al. 2001) 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 (Hoda et al. 2012).

4.1.6 Role 6: Moderator

Agile teams work cross-functionally which implies varying cognition, behavior and personalities (Takeuchi and Nonaka 1986). This may involve different domain languages and working habits, and may cause a lack in understanding each other properly.

The *Moderator* moderates all kind of meetings and builds a bridge between perspectives and domains, and thereby helps the cross-functional team develop a shared understanding.

Several 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)

We have not come across any team within which the Development Team 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 be the Moderator since it was considered to be difficult to remain neutral during a discussion while being part of the operating team.

While Helicopter focuses on the product as a whole and links technical issues to individual's skills, Moderator makes sure that developers talk about the same and discuss constructively with each other.

4.1.7 Role 7: 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.

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)

This interviewee felt that the Scrum Master 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. The role 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.

The Knowledge Enabler supports and reminds the developers of team learning (Edmondson 1999) and of transparent knowledge sharing. For example, the role encourages learning sessions for sharing knowledge openly with colleagues and for learning from each other.

Some Scrum Masters urged developers to take time for learning and to share knowledge more openly with each other.

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)

In a traditional organisation, developers expect their supervisors to own deep technical knowledge and teach them. 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. Developers also described to simply approach colleagues to ask for information, to learn from each other and to work together on tasks. For example, some developers would do pair programming.

Agile projects often tackle new challenges which require an inspect and adapt approach and learning on the job (Takeuchi and Nonaka 1986). The technical issues have not been solved before, and thus the classic approach to learning is less helpful. Therefore, developers learn iteratively and apply team learning (Edmondson 1999).

4.1.8 Role 8: 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 (Takeuchi and Nonaka 1986). The agile team connects to external parties depending on specific Sprint goals.

We uncovered a role *Networker*. It connects the team with relevant stakeholders, from within and outside the organisation, and thereby ensures that the team has access to required support that goes beyond the competences and skills within the team.

Several Scrum Masters reported to approach relevant stakeholders depending on the team's demand. For example, they invited management to gain support 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 Organisation or even outside the project. (AM)

Over time, developers increased their personal network and learned whom they could approach for what. Moreover, during dailies team members 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)

The Networker bridges the team with the organisation. Developers can approach any stakeholder unrelated to position or hierarchy. A project team embedded in a traditional hierarchy has to keep to a official, formal communication chain, and often a team and superior managers do not talk directly to each other. Also colleagues from neighbouring departments tend to not speak directly with each other but via department managers. A team leader or group leader may serve as a mediator. In an agile team, developers can directly approach any colleague or manager in the organisation.

4.1.9 Role 9: Protector

Like Networker, *Protector* is a role that focuses on the organizational setting. Development Teams need interference-free space to focus on their respective Sprint goals (Stray et al. 2011). Therefore, the team needs a Protector who shields the team from disrupting and unreasonable requests (Stray et al. 2011). The Protector aims to maintain trouble-free working conditions as Development Teams are meant to focus on their respective sprint goals only.

> *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)

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

> *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)

However, multiple team members reported on occasions when they encountered disturbances by stakeholders, yet, no team had developed a strategy how to protect themselves successfully from such disturbance when the Scrum Master was absent.

We came across one approach by a team 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. So in terms of learning, people gradually became to learn. Even though the technology felt new to most of us.(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.

While the Disciplinizer on Equal Terms strengthens focus *in* teams, the Protector shelters the team from the organization.

We explored which role the Scrum Master plays (RQ1) and in which way team members play that role over time (RQ2). While Scrum Masters were found to perform nine diverse leadership roles in rather immature teams, more mature teams played some of the roles themselves. Besides, we discovered that some roles were more applicable to be performed by mature teams than others. In the following, we will portray how roles were transferred from the Scrum Master to the team.

4.2 The Role Transfer Process

Our goal was to examine how the Scrum Master transferred leadership roles to team members (RQ3) and discovered that the role transfer occurs during **three consecutive steps** we labelled the **role transfer process** (shown in Figs. 1 and 2). In the following we elaborate on the three steps:

The first step illustrates that the Scrum Master serves as a role model by demonstrating the activities of the nine leadership roles while team members 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.

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)

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.

The second step describes that the Scrum Master provides a leadership gap when the team is considered to be more mature.

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

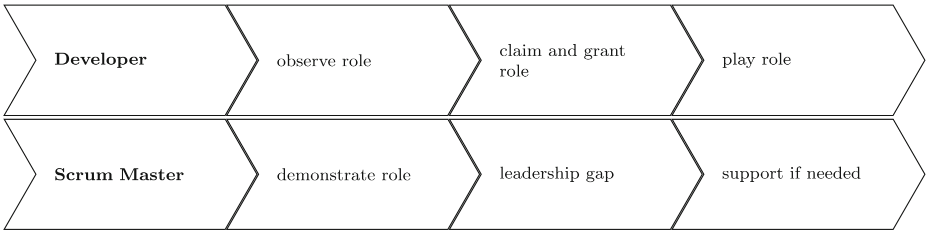

Fig. 1 The three steps of the role transfer process

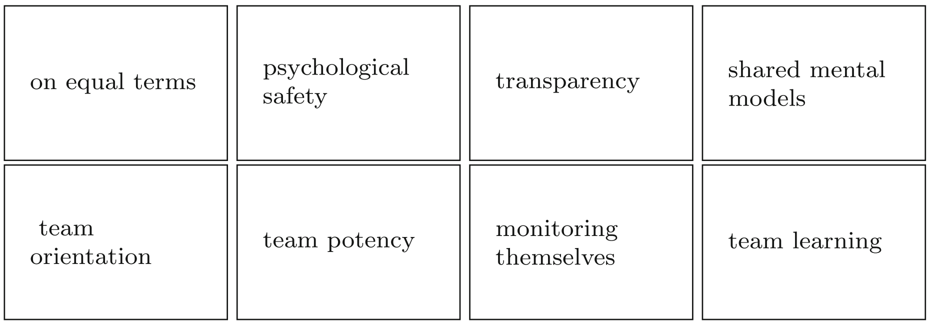

Fig. 2 Factors describing the Internal Team Environment

self-organisation. Because if you do not create some free space or a vacuum, nobody will jump in.(AM)

This quote reveals that 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 team members 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.

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)

This quote exemplifies that a Scrum Master provided a leadership gap intentionally, such that developers could perform the Knowledge Enabler.

Scrum Masters claimed that they either prepared a leadership gap intentionally by withdrawing from particular roles while waiting that team members 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.

The third step contains that the Scrum Master only plays certain roles when necessary while team members perform most of the roles themselves.

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)

Yet, we found that not every role can be passed on to the team equally, e.g. Moderator and Protector tended to remain with the committed Scrum Master. This indicates that the dedicated Scrum Master does not disappear in a mature team but is played to a lesser extent over time.

4.3 Internal Team Environment

Furthermore, we aimed at exploring the team enablers required for the role transfer to occur (RQ4). We found **eight enablers** shaping an **internal team environment** that stimulated team members to take on leadership roles.

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 team members and the Scrum Master. One developer described the hierarchical free space in the following way:

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)

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.

Secondly, teams who communicated on equal terms had established **psychological safety** (Edmondson 1999; Moe et al. 2010) 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.

Additionally, as referred to in Section 4.2 team members had developed a **shared mental model** (Levesque et al. 2001) 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.

Moreover, **team orientation** (Moe et al. 2010) was identified to be important for developers 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, we found that team members who performed leadership roles increased their **team potency** which implies to believe in the team's capability to be successful (Guzzo et al. 1993). 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 developer described it to be painful to take on a leadership role when facing the leadership gap initially.

Additionally, developers felt that **self-monitoring** (Moe et al. 2010) 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.

If 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** (Edmondson 1999) allows the team to continuously reflect upon the leadership roles, e.g. during the retrospective, and therefore, develop themselves further regarding playing the diverse roles.

Answering our third (RQ3) and fourth research questions (RQ4), we found that **an internal team environment** of communication on equal terms, psychological safety, transparency, shared mental models, team orientation, team potency, monitoring themselves and team learning enabled roles to be transferred from the Scrum Master to developers.

4.4 Retrospective

Some teams reported to struggle in developing the internal team environment needed for transferring the roles. Our 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 we 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.

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

Furthermore, the retrospective established *psychological safety* which is necessary to have the courage to take on the diverse 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*. Development 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.

> *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 we found that the retrospective promoted different facets of the internal team environment. We therefore conclude that the Scrum Master empowers the team to develop a supportive team climate and consequently take on leadership roles by facilitating the retrospective.

4.5 Role Conflicts

So far, we have described that the Scrum Master tends to play nine different leadership roles which are handed over to the team via a leadership gap, a supportive internal team environment and the retrospective. Yet, despite a supportive internal team environment, still the leadership roles may not be transferred to the team. 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. Comparing our findings with other studies we learned that also in other companies Scrum Masters tend to perform additional activities, such as project management (Noll et al. 2017), which contradict the nature of the agile way of working, and led to role conflicts (Noll et al. 2017; Stray and Sjøberg 2016).

Our sixth research question was therefore: *Which expectations on the Scrum Master limit the changing leadership?* (RQ6)

In the following we will refer to diverse expectations by the Development Team, managers and the Product Owner, as well as requirements due to the position of the Scrum Master in the organisational chart.

4.5.1 Development Team

Often team members expected the Scrum Master to behave in an agile way, e.g. allowing the team 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 developers also expected the Scrum Master to take on typical project manager tasks, e.g. providing clear direction or taking over responsibility for team decisions. These behaviors are in contrast to maintaining a leadership gap within which the team takes 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 organizational 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 organization would not allow that.* (Dev)

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

4.5.2 Managers

Managers involved the disciplinary supervisor of the Scrum Master or of the Development Team, internal customer and line managers. A few managers expected the Scrum Master improving 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.

> *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 Development Team should take on leadership and is 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 Coach for the agile approach. The Scrum Master was torn in between meeting traditional management requirements and embracing agile leadership behavior, such as providing a leadership gap.

4.5.3 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 Development Team. This led to the Scrum Master also putting more pressure on the team instead of opening the leadership gap. Furthermore, it diminished the willingness of the Development Team to take on leadership roles.

That [the Product Owner position] is not really a nice position to be in in such an organization. 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)

4.5.4 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.

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.

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 Development Team, 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 Development Team takes 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.

5 Discussion and Relation to Existing Evidence

5.1 Summary of the Research Findings

There is a steadily increasing number of attempts to implement agile teams in industrial setting. Simultaneously, a growing body of papers point at challenges such teams face (e.g. (Moe et al. 2012; Nerur et al. 2005)). Scarce empirical research explores how teams can actually learn to work in an agile manner. Our research objective was to explore how leadership empowers a team to work in an agile way in real organizational settings with a particular focus on the changing Scrum Master.

Empirical studies have suggested that leadership evolves while the team matures (Moe et al. 2010; Gren et al. 2017; Srivastava and Jain 2017). Some authors state that team members in a more mature team also take on leadership roles (Moe et al. 2010; Srivastava and Jain 2017). Up to our knowledge, no scientific research project has examined how leadership changes in an agile team over time. We explored how the Scrum Master enables the developers to take on leadership and to adapt the agile way of working.

We reported on the results of a grounded theory study. Based on observations and semi-structured interviews with 53 Scrum practitioners from 11 different business divisions of the Robert Bosch GmbH we answered research questions RQ1 to RQ7.

In the following we summarize our research findings by referring to the respective research questions in consecutive order.

How does the Scrum Master perform leadership in an agile team? (RQ1) We identified that the Scrum Master plays 9 leadership roles. Each role aims at fostering agile team features. Moreover, the goal of one dedicated Scrum Master is to empower the developers to perform the nine leadership roles themselves.

In which way do team members take on leadership overtime? (RQ2) The semi-structured interviews with Development Teams, Scrum Masters and Product Owners provided examples for the team 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. We did not come across examples in which the team played the Protector and the Moderator.

How is leadership transferred from the Scrum Master to the Development Team? (RQ3) Leadership is transferred in three consecutive steps which we name the *role transfer process*. Firstly, the Scrum Master is a role model and demonstrates nine leadership roles while the 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 other formal leader role interferes. This allows the developers to take on leadership. 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 team performs most of the nine roles.

What is the underlying internal team environment required for the transfer of leadership? (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 team to understand the leadership roles and agree upon sharing the roles.

How can the Scrum Master foster the internal team environment? (RQ5) By facilitating the retrospective the Scrum Master helps developers to grow 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? (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 aiming at control of work progress, even though the agile way of working describes that the Development Team monitors itself aiming at transparency. As a consequence of the role conflicts, the Scrum Master is torn between clinging to the leadership roles and handing them over to developers. If the Scrum Master decides to fulfill expectations deriving from a bureaucratic system, the role transfer is diminished.

5.2 Nine Leadership Roles

Following grounded theory, we did a minor up-front literature search before collecting and analysing data. While developing our grounded theory grounded in the gathered data, we compared it to current research on agile software development teams. In the following, we elaborate on the similarities of and differences between our findings and existing research on agile teams.

Our interviews revealed that a Scrum Master plays multiple leadership roles which are transferred to the team while it matures. Consequently, roles are shared between the Scrum Master and a mature Development Team. Our suggestion that developers can learn to take on parts of the Scrum Master is in line with the findings of other researchers (Bäcklander 2019; Moe et al. 2010; Srivastava and Jain 2017).

Our data analysis uncovered the following nine leadership roles:

1. Method Champion: makes sure that the method fits the specific team context. Brings new methods to the team and discusses how to adapt the method to the team context.
2. Disciplinizer on Equal Terms: ensures that the team focuses on relevant topics. Discipline is accomplished via communication on a par.

3. Coach: Observes colleagues and uncovers which kind of behaviour is missing in the team to improve teamwork, provides feedback, and helps others to find out what they wish to change and how to do so.
4. Change Agent: Serves as a role model, and thereby convinces newly established project teams of the agile way of working.
5. Helicopter: Has the ability to see the bigger picture and knows who possesses the right skill for a certain task.
6. Moderator: Moderates all kind of meetings and builds a bridge between perspectives and domains.
7. Knowledge Enabler: Teaches the team a new approach to learning and promotes iterative learning.
8. Networker: Connects the team with relevant stakeholders from within and outside the organisation.
9. Protector: Shelters teams from inappropriate requests from the Product Owner, managers, disciplinary leaders and other departments.

A similar grounded theory study by Hoda et al. (2013) discovers six different self-organizing roles (Mentor, Coordinator, Translator, Champion, Promoter, and Terminator). Up to our knowledge, the research by Hoda et al. (2012) is the only scientific study related to human behavior in agile software development exploring descriptive roles in self-organizing agile teams. Therefore, we compare our findings on leadership roles in agile teams with the findings on self-organizing roles in agile teams. Even though our interviews focused on *leadership* of the Scrum Master, we discovered similarities to *self-organizing* roles in agile teams (Hoda et al. 2012).

When starting our research, we had not been familiar with the self-organizing roles. The research by Hoda et al. (2012) refers to a job description of an *Agile Coach* who plays most of the self-organizing roles in an immature team, but shared the roles in a more mature team. Similarly, we found that the *Scrum Master* played most of the nine leadership roles in a rather newly established team. Yet, even though the content of the roles overlap they slightly differentiate.

Comparing our findings on leadership roles with the work by Hoda et al. (2012) on self-organizing roles, the role descriptions of the Champion, Mentor and Translator clearly overlap with our Change Agent, Method Champion, Networker and Moderator. Yet, the Change Agent does not only convince management of the agile way of working (like the Champion) but also persuades team members of the agile manner. The Networker connects the team with management (like the Champion). Moreover, the Networker also builds a bridge between the developers and experts externally to the team. The Method Champion does not only refer to the implementation of an agile method (like the Mentor), but also consults the team upon which method (e.g. Scrum, XP, Kanban) to use in a given context, and how to adapt the Scrum method to a specific context.

Translator is similar to Moderator. While the Translator refers to translating between the business context of the customer and the technical context of the developers, the Moderator focuses on mediating between team members and is more comprehensive. The Moderator does not only translate between different domain languages, but also between working habits or cognition.

Coordinator and Translator (Hoda 2011) involve customer representation. We have not identified roles that relate to the customer. In Scrum the Product Owner is responsible for the customer integration. Probably we did not identify a role related to the customer integration, because our study focused on the leadership role of a Scrum Master.

Moreover, we found additional roles, such as the Coach, which is not described by Hoda et al. (2012), but by the study of Bäcklander (2019). The Coach observes team behavior, actively listens to the team's concern and helps the team to get along well with each other, to talk openly about developing conflicts and to deliver feedback to each other. While Bäcklander (2019) examines an individual leader within the team and leaves it open whether or not team members can take on leadership, we found that also several team members performed the Coach.

Hoda et al. (2012) describe that mature teams tend to share the self-organizing 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. While Hoda et al. (2012) suggest that the self-organizing roles can be performed by any team member in a mature team and Bäcklander (2019) even speculates if the Scrum Master can be replaced by team members, we do not believe that all leadership roles can be shared equally among the Development Team and the Scrum Master, but instead claim that the Scrum Master keeps the Protector and the Moderator role in a mature team.

Therefore, we suggest that some roles should always be played by the Scrum Master, which builds upon the findings by Hoda et al. (2013) 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. (2019) we found the retrospective to be an important enabler for continuously fostering an internal team climate that stimulates the role transfer.

Grounded theory is always embedded in the context under research (Glaser and Strauss 2017). We speculate that we have identified these particular nine leadership roles in our research context, because they are typical for bureaucratic companies that are in the middle of their agile transformation. Traditional development companies are build upon departmentalized silos as opposed to cross-functional teams (Nerur et al. 2005), upon top-down hierarchy as opposed to self-organization (Moe et al. 2012), and rigid planning as opposed to iterative learning (Boehm and Turner 2005).

The Protector may protect the team from a rather hierarchical organizational context. The Method Champion supports teams that are used to the waterfall model, to learn new methods for building products in incremental steps. The Helicopter, Knowledge Enabler, Moderator and Networker support the developers in breaking departmental silos to work in cross-functional teams.

We are convinced that in a rather traditional development company a Scrum Master will not become obsolete in a mature team, but is continuously needed to empower a team to work in an agile way. Therefore, the implications of our research are applicable to rather traditional development companies.

5.3 Role Transfer Process

We refer to maturity and suggest that the Scrum Master is located within one dedicated leader in an immature team, but shared between one individual and several team members in a more mature team. Consequently, we suggest that leadership is a fluid concept that evolves over time while the team matures. Yet, similar as the research by Gren and Lindman (2020) we suggest that teams will only take on leadership roles if they are embedded in a supportive organizational environment. Thus, we take the context within which leadership emerges into account.

Our role transfer process reflects several elements of the forming-storming-norming-performing model by Tuckman (1965). The forming phase by Tuckman (1965) 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 developers. This is also described by role conflicts during the storming phase by Tuckman (1965).

Over time agile teams establish a shared understanding of roles and responsibilities which is expressed by the norming phase by Tuckman. Developers 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: developers learn to play parts of the Scrum Master themselves while being allowed to do so by other developers, 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 we have not come across many teams during our research in which leadership roles were truly shared among the Scrum Master and the team members. We have two possible explanations for this finding: firstly, previous research from the literature stream on *self-managed* teams finds that leadership roles vary according to the development stage of a team, and suggests that roles of monitoring and coordinating may be replaced by team norms after a while (Yang 1996). Similarly, we 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. Secondly, earlier empirical observations reveal that the performing phase is seldom reached in organizational context (Marks et al. 2001).

We 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. As found in previous studies (Stray and Sjøberg 2016; Noll et al. 2017) we identified role conflicts. The Scrum Master was torn between meeting demands of rather bureaucratic organisations and of the agile way of working. Those conflicts limited the role transfer from the Scrum Master to the team. Therefore, to enable the team 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.

No matter which choice an organisation makes, it is important for organisations to have a common understanding regarding leadership in agile teams and to communicate clearly what is expected in the specific company setting. Otherwise teams may have wrong expectations of the agile way of working which results in frustration.

5.4 Leadership Gap

Command-and-control behavior by Scrum Masters, Product Owners or management was found to weaken self-organisation of teams (Moe et al. 2010; Hoda 2011). Bäcklander (2019) advices 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 (Moe et al. 2010). We believe that our finding of displaying a leadership gap that permits teams to play leadership roles fits well with those previous empirical findings.

We found that a Scrum Master provides a *leadership gap*. The leadership gap generates hierarchical free space in which developers take on leadership depending on the situation.

The Scrum Master may create it on purpose, e.g. by stepping back, or unintentionally, e.g. by being absent. This leadership gap may be rather small at the beginning, gradually allowing the team to step in. In a more mature team, the leadership gap may be larger. The Scrum Master continuously needs to find a balance between performing the nine leadership roles, and letting the team take on the roles. To determine a suitable size of the leadership gap, the dedicated leader may consider the willingness and capability of a Development Team to take on the leadership roles.

6 Practical Implications

In the previous sections we have elaborated on our research model and our empirical findings on the changing Scrum Master. We now suggest ideas on how practitioners can apply the knowledge in company context.

– **Management**

Many practitioners on the management level have set the agile transformation of their organisations as one of their top priorities. One way to reach this goal is implementing the method Scrum. Often managers believe that when implementing the Scrum framework, developers will do "twice the work in half the time" (Sutherland and Sutherland 2014). 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.

Furthermore, 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 tayloristic past being deeply imprinted in parts of the organisational memory, traditional viewpoints on hierarchy tend to remain and it is therefore easy to hold on to traditional sources of power.

When implementing agile teams in rather traditional development companies, often organisations develop hybrid models: they try new approaches of collaboration while keeping to traditional processes and sources of power, e.g. in terms of reporting, resource allocation or decision making. This leads to even more complexity and higher efforts for alignment and communication. For example, agile teams may have an external review and an internal status report to the team 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, they still cling to traditional decision-making rituals. This leads to de-motivation of Development Teams since they were promised more leadership responsibility when launching agile methods.

We therefore suggest to critically examine the existing structure and processes and agree upon decision-making responsibilities while including the agile team. This will build a shared understanding on leadership. Moreover, we 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.

To empower developers to take on leadership, not only the team environment influences the agile way of working but also the organisational environment. For example, external pressure, top-down changed targets and shifted priorities as well as frequent changes of the team setup destroy the sheltered space within which agile teams can grow. Therefore, organisational culture and structure have to change since this strongly influences expectations and scope of actions of the roles. We 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.

Managers should provide opportunities for agile teams to be actively part of the needed structural change towards a more agile organisation. For example, managers should allow the team to design their own working conditions freely within set boundaries in terms of time or budget. Furthermore, we 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 Development Teams in strategic decisions will lead to even more motivation and commitment to take on leadership roles.

– **Product Owner**

The Product Owner is an important mediator between the customer and the agile 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. When developers truly feel the freedom to take on leadership roles, they will be able to tackle complex challenges as a team. In an agile environment every team member brings skills, expertise, wisdom and knowledge openly to the table. Everyone suggests own ideas and challenges the status quo, such that innovative and complex products can be build together.

However, the Product Owner was found to be torn in between delivery pressure and providing freedom to the agile team. 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.

We found that in rather traditional development organisations it is rather difficult for the Product Owner to bridge the expectations by the management with the expectations and needs of the Scrum Master and the Development Team.

We therefore suggest measurements to diminish the organisational pressure. For example, core circles of representatives of the different agile teams could meet regularly and increase alignment between teams to ensure most effective usage of budget and resources. Also, the Product Owner should believe even more in the power of providing freedom to the Development Team instead of asking for regular status reports. Hence, Product Owners must seek to minimize top-down or external influence to a Development Team during defined sprints to foster the process of taking over responsibility for decisions and committed work products by the team.

– **Development Team**

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. Team members must get familiar with interacting with their team and managers in a different way, and to develop the courage to bridge the leadership gap when provided. Developers need time to try out 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 being a leader, Development Teams need time to learn the leadership roles of Scrum.

Furthermore, the Development Team should occasionally invite stakeholders and managers to the retrospective. This allows them to build a shared understanding of working together, a shared leadership culture and ultimately necessary trust in their abilities and shared responsibilities.

We also found that not aligning responsibilities at all leads to insecurity about roles and responsibilities. Worst case, no one in the team will take on the respective leadership roles which may lead to a leadership vacuum within which no one takes on responsibility. Therefore, we 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.

– **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. 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 developers take on responsibility when they face a lack of leadership.

The Scrum Master steps back and may be needed less over time, and 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 organizational coach and focus even more on solving organizational impediments and aligning diverse 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 Development Team with relevant stakeholders.

7 Limitations and Future Work

In the following we 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* (Glaser and Strauss 2017). The grounded theory explains only the specific context, because the theory emerged while analysing the data from that particular context (Adolph et al. 2008). More specifically, the theory explains the behavior of the people in that particular organisation under study (Adolph et al. 2008). Therefore, our results cannot be generalized.

Constructivist grounded theory assumes that the researchers' perspectives and interaction with the practitioners influences the way data is interpreted (Charmaz 2016). The aim of the research is to get as close to the practitioner's reality as possible. Thereby, providing insights into the motives, believes and intended action of research participants (Charmaz 2016). Therefore, subjectivity cannot be completely eliminated from data analysis and data collection (Charmaz 2016).

Despite these features of grounded theory, the following measures aimed to increase the quality of our study:

To increase **construct validity**, we draw data from different organizations 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 conglomerate, mostly in the automotive industry. To increase **external validity**, we 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, we do not claim our results to be universally applicable and acknowledge that they might be limited to the specific context. Further studies should compare our findings on the changing leadership with data drawn from other companies. For example, researchers could examine if the Protector and Disciplinizer on Equal Terms can also be found in more mature agile companies or if they are rather linked to traditional development companies.

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

Acknowledgements Our sincere gratitude to all those who participated in this research.

This document is the result of the research project funded by the Robert Bosch Automotive Steering GmbH.

Funding Open Access funding enabled and organized by Projekt DEAL. This project is part of a larger PhD project on the Scrum Master role by Simone V. Spiegler and is funded by the Robert Bosch Automotive Steering GmbH.

Availability of Data and Material Due to confidentiality requirements by the Robert Bosch GmbH, the raw data cannot be provided openly.

Declarations

Conflict of Interests Simone V. Spiegler is an industrial PhD student and has received a research grant by the Robert Bosch Automotive Steering GmbH. Christoph Heinecke is an employee at the Robert Bosch GmbH.

References

Adolph S, Hall W, Kruchten P (2008) 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, pp 166–178

Bäcklander G (2019) Doing complexity leadership theory: How agile coaches at spotify practise enabling leadership. Creat Innov Manag 28(1):42–60

Bales RF, Strodtbeck FL (1951) Phases in group problem-solving. J. Abnorm. Soc. Psychol. 46(4):485

Boehm B, Turner R (2005) Management challenges to implementing agile processes in traditional development organizations. IEEE Softw 22(5):30–39

Bryant A, Charmaz K (2007) Grounded theory. SAGE Publications Ltd, London

Carson JB, Tesluk PE, Marrone JA (2007) Shared leadership in teams: An investigation of antecedent conditions and performance. Acad Manag J 50(5):1217–1234

Charmaz K (2016) Shifting the grounds: Constructivist grounded theory methods for the 21st century. In: Morse JM, Stern PN, Corbin J, Bowers B, Charmaz K, Clarke AE (eds) Developing grounded theory: The second generation, vol 3. Routledge, pp 127–193

Cockburn A (2002) Agile software development, vol 177. Addison-Wesley, Boston

Cockburn A, Highsmith J (2001) Agile software development, the people factor. Computer 34(11):131–133

Conboy K (2009) Agility from first principles: Reconstructing the concept of agility in information systems development. Inf Syst Res 20(3):329–354

Edmondson A (1999) Psychological safety and learning behavior in work teams. Adm. Sci. Q. 44(2):350–383

Fiedler FE (1967) A theory of leadership effectiveness. McGraw-Hill, New York

Gersick CJ (1989) Marking time: Predictable transitions in task groups. Acad Manage J 32(2):274–309

Ginnett RC (2019) Crews as groups: Their formation and their leadership. In: Crew resource management. Elsevier, pp 73–102

Glaser BG, Strauss AL (2017) Discovery of grounded theory: Strategies for qualitative research. Routledge

Gren L, Lindman M (2020) What an agile leader does: The group dynamics perspective. In: International conference on agile software development. Springer, pp 178–194

Gren L, Torkar R, Feldt R (2017) Group development and group maturity when building agile teams: A qualitative and quantitative investigation at eight large companies. J Syst Softw 124:104–119

Gren L, Goldman A, Jacobsson C (2019) Agile ways of working: A team maturity perspective. Journal of Software: Evolution and Process

Guzzo RA, Yost PR, Campbell RJ, Shea GP (1993) Potency in groups: Articulating a construct. British J Soc Psycho 32(1):87–106

Hersey P, Blanchard KH, Johnson DE (2007) Management of organizational behavior. Prentice Hall, Upper Saddle River

Hoda R (2011) Self-organizing agile teams: A grounded theory, PhD thesis, Victoria University of Wellington

Hoda R, Noble J, Marshall S (2012) Developing a grounded theory to explain the practices of self-organizing agile teams. Empir Softw Eng 17(6):609–639

Hoda R, Noble J, Marshall S (2013) Self-organizing roles on agile software development teams. IEEE Trans Softw Eng 39(3):422–444

Kozlowski SWJ, Watola DJ, Jensen JM, Kim BH, Botero IC (2009) Developing adaptive teams: A theory of dynamic team leadership. In: Team effectiveness in complex organizations: Cross-disciplinary perspectives and approaches, pp 113–155

Levesque LL, Wilson JM, Wholey DR (2001) Cognitive divergence and shared mental models in software development project teams. J Organ Behav 22(2):135–144

Magpili NC, Pazos P (2018) Self-managing team performance: A systematic review of multilevel input factors. Small Group Res 49(1):3–33

Manz CC, Sims HP Jr (1987) Leading workers to lead themselves: The external leadership of self-managing work teams. Administrative science quarterly 106–129

Marks MA, Mathieu JE, Zaccaro SJ (2001) A temporally based framework and taxonomy of team processes. Acad Manag Rev 26(3):356–376

Moe NB, Dingsøyr T, Dybå T (2009) Overcoming barriers to self-management in software teams. IEEE Software 26(6):20–26

Moe NB, Dingsøyr T, Dybå T (2010) A teamwork model for understanding an agile team: A case study of a scrum project. Inform Softw Technol 52(5):480–491

Moe NB, Aurum A, Dybå T (2012) Challenges of shared decision-making: A multiple case study of agile software development. Inform Softw Technol 54(8):853–865

Nerur S, Mahapatra RK, Mangalaraj G (2005) Challenges of migrating to agile methodologies. Commun ACM 48(5):72–78

Neuman GA, Wright J (1999) Team effectiveness: beyond skills and cognitive ability. J Appl Psychol 84(3):376

Noll J, Razzak MA, Bass JM, Beecham S (2017) A study of the scrum master's role. In: International conference on product-focused software process improvement. Springer, pp 307–323

Parizi RM, Gandomani TJ, Nafchi MZ (2014) Hidden facilitators of agile transition: Agile coaches and agile champions. In: 8th. Malaysian software engineering conference (MySEC). IEEE, pp 246–250
Schwaber K (1997) Scrum development process. In: Business object design and implementation. Springer, pp 117–134
Spiegler SV, Heineck C, Wagner S (2018) Interview Guidelines for "Leadership Gap in Agile Teams: How Teams and Scrum Masters Mature". https://doi.org/10.5281/zenodo.2243113
Spiegler SV, Heinecke C, Wagner S (2019) Leadership gap in agile teams: How teams and scrum masters mature. In: International conference on agile software development. Springer, pp 37–52
Srivastava P, Jain S (2017) A leadership framework for distributed self-organized scrum teams. Team Perform Manag: An Int J 23(5/6):293–314
Stavru S (2014) A critical examination of recent industrial surveys on agile method usage. J Syst Softw 94:87–97
Stray V, Sjøberg DI (2016) The daily stand-up meeting: A grounded theory study. J Syst Softw 114:101–124
Stray VG, Moe NB, Dingsøyr T (2011) Challenges to teamwork: a multiple case study of two agile teams. In: International conference on agile software development. Springer, pp 146–161
Sutherland J, Schwaber K (2017) The Scrum guide: The definitive guide to Scrum: The rules of the game. http://scrumguides.org/
Sutherland J, Sutherland J (2014) Scrum: the art of doing twice the work in half the time. Currency
Takeuchi H, Nonaka I (1986) The new new product development game. Harvard Business Rev 64(1):137–146
Trist EL, Bamforth KW (1951) 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. Human Relations 4(1):3–38
Tuckman BW (1965) Developmental sequence in small groups. Psychol Bull 63(6):384
Wageman R (1997) Critical success factors for creating superb self-managing teams. Organ Dynam 26(1):49–61
Yang O (1996) Shared leadership in self-managed teams: A competing values approach. Total Qual Manag 7(5):521–534

Simone V. Spiegler is a researcher at the Institute of Software Engineering at the University of Stuttgart, Germany, and Scrum Master at the Robert Bosch GmbH. The topic of her doctoral research is leadership in agile teams. She studied sociology, business and software engineering in Mannheim, Madrid, Maastricht and Stuttgart. She conducted several industrial research projects focusing on leadership development, worked at a consultancy firm specialized in agile transformation and is a certified trainer for leadership and organizational development.

Christoph Heinecke is a Director at Robert Bosch Inhouse Consulting. He has profound leadership experience in project management as well as in procurement where he served various multi-functional teams as Agile Master or Product Owner. His research interests include leadership in self-organizing Agile teams and its challenges that arise in traditional industry conglomerates. Christoph holds a PhD in Business and Economics from University of Ulm, Germany, a Diplom-Kaufmann degree from Munich School of Business (LMU), Munich, Germany, and is a certified Scrum Master.

Stefan Wagner is full professor for empirical software engineering and managing director of the Institute of Software Engineering at the University of Stuttgart, Germany. He studied computer science in Augsburg and Edinburgh and holds a doctoral degree from the Technical University in Munich. His main research interests are software quality, requirements engineering, safety & security engineering, service- and microservice-based systems and agile and continuous software engineering. He is a member of ACM, IEEE and GI.