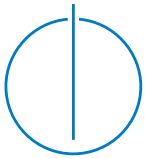
DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Using Multiteam Systems Theory and Team Work Quality to Identify Influence Factors for Measuring the Performance of Agile Teams in Large-Scale Agile Development

Maximilian Doepp



DEPARTMENT OF INFORMATICS

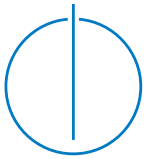
TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Using Multiteam Systems Theory and Team Work Quality to Identify Influence Factors for Measuring the Performance of Agile Teams in Large-Scale Agile Development

Nutzung der Theorie von Multiteam Systems und Team Work Quality zur Identifizierung von Einflussfaktoren für die Bewertung der Leistung Agiler Teams im Large-Scale Agile Development

Author:	Maximilian Doepp
Supervisor:	Prof. Dr. Florian Matthes
Advisor:	Ömer Uludağ M. Sc.
Submission Date:	15.06.2019



I confirm that this master's thesis in informatics is my own work and I have documented all sources and material used.

Munich, 15.06.2019 Maximilian Doepp

Acknowledgments

First and foremost, I would like to thank my thesis advisor Ömer Uludağ for his great support in every way during the whole process of the master thesis. You helped me a lot even before the start of the thesis by finding a working tile until the end and also in between over every stone in the way.

I would also like to thank Professor Dr. Florian Matthes, for providing me with the opportunity to write my thesis at his chair for Software Engineering for Business Information Systems (SEBIS) and his valuable feedback during my first presentation.

Furthermore, I would like to thank Professor Dr. Ing Torgeir Dingsøyr and Finn Olav Bjørnson who always helped with ideas and advice based on their knowledge of TWQ and large-scaled agile development in multiple meetings. Thank you for your time and constant support at every step.

In addition, I would like to thank Sabrina Schweigardt and Christian Schmitz for the close collaboration at the industry partner as well the people working there. Without the chance to participate in the agile transformation at the industry partner this thesis would not be possible. It was a pleasure to investigate the transformation of that scale with such motivated people.

Most importantly, I would like to thank my family, including my better half Katja, who helped with their unconditional and ongoing encouragement. Thank you for your kind support and contribution.

Lastly, I am grateful to all the people from the case study partner who participated in the survey. Only with survey results I was able to analyze the impact on the team and program performance.

Abstract

Minimizing risks through intensive pre-planning, applying the rigid step-by-step processes, and structures has traditionally been common practice for large software development. The biggest influence factor on the overall result for this approach, is a mistake that appears at the very beginning of the chain. A rule of thumb is that fixing one mistake makes it ten times more expensive to fix per step. A missing point in the analysis phase can be quickly added to the original requirements. If this first becomes apparent in the testing phase though, than the entire process must be run again. Furthermore, these projects are very time-consuming and not very flexible considering new requirements from the market.

For this reason, in recent years companies have concentrated more and more on agile development by structuring their organization into smaller units and implementing a team-oriented organization. With more and more successfully finished projects, it became the starting point for larger companies to adopt agile for their needs. However, millions of lines of code cannot be written by only seven developers. This was the point where large-scale agile became relevant to define methods that would, in turn, define light weight processes in order to carefully coordinate such a combination of teams. The new structures bring new challenges and new influence factors that effect the success of the project.

At this point this master thesis starts analyzing the different influence factors on large-scale agile teams and programs. The main goal is to identify models based on the available literature from large-scale agile and other scientific areas concerning the interaction of teams that have the same overall goal. To prove the influence in practice, the influence factors are validated at a case study partner in the banking sector by conducting a survey in a large-scaled agile program with almost 150 team members. The results and findings should provide existing and new projects insights into ways to optimize their performance.

Contents

1 Introduction

This chapter explains why the focus on influence factors for team and program performance is relevant in the current software development period where agile development has become state of the art. Furthermore, it presents the main goals and the research questions (see Section 1.2) that are addressed in the master's thesis and what methodology is applied to answer those research questions (see Section 1.3).

1.1 Motivation

Teamwork and the optimization of people is as old as the history of human cooperation. Sally Bibb, author of "The Stone Age Company" illustrates, that stone age men had to optimize their work (hunting) as opposed to the present-day man who has commitments to other people, tasks, motivation, and the desire to take risks and encourage innovation [8]. The Stone Age man lacked a computer, but today's companies have two challenges in software development that have changed and is still changing the software development process and how it was developed in the last three decades.

Complexity of software

Software has become integral to our lives. Whether it be the software to edit spreadsheets in the office and send them via e-mail, Facebook to check what our friends had for lunch today, or the software in our cars that saves lives in emergency situations, everyone relies on software to deliver its part.
To give an example of how complex software has become, the following comparison shows the increasing number of lines of code throughout history. In 1969, the Apollo 11 flew to the moon and back with 140,000 lines of code [83]. Nowadays, the mobile phone in your pocket will have over twelve million lines of code where you can browse Facebook with fifty million lines in your Chrome browser with ten million lines of code in the car with 100 million lines of code [32]. These numbers of code sound immense. Just as an example, one million lines of code printed out, would cover 18,000 sheets of paper which is the equivalent to fifteen copies of *War and Peace*.
To develop such examples of complex software in the past, one would have required complex organizations and processes to analyze $\rightarrow$ design $\rightarrow$ implement $\rightarrow$ test and maintain it (Royce and Winston (1987) [89]). A great deal of time and energy was invested in the upfront planning and gate processes were introduced so that all eventualities were taken into account to mitigate the risk of failures. But this approach led to a lengthy time-to-market and an inflexible requirements management, which did not allow fast

changes to customer wishes and needs. The introduction of agile development, that is explained in more detail in Section 2.1, emerged as a lightweight development method in the 1990s [6]. More and more people and companies became convinced in the need to transition from the heavyweight approach to agile development.
Agile development emerged from the combination of a limited amount of developers and the cooperation of a small team. Over the years, as more projects were successful, this became the starting point for larger companies to adopt agile for their needs. However, millions of lines of code cannot be written by only seven developers. This was the point where large-scale agile became relevant to define methods that would, in turn, define light weight processes in order to carefully coordinate such a combination of teams.
At this point a new field for software development emerged: the coordination of teams. The old processes were mostly based on a top-down planning approach to coordinate such large-scale settings. Yet, agile development focuses on a strong bottom-up approach that is completely different to most management styles of multiple team coordination. This setup is more similar to the definition of a multiteam system (MTS) from Hoegl and Gemuenden, 2001 (see Section 2.3), which defines a setting of multiple interdependent teams which are combined into one system to fulfill a shared goal [59].
This research area is particularly under developed as only a limited number of studies are available and even less have focused on the performance outcome for an agile program.

Lack of resources

As easy as it may sound: What is needed to develop software? Qualified, available, and affordable developers. However, this is a problem in most European and North-American countries. Based on research from Bitkom in 2018, the lack of IT experts is a pressing problem in Germany. There are more than 82,000 open positions and 81% of the asked participants identify a lack of IT experts [105]. This means that it is nearly impossible to begin a new project with 100 skilled developers in Germany. Each project has to make trade-offs based on the resources that are available. Among the typical solutions are the use of young junior developers and the geographical distribution of the team. Most companies rely on developers in different locations or on freelancers, who usually support the team for an extended period of time. If one does not find enough resources in this way, most companies focus on a nearshoring or offshoring strategy. This strategy increases the coordination effort for the teams, as factors such as language, time zone, or cultural differences have an impact on the team.

Optimization of the outcome

Combining the two factors with time and money, it becomes clear that this is an optimization problem: how does one manage to achieve the best result with the given setup of persons and procedure model?
The goal of this thesis is to answer following question: what factors have an influence on the team and program performance of a given *Scaled Agile Framework* (SAFe, details in

Section 2.2.3) program. The focus is not on which methodology would be better, since there are already a number of studies (Aoyama (1998) [2], Vijayasarathy et al. (2012) [112]) in this area and the agile approach is currently the most widely used. Additionally, the management structures or other predetermined influences are not considered here. The goal is rather to find a model that provides an indication for how each team works together and what the expected performance is. Based on the factors it should be possible to define actions which would have a positive effect on the performance.

1.2 Research objectives

Four questions shall be answered in this thesis in order to achieve the overall objective. The research is divided into team and program levels, because there can be different factors on each level. The questions are validated by the case study partner in order to provide evidence.

> **Research question 1 (RQ1):** What positive and negative influence factors exist on team and program level in literature and which can be mapped to an agile environment?

To answer the first research question, existing literature on challenges and success factors for agile development and large-scaled agile transformations will be analyzed to provide an overview of what factors and models exist in this area. These factors provide an overview of which influences are known so far and therefore it should be clarified which factors have been considered in the model in research question three and which are not.

> **Research question 2 (RQ2):** Can the multi-team system (MTS) concept be applied for large-scale agile programs in order to adapt the existing research in this area?

Question two focuses on influencing factors at program level. Since there are very few studies in the large-scale agile environment in this area, the MTS concept is used here because it is a similar construct. Therefore it is answered whether the MTS studies can also be applied to large-scale agile development or whether there are differences. If the requirements for an MTS are fulfilled by an large-scale agile program, then existing studies from MTS can also be applied to large-scale agile programs.

> **Research question 3 (RQ3):** Can the teamwork quality (TWQ) model be applied by the case study partner on the team level and are there any additional significant factors that can be added?

The third question will be illustrated through a survey at the case study partner that the teamwork model of Hoegl and Gemuenden is applicable here. A survey was carried out and the results compared with a study from Dingsøyr (2018) in an agile environment.

Additionally, further influencing factors were applied to the model, which had an influence on the performance. The (1) team size, (2) geographical distribution, (3) company affiliation and (4) experience are regarded as additional factors, which are confirmed or rejected in extra hypotheses.

> **Research question 4 (RQ4):** Can the TWQ model be applied from the team level to the program level?

Finally, based on the case study, the fourth research question clarifies how the TWQ model can be applied at the program level. A second survey was used to prove this. It is important to note that this is only one program and therefore only an indication is delivered, since several programs are required to verify direct proof.

1.3 Research approach

The thesis is divided into a theoretical part with a literature review and a qualitative case study approach [40] that uses a mixed methods exploratory research design (Creswell and Clark (2011) [27]; Tashakkori and Teddlie (2010) [109]) to combine a literature review, a case study, and a survey into the research approach. Mixing methods can give additional insight which is not possible if only one method is used (Kaplan and Duchon (1988) [65]). This combined approach has already been used and proven in multiple research papers (e.g. Cyr et al. (2009) [29]; Turel and Bart (2014) [110]).
The mixed method approach helped to match the current literature with insights regarding large-scaled agile development and multiteam systems in order to see if the results applied to the case study partner. The case study results are then used and combined with the survey results to analyze correlations between team setup and team performance.

In detail, to following approach was used:
The literature review was made to build the knowledge base for three topics, besides the fundamental explanation of Agile, Scrum and SAFe. The following three literature reviews are used for this theses:

Challenges and success factors for large-scale agile transformations In order to first identify the overall challenges and success factors, a structured literature review is done based on the recommendations of Brocke et al. [113]. Based on the motivation, scope, and research questions, a relevant search term is designed. The following terms and their synonyms and abbreviations: "Title:(Performance OR "Success factor" OR Challenge) AND Title:(Product OR Program OR Group OR Teamwork OR Team) AND ("Software Development" OR "Software Engeneering" OR "Large Scale Agile Development" OR "Extreme Programming" OR "Scrum") AND NOT(Manufacturing)" and later than 2010. Using these search terms in ACM, Google scholar, Web of Science and IEEExplore, 351 initial results of relevant literature were found in November 2018. The results were checked for duplicates

and first filtered by the tile. The remaining were ordered by the number of cities in order to find the most relevant results. The results are shown in Section 2.2.4.

Teamwork Quality (TWQ) Two required results are delivered by a forward and backward search based on the initial TWQ paper. (1) What works already exist that combine both TWQ and agile? (2) What other models are available that investigate team performance, especially with regard to team size and geographical distribution?

Multiteam systems (MTS) Additionally, for the MTS a forward and backward search was conducted to find existing MTS models in this research area. A focus was made to identify models with relevant influence factors and how they are related to the performance.

The single-case study was used to address research questions 3 and 4 with two online surveys. Additionally, semi structured interviews and observations were done to understand the program setup and the team dependencies in order to explain the results and give insights into the program.
More details about the case study design are described in Section 4.1 by following the guidelines from Runeson and Höst [90] and in order to answer the elements of the plan by Robson [88] about the objective, theory, research questions, methods and selection strategy of the case study.

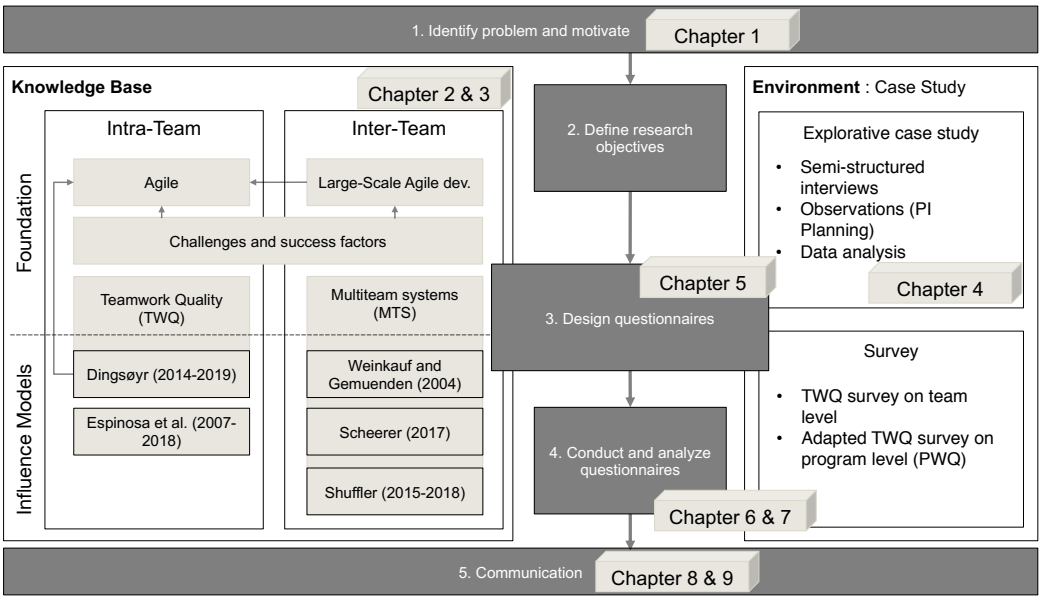


Figure 1.1: Research approach overview and mapping with thesis chapters

Figure 1.1 shows the structure of the thesis by clustering the knowledge base and environment mapping them to the chapters. In Chapter 1 the topic and the motivation of the thesis are defined along with the objectives that should be achieved and how it was done.

Chapter 2 and 3 give an overview of the main knowledge that is required to understand the case study in Chapter 4 and the survey from Chapter 5. Chapter 6 and 7 provide insights about the survey methodology and the results from the two surveys. Chapters 8 and 9 summarize key findings, show limitations and provide an outlook for further research and next steps.

2 Foundations

This chapter provides the theoretical foundation for the following chapters of this thesis and shows how the different theoretical constructs map together. It is important to know not only the fundamentals for scaling agile frameworks (see Section 2.2) and what values and principles are being used for Agile (see Section 2.1)/Scrum (see Section 2.1.2), but also how teams are working together in multiteam systems (see Section 2.4).

2.1 Agile software development

The increasing number of failed projects and adjustments to the requirements in IT projects with traditional software development (e.g. waterfall model defined by Royce in 1970 [89]) in the 1970s to 1990s initiated the transition to agile development. The challenges in a plan-based approach are that the requirements must be clearly defined at the beginning of the project, and that changes, undefined, as well as emerging requirements, cannot be addressed [22].

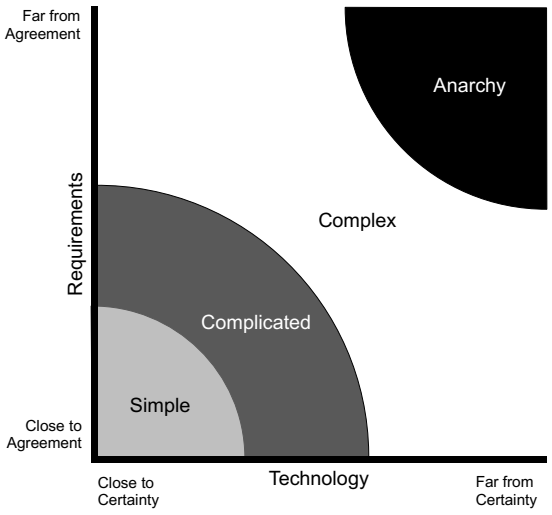


Figure 2.1: Usage of agile development in case of uncertainty with requirements or technology or both [66]

This is due to the growing software complexity over the years as a result of the growing availability of PCs. Schwaber and Beedle prove that agile can be applied when there is

uncertainty with requirements, technology, or both (see Figure 2.1). They call this the "complex space" and include projects such as the development of brand new products or knowledge work [66].

To address these challenges a number of software methods arose that follow an agile approach. Some of the most popular methods are Scrum (Schwaber (2004) [95]), eXtreme Programming XP (Beck and Gamma (2000) [7]), Crystal (Cockburn (2004) [23]) and Feature Driven Design FDD (Coad et al. (1999) [21]).

Compared to traditional software development approaches, agile methods are more adaptive than predictive. They focus on providing value to customers and supporting the variability of requirements (Boehm (20002) [10]). They want to achieve these goals with high customer involvement based on *User Stories*, as well as through customers participating in discussions and therefore giving faster feedback for the development team (Fraser et al. (2004) [47], Lohan et al. (2011) [76]). An additional difference is the people-oriented approach rather than the process-related (Beck et al. (2001) [6]) one. It has been proven that the human factor in agile projects is a critical component for the project's success. Subsequently, the agile methods promote team cohesion and interaction between developers and customers (Ceschi et al. (2005) [19]).

2.1.1 The agile manifesto

Based on the new approach, in 2001 seventeen software developers analyzed the existing solutions and published the *Manifesto for Agile Software Development* (Beck et al. (2001) [6]). In it they combined ideas on how to make software development more agile. This is mainly defined in four values and twelve principles.
The "Manifesto for Agile Software Development" posted on the *Agile Alliance* website [1] contains the following statement:

> "We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value"
>
> 1. Individuals and interactions over processes and tools
> 2. Working software over comprehensive documentation
> 3. Customer collaboration over contract negotiation
> 4. Responding to change over following a plan"

Figure 2.2: The four core values of the agile manifesto [6]

With its core values and twelve core principles, the *Agile Manifesto* provides a theoretical basis for effective software development. However, in order to benefit from the agile

[1] `http://www.agilemanifesto.org`

software development, companies must adapt the core values and principles to their own organization and software development processes.

1. Our highest priority is to satisfy the Customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the Customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity–the art of maximizing the amount of work not done–is essential.
11. The best architectures, requirements, and designs emerge from self organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Figure 2.3: The twelve principles behind the agile manifesto [6]

2.1.2 Scrum

The idea, originally presented at a conference in 1995 (Sutherland and Schwaber), is based on research by Takeuchi and Nonaka (1986) who investigated the development of new products in Japan and the United States. The companies followed a new approach for product development called the rugby approach. Companies using this approach show six characteristics of new product development: (1) built-in instability, (2) self-organizing project teams, (3) overlapping development phases, (4) multilearning, (5) subtle control and (6) organizational transfer of learning [108].

Definition of scrum:

Scrum: A framework in which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value [97].

Scrum is:

1. Lightweight
2. Simple to understand
3. Difficult to master

Scrum is a process framework that has been used to manage work on complex products since the early 1990s. Scrum is not a process, technique, or definitive method. It is, rather, a framework in which one can employ various processes and techniques. Scrum highlights the relative efficacy of one's product management and work techniques so that one can continuously improve the product, the team, and the working environment.
The Scrum framework consists of Scrum teams and their associated roles, events, artifacts, and rules. Each component within the framework serves a specific purpose and is essential to Scrum's success and usage.
The rules of Scrum bind together the roles, events, and artifacts, governing the relationships and interaction between them. The rules of Scrum are described throughout the body of this document.
Specific tactics for using the Scrum framework vary and are described in other scientific works.
The Scrum process is iterative and incremental. A visual representation of the flow can be seen in Figure 2.4. The *Scrum Team* consists of the *Product Owner* (PO), the development team (dev. team), and the *Scrum Master* (SM). They are self-organizing and cross-functional and the teams have all the skills needed to execute the job without being dependent on others who are not part of the team. The Product Owner is responsible for maximizing the value of the product resulting from the work of the development team. The approach can vary significantly between companies, Scrum teams and individuals. The *Product Backlog* is an ordered list of all things the product is currently required to do; it is the central point of contact for all change requirements to the product. The *Product Owner* is responsible for the *Product Backlog*, including its content, availability and ordering. The expectations for the team are discussed in the *Sprint Planning* meeting between the PO and the dev. team members. The scope that was part of the *Product Backlog* will than be the *Sprint Backlog* for the *Sprint*. The core of Scrum is a *Sprint*. A standard time line for a *Sprint* is between two weeks and one month. All *Stories* from the *Sprint Backlog* that are completely finished become a release-ready *Product Increment* (PI). *Sprints* have a constant duration throughout the development time that should not be adapted too often. A new *Sprint* begins immediately after the previous *Sprint* has been completed. The *Sprint* results are shown to the *Product Owner* by the *Development*

Team members. The *Product Owner* decides if all *Acceptance Criteria* have been fulfilled so that the *Story* is "Done". All *Stories* that are not "Done" will be moved back to the *Product Backlog* and are potential candidates for the next *Sprint Planning* meeting. The *Development Team* consists of seven plus minus two people. They work cross-functionally and are self-organizing. The *Scrum Master* is responsible for promoting and supporting Scrum. The SM help everyone understand the Scrum theory, practices, rules, and values. The *Scrum Master* (SM) helps the Team to archive goals and remove existing impediments. Two additional meetings (*Daily Scrum* or *Daily Stand-Up*) are used to make the progress more transparent to all development team members and to optimize the team and the Scrum process (*Sprint Retrospective*) (Schwaber and Sutherland (2011) [96]), [97]).

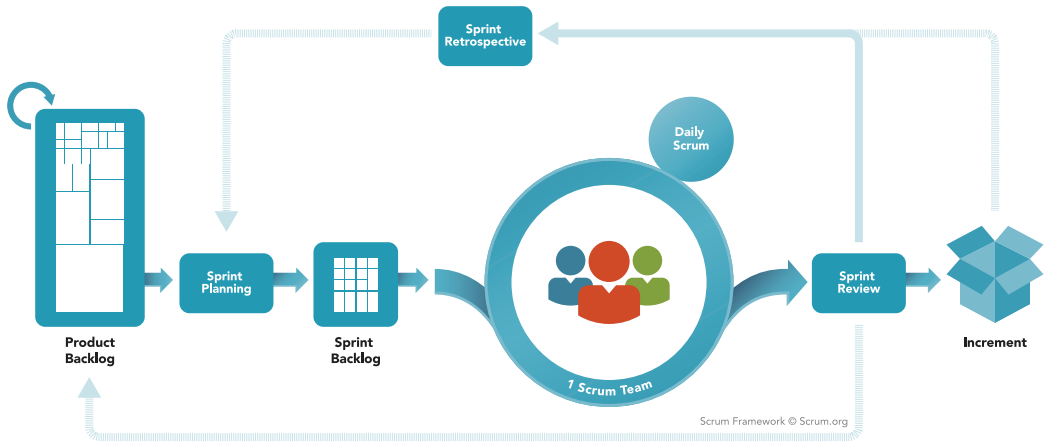


Figure 2.4: The Scrum Framework [99]

2.2 Large-scale agile software development

Originally agile methods were designed for single-team projects [11], but the success and the easy structure of agile methods made them appealing for larger projects with more than ten people [37]. It is worthwhile to note that one should not keep filling an agile team with more and more team members, because then the Ringelmann effect becomes bigger. This effect demonstrates that people become less productive when they work with others; this loss in efficiency increases with group size, but with gradually decreasing speed [86]. The effect is shown in Figure 2.5 that shows the effect investigated by Steiner (2007) who determined that groups never perform at the productivity level they could, because of process losses. Steiner (2007) summarizes all influencing factors that occur when people work together as process losses. Such examples are coordination, communication, challenges, and motivation issues. The peak productivity of a team is four to five people according to Figure 2.5 and this is the same result that was investigated by Hackman (2002) [54]. Therefore, in a large project, there is no alternative than to divide

the team members into several teams in order to keep the process losses as low as possible. However, a compromise has to be found between team size and overhead per team, as each team needs a PO and SM. Additionally the coordination between a few teams is easier than between many teams, therefore the recommended team size of Scrum is slightly larger than the results of Steiner (2007) and Hackman (2002).

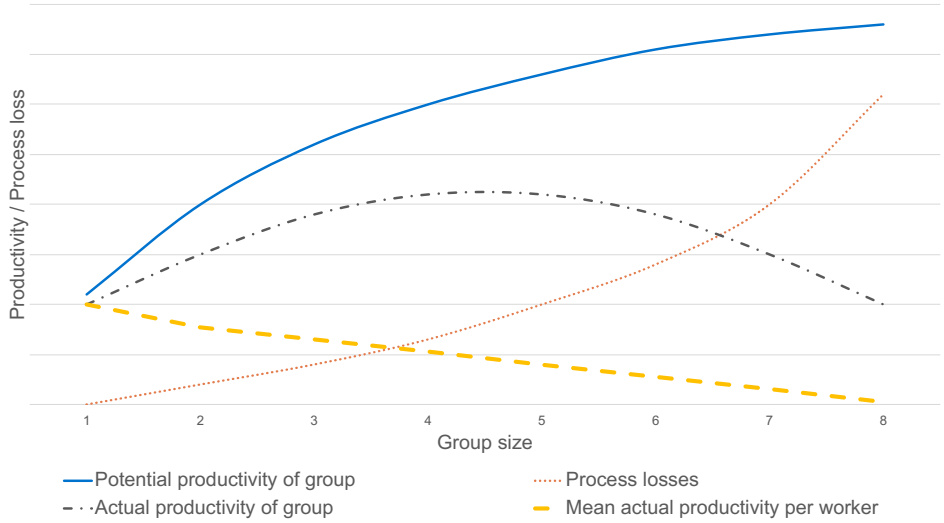


Figure 2.5: Productivity development in correlation to the team size [106]

However, it should be noted that larger projects require additional coordination especially regarding the inter-team coordination, architectural guidance, and clearly defined requirements outside of the team. Naturally, the additional effort brings advantages, such as shorter time-to-market for new features and higher velocity [71]. Scaled frameworks offer a solution for exactly this problem in order to use Scrum as a method on a large-scale and for a large number of product teams. Companies can use different scaling frameworks. These frameworks are usually developed on Scrum or other agile methods as a foundation. This definition shows what must be taken into account by scaling agile:

> "There are two fundamental visions on what it means to scale agile. The first, tailoring agile solution delivery strategies to address scaling factors such as geographic distribution, regulatory compliance, and large team size is referred to as tactical scaling. The second, adopting agility across your IT department and your organization as a whole, is referred to as strategic scaling. The good news is that many organizations are tactically applying agile techniques at scale and are succeeding in doing so." [1]

At the XP2014 conference, Dingsøyr and Moer and other participants determined what defines large-scale agile development. The main criteria are size, code base size, project

duration and project budget; e.g., "Over 50 developers OR 1/2 million lines of code OR more than 3 sites / time zones" [36]. Dingsøyr and Moer defined it more precisely in 2018 with the following definition: it is large-scale development when more than two teams work together and a very large-scale development when more than ten teams work together [37].

Based on the growing demand, more and more scaling agile frameworks have appeared over time, among them are: Spotify Model, Scaled Agile Framework (SAFe), Disciplined Agile 2.0, Scrum of Scrums or Nexus. Uludağ et al. compared twenty scaling agile frameworks in their paper "Investigating the role of architects in scaling agile frameworks" [111]. *SAFe Essential* will be described in more detail in Section 2.2.3. Besides SAFe, LeSS, and Nexus are the two most common frameworks that could be compared to SAFe in order to see the differences in framework.

2.2.1 Large Scale Scrum

The framework Large Scale Scrum (LeSS) is following the motto "LeSS is more". It stands out due to its simple structure, as it is based on functioning Scrum teams. These teams are responsible for agile product development under the control of a single PO. The LeSS framework is characterized by three basic elements: Principles, Rules and Experiments. It is an organizational framework that enables Scrum teams to work together as effectively as possible. Figure 2.6 shows an overview of the process with a combined *Sprint Planning 1* where the product backlog is defined and than the teams split up and define a *Sprint Backlog* for each team. The self-organized coordination between the teams during the *Sprint* is also a part of LeSS and can be done either independently or with a *Scrum of Scrums* meeting. The final result is presented as a combined effort by all teams in the *Sprint Review*.

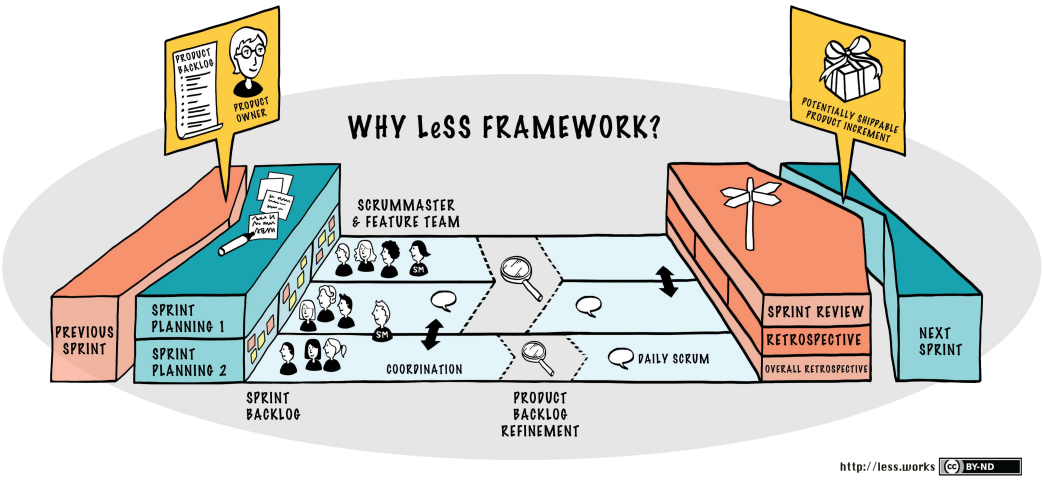


Figure 2.6: LeSS Framework Overview [15]

The comparatively low implementation costs and scaling possibilities of POs are certainly

advantageous. On the other hand, there are certain disadvantages in comparison to other scaling frameworks. Integration of LeSS into larger organizations is a difficult and radical approach. Furthermore, scaling with Scrum teams is limited to a certain size and is primarily suitable for medium-sized companies. An extension to LeSS is called *LeSS Huge* and it allows the opportunity to include four to eight teams each working under an area PO.

2.2.2 Nexus

The third most common scaling framework is Nexus, which is also an extension of Scrum. There are a maximum of nine Scrum teams working in parallel and only one Nexus integration team is integrated as an additional team. The main task of this team is to ensure a universal understanding and use of the necessary practices and tools in the Scrum teams. The clear advantage of Nexus over SAFe and LeSS is the minimalist structure and low level of formality of Nexus. On the other hand, this means that a high amount of personal contribution is required for the concrete design and transferal into business practice. Compared to the other two industry leaders, Nexus is much less widespread in practice. Although Nexus, like LeSS, does not provide the company with any information on how the agile processes relate to strategy processes, portfolio management or operations, applying Nexus makes it easier to scale from Scrum.

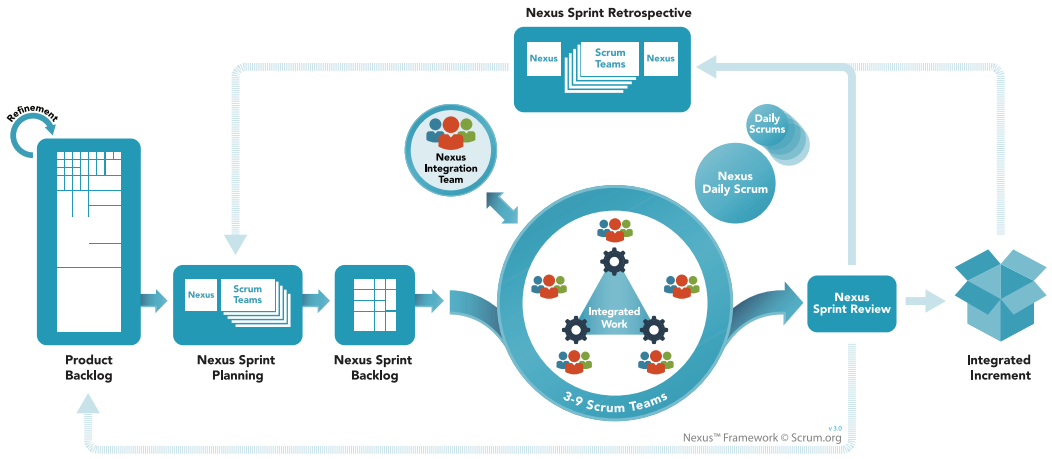


Figure 2.7: Nexus framework poster [98]

2.2.3 Scaled Agile Framework

The *Scaled Agile Framework* (SAFe) is based on the VersionOne 12th *Annual State of Agile Report* [62], the most popular scaling method by 29% of the respondents, as compared to *Scrum of Scrum* (19%), and with internally developed frameworks (10%). It was first presented and then continuously developed from version 1.0 in 2011, to the current version 4.6 by Dean Leffingwell (2018) [91]. SAFe is preferred especially by large companies that

have adapted large-scale agile for their companies [92]. The reasons for this preference is that SAFe extends the Scrum (see Section 2.1.2) idea with XP and adds existing roles in larger companies like architects or legal departments. Additionally, SAFe has four types of adoptions that can be used (Essential, Large Solution, Portfolio, Full) with very detailed documentation and training which makes the implementation easier. Essential SAFe will be explained later in more detail, based on the use of the case study industry partner.

Essential SAFe

Essential SAFe is the basis for all other SAFe configurations. Figure 2.8 shows the overall picture of the Essential SAFe process with the foundation layer, the team level and the program level. Together the team and program levels make up the *Agile Release Train* (ART). The program level (upper part) focuses on the high-level *Backlog Items* (Epics) from the portfolio level. Based on the *Vision* from the program level, the goals for the team level and the next releases are planned as a road map. The Team level (bottom part) consists of multiple teams that work together in *Sprints* to fulfill *Stories* like in Scrum (see Section 2.1.2) [69].

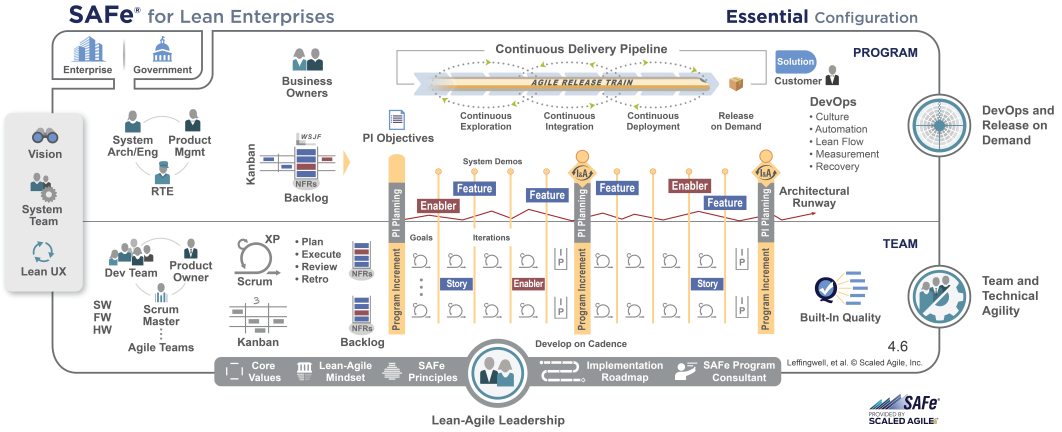


Figure 2.8: Essential SAFe Overview Version 4.6 [93]

SAFe sets the playground for multiple teams to work together by defining the roles, events and artifacts on both team and program levels. It has additional highlights as well. These can be understood as setup, definitions or preconditions which are necessary for SAFe. The following list gives insights and details concerning the fundamental knowledge of SAFe. The relevant details that must be understood about the roles, events and artifacts, as well as how they act together in order to build an scaled agile framework, which is the final set of rules of the game for all participants in the program.

Important SAFe features

The SAFe features on the team and program levels are a combination of definitions that are important for SAFe. For example, the use of integration and program increment and how they are connected with each other. The highlights also describe common IT techniques, such as build-in quality or DevOps, that help optimize the release train. SAFe defines ScrumXP as a combination of Scrum and the XP framework that fit into the SAFe overall process.

Team Level

Iterations is a defined timebox where the program will deliver features and components as new functionality with a value.

Program Increments (PIs) is a fixed timebox for all the teams on the ART to synchronize all teams and set boundaries.

Develop on Cadence is a key method for managing the inherent variability of system development in a flow-based system by ensuring that important events and activities take place on a regular schedule.

ScrumXP is a combination of a foundation of the Scrum framework for the project management with XP framework for software engineering.

Team Kanban can be used either as an alternative or in combination with ScrumXP. It should visualize the workload and the limits of the tasks in the team based on Kanban and the lean principles.

Built-In Quality should ensure that the product has been produced according to the high standard and is maintainable and ready to be used in the future. The assurance that the product can be adapted and developed over multiple years without many instances of re-factoring is a very important product feature. Otherwise the velocity for the future PIs will be decreased if the number of BUGs is high or the code must be completely changed for each new feature. This can be achieved with good software architecture, pair programming, *Unit Test* or *Test-Driven-Development* (TDD).

Program Level

Agile Teams is a program level feature that implies that each program should consist of five to twelve teams resulting in roughly 50 to 125+ people. The responsibilities, tasks, and the roles of the team members are described in the role definition on team level.

Program Increment (PI) is, as a rule, an 8-12 week *timebox* where the ART team develops, in typically five iterations, new features according to the *Program Backlog* and the *PI Planning* meeting.

Continuous Delivery Pipeline should ensure that the time to market for new features is as short as possible for all features, which are combined by overlapping the required steps that a feature should pass. Each feature should move from the continuous exploration process (Funnel → Analyzing → Backlog) to the continuous integration process (Backlog → Implementing → Validation on staging) to the continuous deployment process (Validating on staging → Deploying to production) to the final release product. This process can be visualized with the Kanban board as shown in Figure 2.9. These steps should be done in parallel for multiple features by separating the features as precisely as possible.

DevOps is an idea according to which, unlike the standard practice where the developers are separated from each other, the developers work closely together enabling the whole *Release Train* to work faster. According to the "The DevOps Handbook" the deployment can be done 30 times more frequently and they have 60 times less failures compared to traditional operation structures [68].

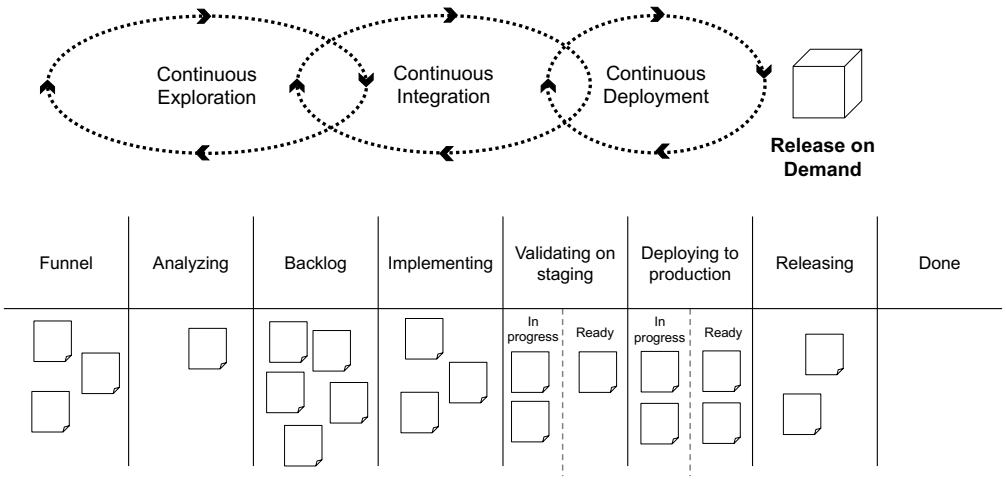


Figure 2.9: SAFe continuous delivery pipeline combined with a program Kanban [93]

Roles

The roles on the team level are similar to Scrum (see Section 2.1.2), whereas on the program level new roles are defined to manage the dependency overhead and other management tasks which are supposed to give guidance and direction to the Team level. Additionally, these roles fulfill the agile manifest principles (see Section 2.3) so that the ART teams are self-managing and self-organizing thus being able to plan and work together.

Team Level

Development Team (Dev Team) is an agile cross-functional development team that consists of 3 to 9 (plus PO and SM) people and include developers, testers, and other specialists.

Product Owner (PO) is responsible for the *Backlog* of the team regarding the *Stories* and the priority. At the end of the *Sprint* the PO will approve the *Stories* based on the acceptance criteria.

Scrum Master (SM) is the coach of the team and is responsible for promoting and supporting the team. They help the team to archive the goals and to remove existing impediments.

Program Level

Product Management is responsible for the *Program Backlog* by defining and understanding the customer needs. The management ensures communication between the customer and the POs of the teams and partially checks the results.

System Architect/Engineer might, depending on the project, be represented by a cross-discipline team or one person, who is responsible for the overview of the overall system architecture by supporting or defining interfaces, or defining what frameworks should be used. Their main attention is dedicated to the functionality of the system as a whole.

Release Train Engineer (RTE) , in other words, the *Chief Scrum Master* for all the teams, supports the teams and the SM by providing workshops or reports about the progress of the program.

Business Owners represents the stakeholders who are responsible for the management part of the project. During the PI they will present the *Vision*, as well as participate in the *Management Review*, *Solution Demo*, and *Post-PI Planning* events by providing feedback on the current solution.

Events

The events for the Team level are internal team meetings organized to plan and work on the current iteration or meetings with the ART to synchronize and coordinate with the other teams.

Team Level

Iteration Planning is a type of meeting where the PO and the dev. team decide what *Stories* can be completed during the next iteration. It is the equivalent of the *Sprint Planning Meeting* from Scrum with the difference that some of the planning can also

be completed during the *PI Planning* meeting. The SM must attend the meeting to ensure the smooth and productive discussion; other stakeholders may attend as well if they are interested in some of the Stories under discussion.

Iteration Review is the review that, together with the archived *Stories*, is being presented by the Dev team to the PO and other stakeholders at the end of the iteration progress in order to receive feedback concerning the results. This is the equivalent of the *Sprint Review* from Scrum. Based on the status and the done *Stories*, the SM can plan the upcoming *Iteration Planning* meeting.

Iteration Execution is the process of the agile team's work and their tracking of the progress during the iteration with the goal to fulfill the required *Stories*. A widely used technique is the *Daily Stand-up* (DSU). In the Daily Stand-Up (DSU) each team member will answer three simple questions: (1) What did you do yesterday? (2) What will you do today? (3) Are there any impediments in your way? [103]. No detailed explanation is required and the SM is responsible for resolving the impediments as quickly as possible.

Iteration Retrospective is the manifestation of the 12th principle of the agile maifesto (see Figure 2.3) which claims that the team should reflect on how they can become more effective. To fulfill this the dev. team will meet at the end of the iteration and discuss the results and try to identify possible improvements. Quite a few ideas on how to organize a *Retrospective* can be found in "Glad, Sad, Mad" and "The 4 L's (Liked, Learned, Lacked, Longed for)" [72].

Backlog refinement is the meeting that should be done once or twice in an iteration to help the *Scrum Master* prepare and estimate stories for the upcoming *Iteration Planning*. This is facilitated by talking trough *Story* details and whether they are clear and ready to be implemented, or if they have preconditions. This will help optimize the *Iteration Planning* and avoid points or questions raised in the *Iteration Planning* meeting that could make it impossible to plan the coming *Sprint*.

Innovation and Planning (IP) Iteration is usually done after four iterations where *Features* are being developed to complete the *Program Increment*. In this iteration the teams are allowed to work on innovative features or solve complex problems. This integration is also used to make final system checks or integration tasks, and to verify or to write the documentation and other pending tasks remaining from the previous four iterations. This period will help improve the team spirit.

Program Level

PI Planning is the core event dedicated to setting the stage for the teams on the ART mission. All team members meet face-to-face during a two day period of time and make a rough plan for the next iterations for the whole increment. On day one, the event starts with the *Product Managements* presenting a short overview of the

current status, the customer's needs, and the *Vision* with the top features for the next increment. Then the *Architecture Vision* is refreshed or explained in more detail to all participants. After the RTE explain the planning context, each team starts with the first *Team Breakout* which is used to roughly plan the whole increment by checking the team's capacity and what *Stories* can be done in what iteration. During this planning the teams must talk to the other teams to find dependencies between teams and how these would effect the time-line. To make this more transparent, a *Program Board* is used to show the iterations and the teams an the axes, as well as the Stories with dependent *Stories* connected to another team with a red string shown in Figure 2.10. After the *Breakout* each team presents the planning outputs, dependencies, objectives, and risks. This helps the stakeholders and other teams see a clear image of the last agenda item of the day. In the *Management Review* and *Problem-Solving Meeting* the mentioned feedback is taken into account and solutions are developed. On the second day, the managers present potential changes in the planning, resources, or scope from the last *Management Review* to all team members. This makes it clear for the second team breakout whether the plan will be detailed or redefined. At the end of the second team breakout the Business Owners give feedback about the business value for each team and team goal. After the planning is done, the teams present the final plan to the audience. The risks are discussed separately after the *Review* to make them transparent and to check if they are resolved, owned, accepted, or mitigated. After the program risks have been addressed, a *Confidential Vote* is organized for all team members in order to check how everyone feels about the current plan. Based on the vote's results, the plan must be modified until a higher confidence level is reached. Once this has been reached, the RTE leads finalize the PI by analyzing the discussed agenda of the current PI and making points for improvement for the next one. All the results will than be used by each team and the SM for their planing of the upcoming four iterations.

System Demo is used to present the current fully integrated system to the ART stakeholders to present an overview about the overall progress of the program. It forces the teams to integrate the whole program to one artifact. This will mitigate problems at the end of the PI.

Inspect & Adapt is a type of meeting combining three other meetings in one at the end of each PI. First a *PI System Demo* is presented to show the current status, then a quantitative measurement is presented concerning the progress of the program and each team. In the end the *Retrospective* and problem-solving workshops are organized to help improve the program itself by identifying the root cause of problems.

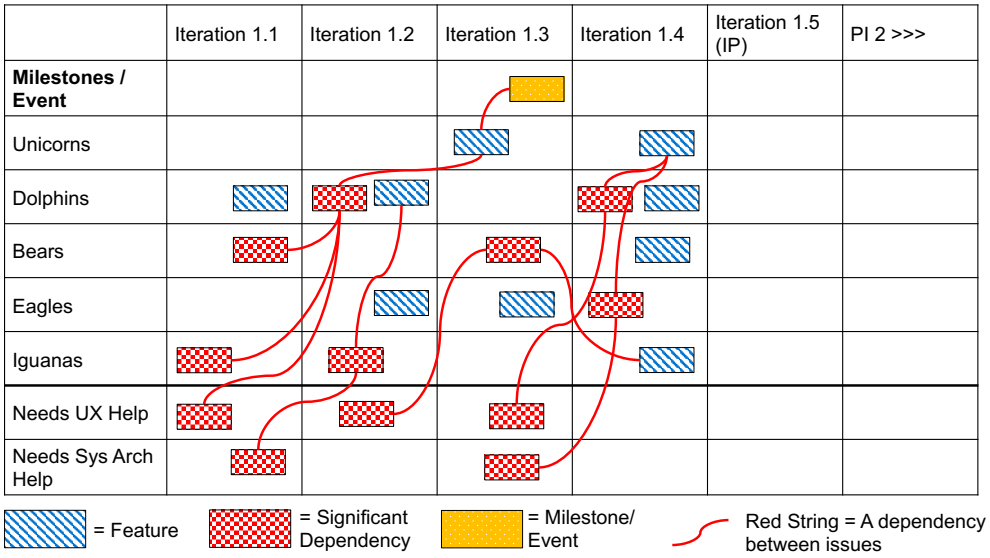


Figure 2.10: SAFe program board [93]

Artifacts

In archaeology, the term 'artifact' refers to an object/thing that was made by humans. In agile development it is something that was used to develop products, track progress, and make this transparent.

Team Level

Story is a description of a functionality from a user's perspective and written in their language. It should make the requirement and the value transparent to everyone involved. Good stories are (1) independent, (2) negotiable, (3) valuable, (4) estimable, (5) small and (6) testable [70]. Detailed information is given in the acceptance criteria for each Story to define what should be implemented and which criteria will allow a PO to accept the *Story* in the *Iteration Review*. They should be as simple as possible, but also very detailed as they are required to fulfill the 2nd agile manifesto value (see Section 2.1.1).

Enabler Stories . Not every task has an end user benefit that is required for a *Story* like architecture or infrastructure tasks. To manage this in the *Team Backlog* the team can use *Enabler Stories*. They can be described in technical terms, but they are handled similarly to "normal" *User Stories*. They are the equivalent to re-factoring and *Spikes* from XP.

Iteration goals are defined by the team in *PI Planning* and are business and technical goals. They are used by the dev. team and the PO to have a mission and to show

other teams the goals by cross-team dependencies.

Team backlog is an ordered list of *User Stories* and *Enabler Stories* managed by the PO. The *Stories* are defined in the *PI Planning* or by the PO and discussed in the *Backlog Refinement Meeting*. In the *Iteration Planning Meeting* the *Backlog* is used to select *Stories* for the next iteration. Figure 2.11 shows the three primary factors that have an impact on the *Team Backlog*.

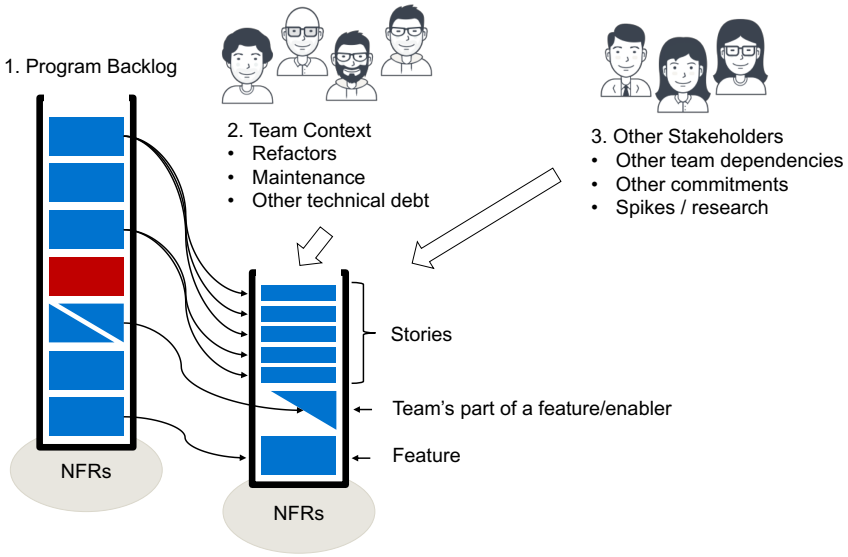


Figure 2.11: SAFe team backlog [93]

Program Level

Features describe an assumption made with the goal of meeting the needs of stakeholders. To make this understandable, each feature is given a name, one or more benefits to fulfill, and acceptance criteria. They should be short enough that they can be completed within one PI.

Program Backlog is the equivalent of the *Team Backlog* with the difference that *Features* and not *Stories* constitute the program and the content.

PI Objectives are the aggregation of all the goals from the agile teams and the train to be achieved in the upcoming PI.

Architectural Runway is the technique and the code that is used to implement the next *Features*, without having to completely redesign the software which causes delays. It is a combination of code, frameworks, and infrastructure that is required for the implementation. It works hand in hand with the *Continuous Delivery Pipeline*.

Metrics

One important component in order to have a working program is not directly shown in Figure 2.8. The metric is part of the "bigger" SAFe models (>= Large Solution SAFe). But it is indirectly included in some of the meetings or it is a required tool for the SM, PO and *Product Management*. It is the metric of each team and the program overall. The metrics should give indications and KPIs as a tool, or clue, to "manage". Here are some reasons why metrics are relevant in software development [81]:

- To make decisions in business
- To challenge team members
- Team members can be proud of the results
- To determine success
- The Agile system cannot easily be eliminated
- It increases satisfaction
- It helps to change the behaviour of teammates
- It supports the decision making process

It is, of course, known that metrics are also used in traditional software development where most of the metrics can also be adapted for agile development. They can be grouped into the following metric classes [81] : (1) product metrics: e.g., lines of code, quality metrics (defects, reliability, maintainability); (2) process metrics: e.g., development time, developer level of experience; (3) objective metrics: values obtained by observers; (4) subjective metrics: personal feelings of different individuals about a metric; (5) resource metrics: e.g., effort for one task, used computer resources; (6) project metrics: e.g., project size, effort, resources, budget, time-line; (7) direct metrics: e.g., duration of testing; (8) indirect metrics: e.g., productivity, requirement stability. All of these metrics could also be used for agile development. In addition, some new metrics have emerged regarding agile development based on the structuring in teams, *Sprints*, and *Stories*. The most common metric is the velocity that shows the amount of work done during the *Sprint* (usually, in *Story Points*). This is very important for the PO and the team for the next *Sprint's Planning*. Knowing the velocity, the number of available working days in a *Sprint*, and the planned *Story* offers a clear possibility for the team's optimization. For large-scale agile development the metrics become even more important, because of the number of people and the size of the project, it is mandatory to track the progress, as well as all the other influence factors that are relevant for a solid software. SAFe described thirty metrics for managing the program on the website, which are grouped into five clusters: (1) enterprise, (2) portfolio, (3) program, (4) large solution and (5) team [104]. Most of these clusters focus on a number of tasks, *Stories*, *Features*, bugs, ... done, as compared to the amount of effort or the timeline. Others focus on the implementation of lean, agile, operational, (DevOps) or SAFe principals. Both of the above mentioned metric

approaches are very important for the management and control of a project as well as for the reporting to the stakeholders. Figure 2.12 grouped the criteria in four clusters based on *Lean Portfolio Management*. These clusters could be used based on other metrics to determine whether the program is in balance and prevent an outcome where one reaches extreme efficiency, but fails in quality.

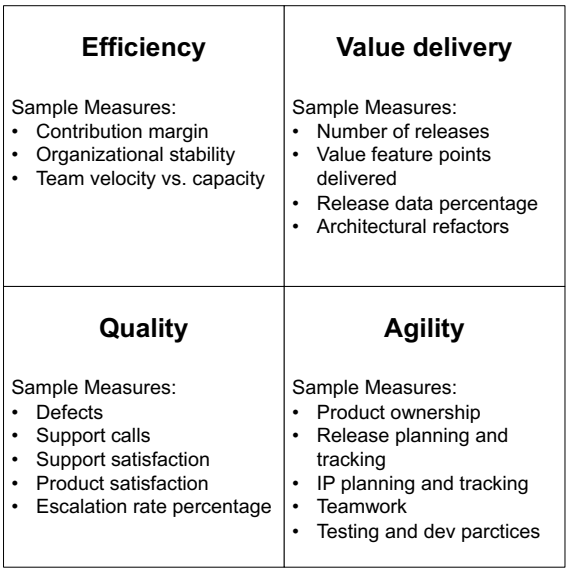


Figure 2.12: SAFe: balanced scorecard [104]

Unfortunately, the metrics of SAFe do not pay much attention to such important factor as the team spirit or the personal success, which might result in a negative outcome for the team. Once a team member ceases feeling happy in their work environment, they gradually become less efficient and their eventual withdrawal from the project will have a large impact on the team and program. It might take up to three months to get a new team member up to speed.

2.2.4 Challenges and success and factors

Understanding the challenges and success factors, especially for smaller teams, is essential for the transition or the implementation of large-scale agile development. A poorly performing team or a failed project is unfortunate for smaller teams, but the word "large" indicates that many more people are involved which means that a team performance of 5% less than it could be is very likely to bring larger financial losses. Imagine the team consists of 100 people with, on average, 100.000 Euro costs per year, than the 5% will be half a million Euros a year. By identifying challenges and success factors and taking into account possible risks, one can reduce the latter in advance.
Using the *Critical Success Factor* approach would be helpful in identifying success factors. According to Bullen and Rockart (1981): “Critical Success Factors (CSF) are the limited

number of areas in which satisfactory results will ensure successful competitive performance for the individual, department or organization." [14]. This means that success factors are the key areas in which a project must succeed to ensure the achievement of the goals.
In literature research on challenges and successes in agile development, most research is carried out by taking into account the change of working method from traditional development to agile development, therefore these two different research directions must be separated. Interestingly, some of the papers focus on the factors for agile development only, whereas others investigate the large-scale agile development.

Solid on-side setup

Cockburn (2001) defined five factors for agile development which will ensure that the project or program will be a success. The first factor is to have two to eight people in one room working and communicating together, which should facilitate a fast information exchange and direct feedback. The second factor is closely related to the first one and is to have on-site experts enabling direct communication. A third point he proposes is a one-month increment as the ideal time frame for the development process with development, testing, and deployment. The next point is related to the fast release train with a fully automated regression test to have a fast development process with a solid quality. The last is to have experienced developers on the team [24].

A project is more than an agile approach

Chow and Cao (2008) grouped the challenges and failure factors into five categories: organizational, people, process, technical and project, as shown in Figure 2.13. What becomes visible here is that the agile and scaled agile frameworks focus on only a part of the influence factors that have direct impact to the success. For example, a project needs more than a SAFe process/program covers [20].

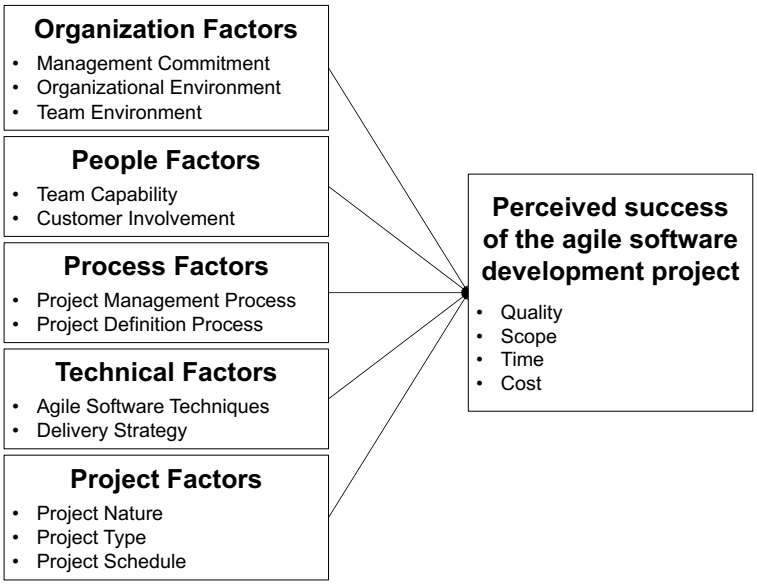


Figure 2.13: Research model for success factors in agile development [20]

The factors have a tremendous impact on the project success in terms of quality, scope, time, and costs. They are used to define the success. Personal success factors e.g. the leanings or the team member satisfaction, are not part of his model. What is missing in the model is how big the impact is depending on the factors and how they could be optimized to increase the outcome. However, the model allows new programs to check if most of the relevant factors are taken into account.

People background and mindset

Cohen, Lindvall and Costa (2004) focus on the team as a whole and the individuals in the team and determined three success factors. They determined that culture, people and communication are the key values of success. Each team consists of team members and if they have the same cultural background and the same mindset this will help to avoid misunderstandings. These factors are also important for closer contact with the customer [26].

Correct splitting of team

In addition to the success factors by Dikert, Kim & Paasivaara (2016), Chow & Cao (2008), and the focus of the background and mindset by Cohen, Lindvall & Costa (2004), and the team setup by Cockburn (2001), there are even more factors that have an impact. One example is the approach by Maurya (2018) to classify the teams in four groups to reduce the complexity of dependencies and make the whole process faster [80]. To achieve this he classified the teams into 4 types with specific characteristics and responsibilities.

Figure 2.14 shows the interaction between the teams and how they work hand in hand to create a final product. To result in less conflicts and to reduce the coordination effort, he works with boundary control so that teams have independent components on which they can work. The first teams are the feature team that represents basically a base Scrum team with the focus on customer needs and to deliver end-to-end features. However, the team has a lower focus on building backend services and flexible architecture, instead they consume existing services. These services are the responsibility of the component team. They build the core service stacks with a modern architecture and the minimum number of dependencies that is maintainable and extendable. These two teams together should be able to build a system. In addition, the setup is extended by two more teams, which normally also belong to the tasks of a Scrum team. As a result of the division, the responsibilities are clear and the experts are not distributed among several teams, even if this contradicts the vertical approach of Scrum. The third team is the specialized team that should be smaller with two to five experts who can help with complex tasks that require a more developed knowledge in certain areas like Oracle DBAs, data security or *Center of Excellence* (COE). Besides the support, they will provide training, setup policies and procedures, or monitor the application. The last team is the support team, as the name suggests the team takes care of the support processes of the application. The team is optimized to analyze and locate incidents raised by the customers and stakeholders. To fulfill the task it is important to understand the complete functionality of the product. After a problem is found, the team will coordinate the fix process with the component and feature teams and track the progress. This has two advantages that effect a agile program: (1) the entrance point for problems are clearly defined and (2) the effort and the first analysis will not effect an ongoing sprint.

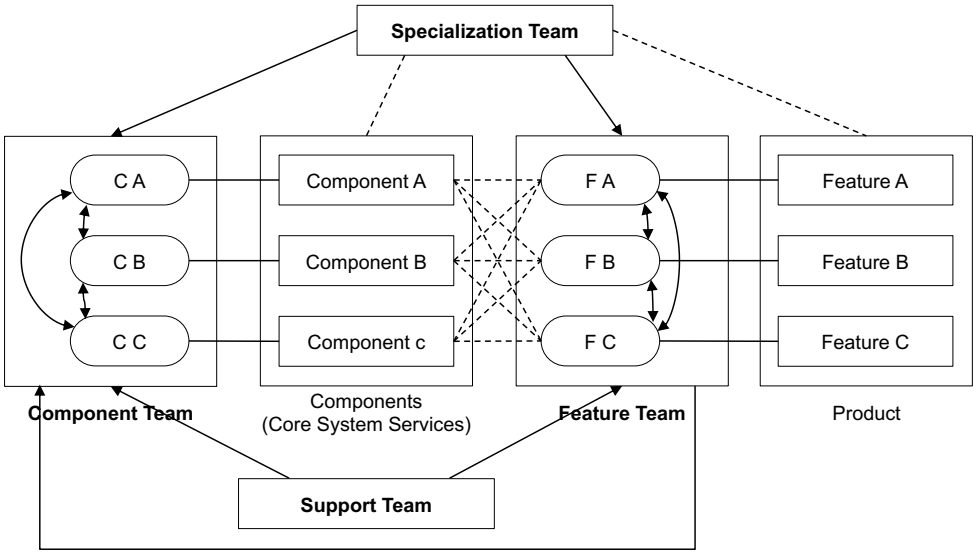


Figure 2.14: Team classification and collaboration process [80]

This approach tries to define core areas for teams so that they can focus on certain tasks while reducing overlap. Each team has a clear boundary area with defined characteristics and a procedure of how to work together, thus reducing the complexity for a program which only results in benefits.

Overview of common success factors and challenges

Besides the success and challenges that are mentioned in this section, that illustrates the variety of factors, there are hundreds of papers and publications available regarding this topic for Agile, Scrum, or scaled agile development on Sprinter, Google scholar and IEEE. As an overview, without mentioning all of them, Dikert, Kim and Paasivaara (2016) produced a systematic literature review of fifty-two papers that describe fort-two different organizations in order to determine the challenges and success factors for large-scale agile transformations [35].
Difficulties with the implementation and the integration of the non-developmental functions based on the number of cases were mentioned among the most success-challenging factors.

Change resistance was one of the most common mentioned challenge with a total of 38% . This is a very general point, but in many project and companies the general resistance to change and the skepticism towards the new way of working is a problem for many people. The top down mandate creates resistance that filters down to the team. If even the management is unwilling to change why should the rest change? When the agile change begins - the whole organization must change the mindset to not only talk about agile and rename meetings.

Lack of investment means that a change will also cost money and effort. The people need catch-up meetings and training to understand the new process for the future. In this time the workload must be decreased to compensate for this added time. This means that old commitments are not relevant anymore that will cost money. Additionally, an investment in infrastructure is necessary so that a continuous integration pipeline can be set up or more physical space, so that the people can sit as a team in one place, while also having space for ad hoc meetings.

Agile is difficult to implement was mentioned by 48% in nearly every second paper. The misunderstanding of the agile concepts combined with the lack of guidance from literature or coaches result in misunderstandings of the agile approach. After this, two things mostly happened in many teams: (1) The team adapt in their own custom version of their poor understanding of agile or (2) they revert to the old method of working, simply because this worked in the past. Both ways do not help the success of the project.

Coordination challenges in multi-team environment are hard to handle due to the complexity. The interfacing between teams is difficult and each autonomous team

has their own challenges that will impact related teams. In the worst case, this is combined with global distributed teams in multiple time zones with different languages.

Different approaches emerge in a multi-team environment is similar to "agile difficult to implement" however it means that the approaches are different in one project between teams. The teams have a different interpretation of agile between teams or they are using old and new approaches side by side. Both cases have only negative effects.

Hierarchical management and organizational boundaries is a side effect if the agile change is only partly done because of a focus on traditional organization with management hierarchies on teams and departments. As an effect the middle managers' role in agile is unclear or the management still acts as in waterfall mode. None of this is needed for agile development, but in many cases the management structure stays the same based on roles and contracts, but this leads to old bureaucracy or internal silos keeping.

Requirements engineering challenges is a challende in itself; a high-level requirement management is often missing in agile and there is a gap between long and short term planning. Additionally, the creating and estimating of user stories is harder than expected in reality and they must be continuously adapted.

Quality assurance challenges are mostly due to a need to accommodate non-functional testing which is performed by specialized members in a separate team or even department. Even if the Scrum idea is a completely different one. The same applies to automated testing that normally does not exist or only covers parts.

Integrating non-development functions is difficult because roles are not directly defined and so they are resistant to change.

A second research question they wanted to answer is what are the success factors for large-scale agile transformations. The most frequently mentioned factors of a careful agile approach will be described below with the mindset and alignment as the most relevant ones. Other factors, like leadership, are only mentioned 17% of the time.

Management support must be ensured and the management support must be visible for the teams. This is only possible if the management is well educated in agile processes.

Commitment to change is non-negotiable and no one, and no team, must be left behind.

Leadership must recognize the importance of change leaders and place them into the teams. The leaders should not consentrate on the past but look in the feature.

Choosing and customizing the agile approach. Customize the agile approach carefully and map the transition from the old way of working to agile development for a more smooth adaptation. Always conform to a single approach and most importantly: keep it simple.

Start piloting to gain acceptance with a first version by the stakeholders and in the project. Gather insights from a pilot and use the findings. However, one must build a stable version that is not based on the pilot. One does not want to extend the pilot until it is similar to Frankenstein's monster that can not be maintained.

Training and coaching is essential for the project members. Even if they have used Scrum before, a training of agile methods is very important. Additionally coach teams over time, but also allow them to learn by doing.

Engaging people so that people bring ideas into the project/teams. Start with agile supporters and include persons with previous agile experience. They will engage everyone in the organization, then it is not only a management decision, it is something the people in the projects want.

Communication and transparency make the change visible for the people. Make the steps, timeline, training's, executions etc. transparent. Create and communicate positive experiences in the beginning.

Mindset and Alignment by arranging social events which cherish agile communities. Always concentrate on agile values.

Team autonomy allows teams to self-organize so that the teams are in power, but feel a sense of responsibility compared to traditional models. By supporting grass roots level empowerment the teams are empowered to change by themselves.

Requirements management. Recognize the importance of the PO role with all its obligations, but also its powers. Invest in learning to refine the requirements and use backlog grooming to spend more time discussing the requirements in order to avoid unexpected results and chaotic planning meetings.

Having these factors in mind and the selection of the correct scaled agile framework is very important, however additional factors have a tremendous impact on the final project success.

2.3 Teamwork quality

Based on the amount of challenges for large-scale agile development, understanding how a team works and what influences the team performance is an important aspect that contributes to a successful outcome. Poor performance of the team will make an impact on the whole program, forcing it to perform worse than it could.

In 2001, Hoegl and Gemuenden presented the teamwork quality (TWQ) model and a survey to increase the success of projects. They wanted to measure the teamwork quality as a metric based on the results of the survey completed by 575 participants (developers, team leaders, managers) out of 147 German software teams. To find the relevant factors they completed a literature review concerning teamwork and crucial success factors [59]. Based on the literature and on Hackman's "The design of work teams" model (1987) [55] they presented six concepts they called TWQ facets. These facets are shown on the left side of Figure 2.15 with a direct influence on the team performance and personal success.

Communication within a team is its fundamental component. However, communication can be defined in a more detailed way, by analyzing its four measurements: (1) frequency: how much time is spend in communicating, (2) formalization: are the teams able to meet spontaneously or do they require planning, (3) structure: are the team members able to exchange information in a structured way so that the information is being communicated in a way that is understandable for the receiver, (4) openness: is all information shared with the rest or is anything being held back.

Coordination defines how structured and synchronized the team is. Processing tasks in parallel and finishing tasks in time without any gaps or overlaps, enabling other team members to continue with their tasks, is extremely important for a team. Other factors which influence the quality of the end result are the team product, the budget, and the overall time line.

Balance of Member Contributions The balance between the effort, knowledge, and the recognition of work has a high relevance for building team spirit. Not all team members can contribute same as others based on their knowledge or a different skill set. However, all team members must contribute to the furthest extent of their abilities to the team and the other team members should recognize this.

Mutual Support The ability and willingness to help another team member is essential. To be in one team should mean that one can achieve the best results and mutual support to find solutions to the problems faster. Competitive behavior between team members can lead to frustration, poor quality, and additional coordination difficulties.

Effort Apart from the fact that the whole team must deliver the best results, it is also essential that all team members deliver the highest possible results according to their skill level. If one member delivers considerably less than what they are capable of, and as compared to other members, this will lead to internal team problems. All team members must push the project and make it their highest priority. If the team members do not, this could create personal conflicts between team members or lead to an overall bad performance and poor cohesion with the whole team if this behavior is tolerated by the management.

Cohesion can be understood as the team's general feeling of togetherness or the so-called "team spirit". It is difficult to be defined or measured, yet essential for a well performing team. Mullen and Copper (1994) [82] distinguish between three forces of cohesion: (1) interpersonal attraction of team members, (2) commitment to the team task, (3) group pride-team spirit.

To see if the TWQ facets have an impact on the team results Hoegl and Gemuenden defined that effectiveness, efficiency, work satisfaction, and learning are the key factors for a team success. They have a positive and direct relation factor from the TWQ facets shown in Figure 2.15.

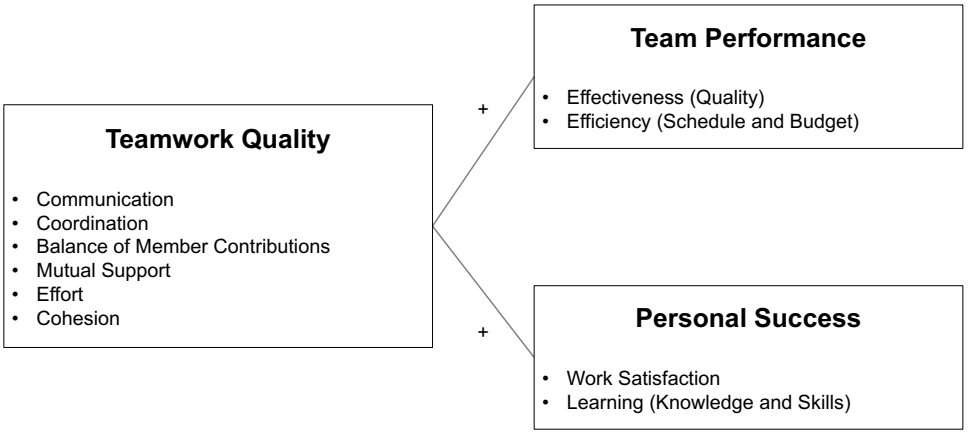


Figure 2.15: TWQ conceptual model [59]

The *effectiveness* and *efficiency* are grouped under team performance. They are what constitutes a successful project in most people's understanding. Most of the other SAFe metrics (see Chapter 2.2.3) focus on velocity, performance of the teams, and the program, to see if the project is on track. *Efficiency* and *effectiveness* are often considered synonyms, however Drucker elaborates on their differences with this comparison [39]:

> "Efficiency is doing the things right"

vs.

> "Effectiveness is doing right things".

The second group is the *personal success* with *work satisfaction* and *learning* as metrics. Satisfaction of the team members is a relevant point for the teamwork and the team cohesion. If the *work satisfaction* is low, the risk is high that a team member will not do this kind of work again. *Learning* is a personal benefit for the person and also a benefit

for the team, because the knowledge will help both the individual and the team perform better in the future.

Survey

The model was confirmed by a survey in 2001 with 145 German software development teams with, an average 6.3 members (median = 6, standard deviation = 3), and mainly male participants. The sixty questions for the team members were Likert scaled (five-point answer scale). Only team leaders and external managers were asked about the team effectiveness and efficiency using the same measurement scales. The questions of the survey are shown in Section 10.1.1.
Based on the aggregated team member response for TWQ facets, the data is analyzed with a factor analyses (Principle Component Method) on the team level. The Kaiser criterion [64] and latent construct assumption proves that all facets are loading high. Based on this a *Structural Equation Modeling* (SEM) was done in AMOS to test the measurement and create a structural model using the *Unweighted-Least-Squares method* (ULS). The results are shown in Figure 2.16. They also compared the results between the team and team managers, and team leaders. Both groups had more understanding of the effectiveness and efficiency compared to the team members. Hoegl and Gemuende reflect that the team leaders normally have a feeling concerning the performance and know more about hidden problems when compared to the team members. The managers also have a higher level view that is more reliant on a metric view based on schedule and budget.

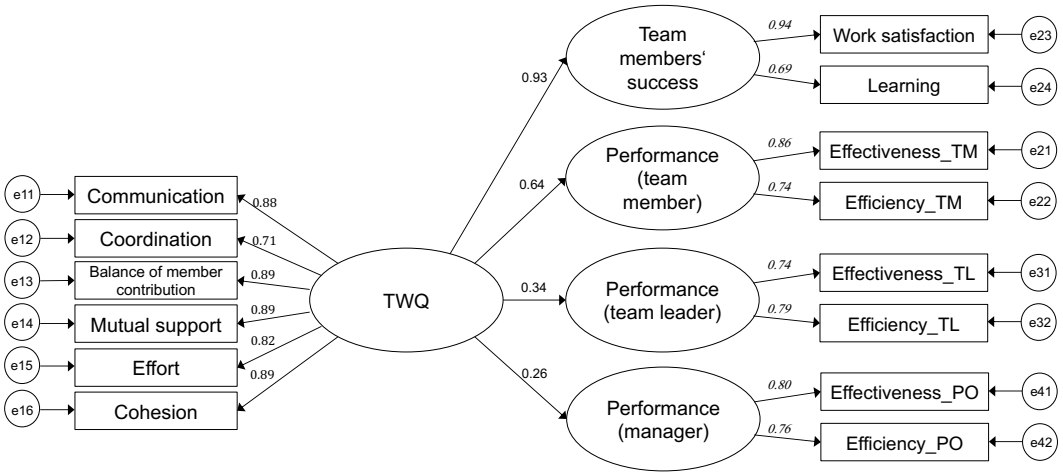


Figure 2.16: Standardized factor loadings and path coefficients for the investigated model of the TWQ conceptual model [59].

The data could also prove that the hypotheses hold. Based on the model, including the

factor loadings, standardized coefficients, variance explained, and goodness-of-fit measures [59], all values prove that there is a positive influence of TWQ on team performance (Hypothesis 1). It also demonstrates that there is a very strong association with team members' personal success (Hypothesis 2). This means that the model from Figure 2.15 is approved and can be used to calculate the TWQ for a team.

However, there is also other research conducted based on the model that proved that there are other factors that are currently not in the model (see Figure 2.15) which can also be proved and are valid. One example is the paper from Weimar et al. "The Influence of Teamwork Quality on Software Team Performance." (2017) [114]. They proved that the six TWQ facets can be adopted by adding expertise to the coordination and they replace *Balance of Member Contributions* and *Effort* with *Value Sharing* and *Trust*. They argue that *Trust* is an important support mechanism and is essential for a team. It is the same for *Value Sharing*. The team must have the same understanding of importance regarding the same topics and task, to be efficient and reduce the conflicts inside the team. But it is always possible to adapt the criteria and find a model. That is one reason why the model from Hoegl and Gemuenden will be used as a base for the organisation's own model. The other reason is the the original paper was cited 1624 times [50].

TWQ model for multi-team-projects

Weinkauf and Gemuenden (2004) used the TWQ model that was co-developed by Gemuenden and extended with positive and negative factors which influence the team performance if the team must cooperate with other teams. They use the TWQ model in the left bottom corner in Figure 2.17 and combine the team performance and personal success factors in one overall team success on the right side in Figure 2.17 as the basis of the model. Based on the simplified TWQ model their study added an inter-team collaboration element which had an impact on the intra-team collaboration and the team success. They prove that three factors must be added if the team must collaborate with other teams: (1) integration (2) conflicts and (3) commitment. Integration and commitment have a positive impact on the inter- and intra-team collaboration and the success. Conflicts have a negative effect on both.

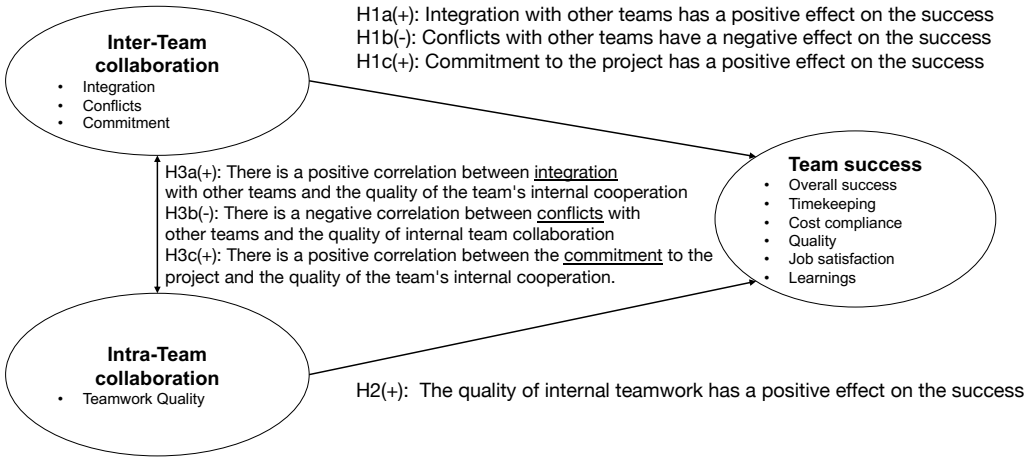


Figure 2.17: Extension of the TWQ model for multi-team-projects [115]

One of the key findings of the study is that the collaboration within and between teams should be actively challenged, encouraged, or managed in order to ensure high performance. They prove, with 407 interviews, not only that the hypotheses from Figure 2.17 are correct, but also that it is not a final model with all the factors. It is only a starting point to find additional performance factors for multi-team projects.

2.4 Multiteam systems (MTS)

During the last decades, the topic of work organizations and structure between teams and companies have been researched by sociologists and other scientists to find connections and influence factors (Gerth (1952) [49], Stinchcombe (1952) [107], Blau and Scott (1962) [9] or Devine et al. (1999) [33]). But not all organizational setups are the same. In the past most of the bigger organizations were set up as matrix organizations [48] where a defined task and management structure will control inter-team communication. Large agile teams act like independent units which must work together to fulfill a task that is necessary so that the whole program works in the end. It is not defined at the beginning who has the lead and what exactly has to be done. This will be decided depending on the task and who can provide the best solution. Based on this structure, the inter-team connections for agile teams are more like police forces, firefighters, emergency medical services, and recovery teams in which each team had to do their own job, but they have to work hand in hand to rescue as many people as possible in an accident. This collaboration model was analyzed by Mathieu et al. (2001): multiteam systems (MTS) [79]. They defined MTS as:

> "two or more teams that interface directly and interdependently in response to environmental contingencies toward the accomplishment of collective goals. MTS boundaries are defined by virtue of the fact that all teams within the system, while pursuing different proximal goals, share at least one common distal goal; and in so doing exhibit input, process, and outcome interdependence with at least one other team in the system" [79].

Each team inside of an MTS (component teams) can have a different goal at the same time, but they have to come together and work together to fulfill the higher level goal that exists for the MTS [5]. A good comparison of who are members of the MTS and who are not is the comparison of an accident. Figure 2.18 shows typical inter-team setup for an accident where four teams (fire fighters, emergency medical technicians (EMTs), an emergency room surgery team and a recovery team) are included inside of the MTS. They all share the overall goal to save as many victims as possible, but each team has their own goal in order to make this happen. For example: the fire fighters want to extinguish the fire as fast as possible and the objective of the EMTs is to bring the injured people to the hospital as fast as possible.

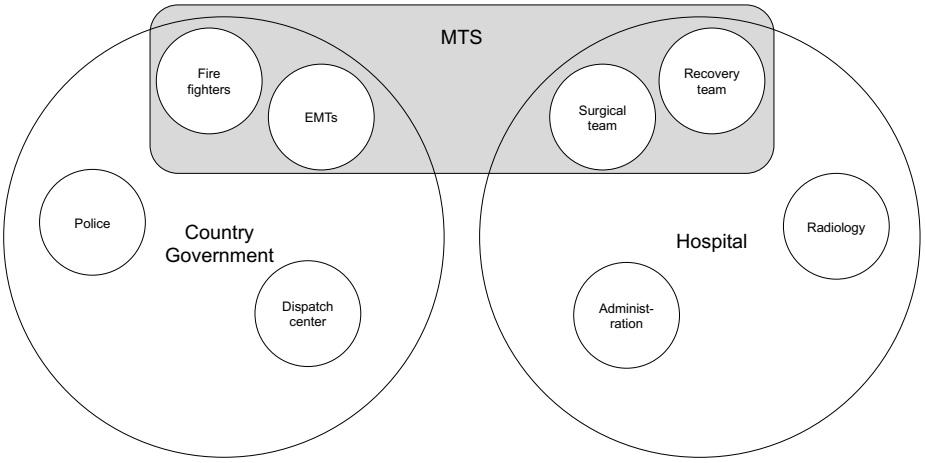


Figure 2.18: Multiteam system for handling severely injured accident victims [79]

Characteristics

Mathieu et al. defined the characteristics of an MTS with five characteristics that must be fulfilled to be a MTS. A large agile development program must fulfill these characteristics so that the MTS model can be applied [79].

MTS are composed of two or more teams that are unreducible, distinguishable, and complete with interdependent members and proximal goals (short-term goals that can be achieved sooner rather than later)

This definition fits for large-scale agile teams. A large-scale agile team must have more than two teams based on the definition by Dingsøyr and Moer [37]. The teams have interdependencies that should be identified during the *PI Plannings* in SAFe to fulfill the proximal goals.

MTS are unique entities that are larger than teams yet typically smaller than the larger organization(s) within which they are embedded

It is important to differentiate a MTS against other team-based organizational forms like a matrix organization, a task force, and an ad-hoc group. All of them define a group of people to perform a job and fulfill a task, but in a matrix organization not all members are connected to each other. So a MTS could be a subset of a matrix organization, but it could also consist of more than one organization. A task force or an ad-hoc group are very similar to an MTS, but by the definition from Hackman (1990) [53] they have to fulfill the following requirements: (1) do not work closely together in their permanent jobs, (2) come together to perform a team task, (3) perform a one-of-a-kind task or create a unique project, (4) have an unusual amount of autonomy of operation, (5) are dependent on external constraints that exist (e.g., clients), (6) are temporary groups given a specific deadline for accomplishing their objectives [53]. A MTS could also fulfill the six points, but unlike MTS, task forces are typically designed to respond to a specific need or project and then disbanded. MTS usually have a much more permanent character. A large-scale agile project has a more permanent nature. Additionally an ad-hoc group normally consists of four to twenty members. Twenty would be the minimum for a large-scaled agile program.

All component teams exhibit input, process, and outcome interdependence with at least one other team in the system

Mathieu et al. grouped the interdependencies in their paper in tree groups based on the research by Hitt et al. (1998) [57]: (1) outcome interdependence: They connect two or more teams to join activities together to achieve a goal at lower level in the goal hierarchy (sub goals) than the overall goal. (2) process interdependence: Describes situations when teams must collaborate simultaneously and collectively together, in the situation when one team's task depends on the result of another team so that the process can work hand in hand (e.g., the EMT achieved to goal to stabilize a victim. After this goal was achieved, to victim must be transported to the hospital.). (3) input interdependence: the inputs (people, facilities, environmental constraints, equipment, ...) necessary to accomplish a goal that are being shared between two or more teams (e.g., the EMTs and the firefighters use the same rescue equipment at the accident). The police and the hospital administration in Figure 2.18 do not share all three forms of interdependence with the teams in the system, which is why they are not part of the MTS network. The interdependence also

matches the large-scale agile program idea. For example, *Enabler Stories* used to have a defined process for process interdependence if one team task required a task from another team to start the following *Sprint*.

MTS are open systems, which particular configuration system shapes the performance requirements of environment that they confront and the technologies that they adopt.

To do this it is important to have a flat hierarchy in order to adapt quickly to new situations and how well the individual teams coordinate their activities. This is one point where most of the MTS research differ compared to the large-scaled agile approach, because they are focused on public teams or the military where the leadership is strictly defined and must be followed. However, Firth et al. (2015) demonstrated that well trained professional teams are more efficient if they are only partially guided [45], which applies to agile teams. The second point where large-scale agile teams are different as compared to a "normal" MTS team is that each team should have one central role of leadership to facilitate horizontal and vertical integration of related activities. Though in case of a Scrum team the leader's position is being shared between two roles, which cover three major tasks that must be fulfilled by a team leader: (1) identify key individual resources and plan the correct timing, sequence, and level of tasks based on the resources (this is done by the PO of the Scrum team) (2) coordinate inter team dependencies and (3) adapt or reconsider the team resources to optimize the team. Points two and three are performed by the SM of the team.

Although MTS component teams may not share proximal goals, they share a common distal goal or set of goals. Furthermore, MTS have a single superordinate goal, in which all component teams have a vested interest

According to Locke and Latham (1990) [75], goals are "desired outcomes in terms of level of performance to be attained on a task". In MTS the teams have their own goals, but they all will contribute to the overall goal. This can be presented in a network where short term goals are at a low level and connect to midterm goals, and at the top there is the long-term goal. All MTS have at least two levels of goal hierarchy. The comparison to large-scaled agile development is that each *Story* is a short term goal. Multiple *Stories* combined are a feature and they combine to form an *Epic*. The priority is to have not only the hierarchy defined, but also the second factor which is defined by the PO and the product management in SAFe. This is combined with the teams in order to have iteration goals and *PI Objectives*. These artifacts help the teams adapt the performance requirements as an open system. Marks et al. (2005) demonstrated that cross-team processes had their biggest value in MTS with a highly independent goal hierarchy [78]. This also makes sense for large-scale agile development. The fewer dependencies between teams, the less communication and administration overhead will be created. This must be done by the POs to cut the *Stories* to the best of their ability to result in fewer dependencies between teams.

Based on the characteristics of MTS, compared to the implementation of SAFe as a scaling agile framework, both match so that the the MTS theory can be used for large-scaled agile programs. This means that SAFe program = one MTS.

2.4.1 Influence factors on MTS

As the characteristics of MTS and SAFe match, the research and results of the MTS area from the last 18 years can also be used. For an overview, the work of Shuffler et. al. will be shown in this section. They have investigated the existing MTS literature for common influencing factors and developed models based on it.

Influences and interactions of MTS attributes

To get an overview of existing influencing factors for MTS, the work of Shuffler is a good starting point. Shuffler is researching the MTS with a psychology point of view, not an IT background. However, there are still interesting conclusions about how the MTS can be used for large-scaled agile development. A cancer care MTS that consists of three teams: (1) day-to-day patient care team (2) oncology team (3) palliative care team are comprised of completely different persons with different backgrounds and different goals, but the process model of the intra-team from Figure 2.19 can be easily adapted to a SAFe project.

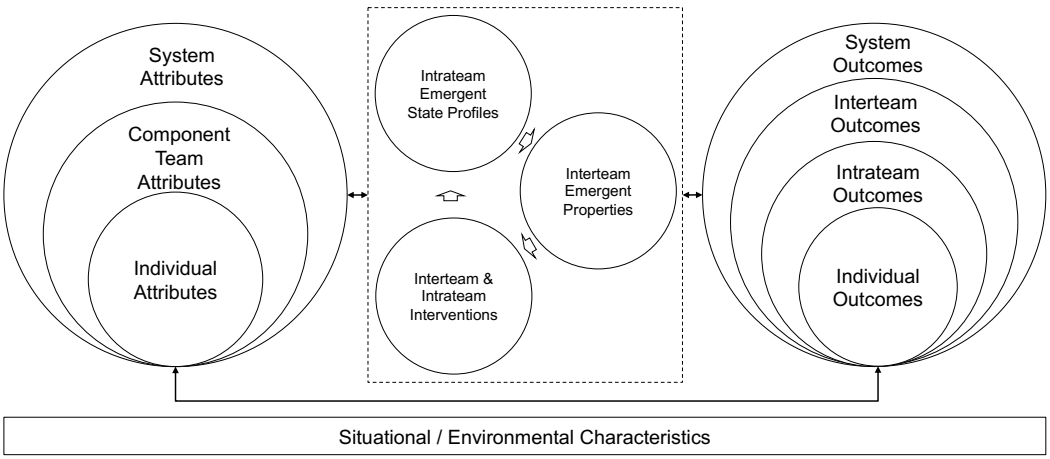


Figure 2.19: Shuffler: MTS - Process model of intra-team state profiles and outcomes [102]

A SAFe program also consists of individuals with attributes that make one team. Together they are the program (system) on the input side. On the outcome side each individual has their own outcome. The whole team outcome is the intra-team outcome which is also a key KPI in SAFe and one performance part from the TWQ model where effectiveness

and efficiency for each team is tracked. The inter-team outcome is the outcome that two or more teams perform to fulfill a task. If this is mapped to SAFe metrics, this is normally not tracked in the standard KPIs, because it is not the outcome from a team that used the Scrum metrics or the program metrics. It is only reflected in the dependencies that teams have to work together. However the negative outcome for interteam issues is normally a problem that has a large impact on the team and program level. If a task is not done in time it will affect the overall timeline of the dependent team that will again effect the rest of the PI.

Shuffler conducted a review of existing studies about MTS to assess intra- and inter-team factors for the outcome [101]. She created a model (see Figure 2.20) based on the literature where the factors are grouped and mapped together depending on how they interact with each other. They are grouped based on the input-mediator-output model by Ilgen et al. (2005) [61] into composition, linkage, and developmental attributes on the left side. These attributes interact as inputs by influencing the inter- and intra-team processes (mediator). The processes use the attributes to provide an outcome in the end. The bold terms in Figure 2.20 refer to constructs that have received significant empirical-theoretical attention, while the italic terms are constructs that require future research. The research focused on the MTS attributes and how they impact the outcome based on the research. Just to highlight some of these factors based on the hypotheses of this master theses they could be grouped in relevant attributes, such as the size and number of teams, but also in attributes that are relevant in MTS, but not for this study, such as the timezone the teams must work together.

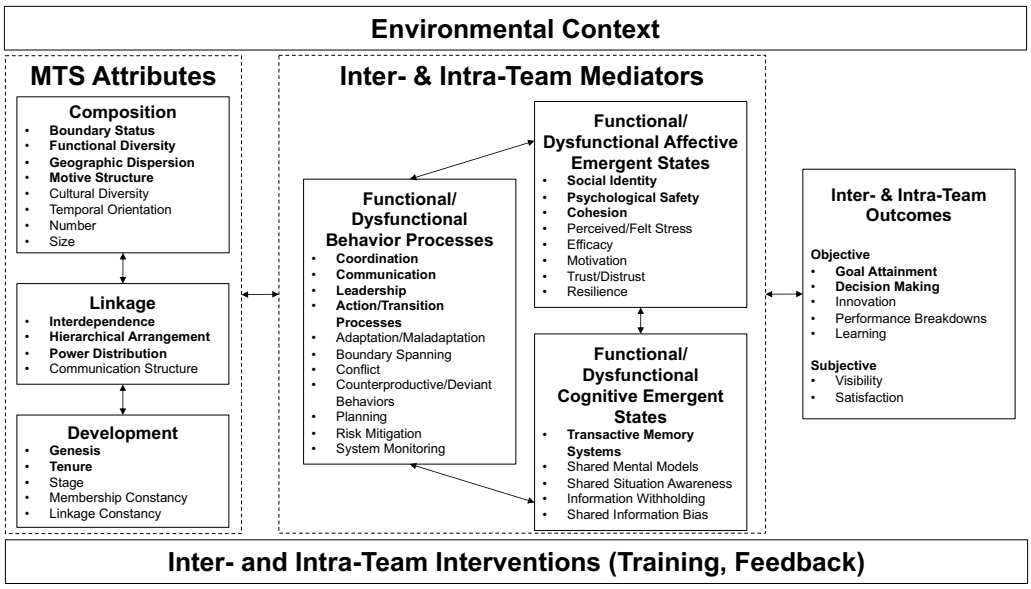


Figure 2.20: Influences and interactions of MTS attributes model [101]

Geographic Dispersion If a *Component team* is spread across more than one location then the team is geographically dispersed [31]. This has negative consequences for information exchange and synchronicity. The impact is higher the more complex the tasks are. To reduce this impact the team must be (1) highly trained at the beginning so that less coordination is required, (2) the leadership can compensate the impact by distributing the tasks across the members or (3) the team can use modern tools and video conferencing solutions to receive updates if the team is geographically dispersed.

Motive Structure It is important for a team to have an unique goal on the team level. If a team has a mixed-motive, then the team does not share the same level of interdependence. This will lead to a lower outcome [34]. A mixed-motive, in many cases, occurs due to bad leadership. A team needs a defined goal and a clear sense of prioritization regarding what must be achieved next, otherwise each team member will decide by themselves.

Size and Number of teams If the MTS size increases this will have two characteristics: (1) the team size increases or (2) the number of teams increases. Both characteristics will have an impact on the complexity, but on a different level (coordination, communication, ...). The bigger the MTS becomes the more important is the leadership, the exact goal definition, and good training to compensate the new environment. Firth et al. compared the outcome between two teams (249 participants). All received an introduction training of forty-five minutes. Half of the teams (119 participants) received an extra frame of reference training (fifteen minutes) concerning quality. Defining the same standards for performance helps coordination between teams in MTS [45].

Hierarchical Arrangement and Power Distribution This attribute depends on two factors: (1) is the team a support team (many external dependencies) or not. For support teams the performance increased with horizontal coordination compared to vertical coordination. Teams with less "outgoing" dependencies have nearly the same performance for vertical and horizontal coordination [30]. (2) Is the team lead prepared and enabled to lead the team. If a team lead is competent and knows the skills of the team members, than they should take full control and have the last decision on the topics. If the team lead has only a rough knowledge and no management skills than they should only coordinate and let the team members devise by themselves.

Dependence between Team and MTS goals

This MTS model (see Figure 2.21) was developed by Guthrie et al. (2005) and it uses the common factors that are mostly mentioned in other models and papers [52]. However, it makes one very important point clear: there is a direct connection, with a positive and/or negative impact, between the team and the MTS performance. If a MTS team is focused

on the more global MTS goal, individual team processes will be worse. And if a team is focused on the team goals then the overall MTS goals will be worse. So it is important that both goals keep the balance. The research focused on two primary processes: (1) Team processes (i.e. communication, coordination, leadership, conflict resolution, decision making etc.) are important for effective performance. (2) Communication and leadership are more relevant the more teams are involved. An additional research result was that a MTS with a lack of sharing mental models will likely perform poorly because communication will be less effective. However, as the teams communicate, shared mental models can be developed increasing the effectiveness of communication and improving performance.

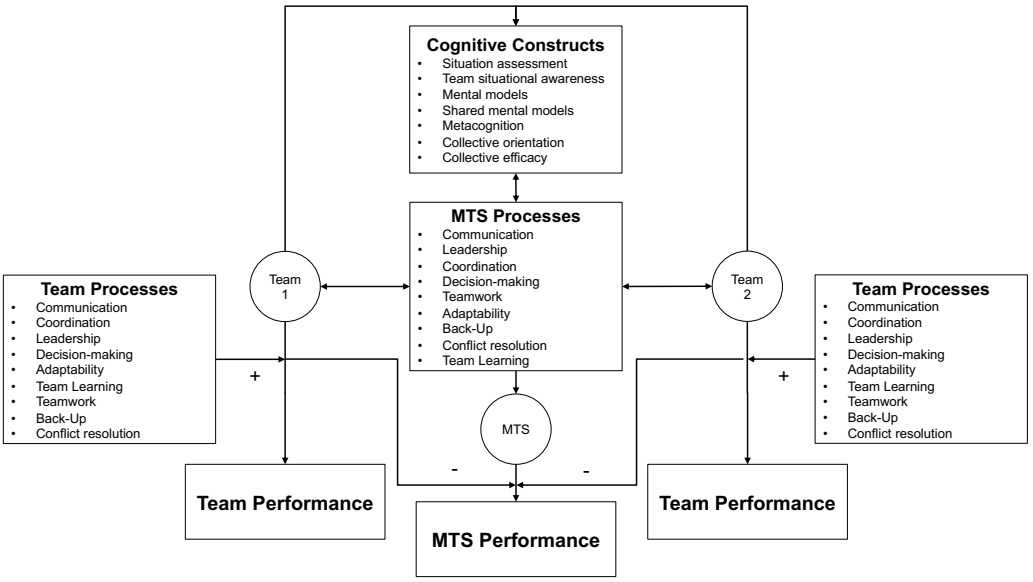


Figure 2.21: Performance in MTS - effectiveness model [52]

3 Related work

This chapter details the related publications of influence factors for the team and program performance for agile development. Some of the publications have an own model that can be compared to the model from TWQ in Section 2.3. The related research is split in two parts. The first part lists relevant research, with a focus on intra-team performance, and the second part mainly focuses on inter-team performance models for large-scale agile development.

3.1 Intra-team

Since agile methods (see Section 2.1) have been around for several years, research has already been conducted in this area. Various influencing factors have been compared at a team level to see how they impact success. In this paper, the focus will remain on the factors that are easier to change in an ongoing project, rather than the underlying decisions and management.

Dingsøyr (2014-2019)

Dingsøyr is one of the most known researchers focusing on large-scaled agile development in the last years. He was one of the participants who created a definition for team sizes in large-scale agile development [37], coordination of large numbers of teams which had an impact on inter-team coordination in large-scale agile development [36] [38] and based on the TWQ model, he compared the factors and the performance between agile and traditional development. To achieve these results he conducted a survey with 477 respondents in 71 agile software teams to determine how the TWQ performance and team members' success is effected for agile software teams. He also wanted to determine how they can be compared to traditional development teams by using the same questionnaire (see Section 2.3) that was used by Hoegl and Gemuenden. They found out that TWQ is more important in traditional teams for the team performance than for agile teams, but only with a marginal difference. The mean values of the factors are also very similar compared to the other model shown in Figure 3.1. One explanation could be that TWQ increased in implementation over the years, but the expectation of agile teams are higher now compared to before. Based on the similar results, the TWQ also influences the team performance for agile teams for the team members. On the other hand the team leader's perception only has a medium correlation to the team performance and the product owner has no impact on the team performance. Their suggestion for future work is that additional factors could be taken into account to see if they have an impact on the team

performance such as the effect of offshore vs. local teams, public vs. private sector or the level of team interactions with the PO [74].

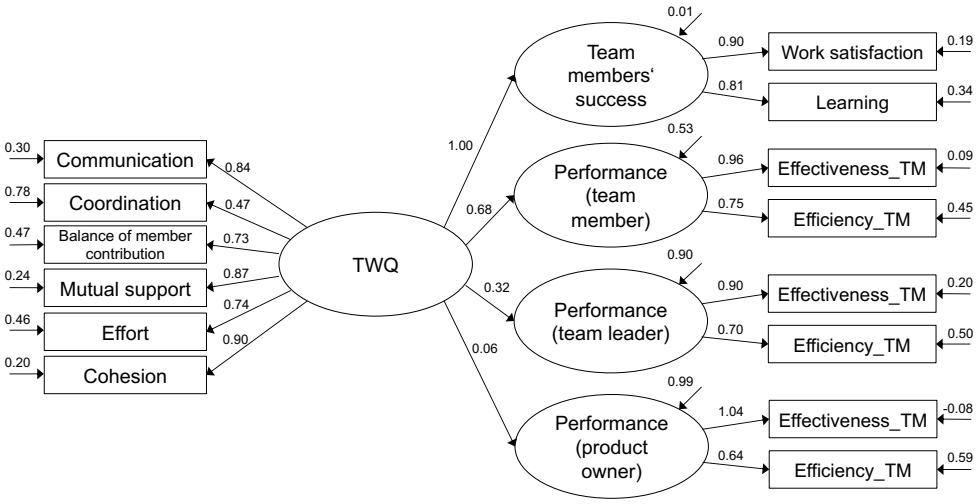


Figure 3.1: TWQ Model with standardized factor loadings, path coefficients and error variances [74]

Espinosa et al. (2007)

Due to the high demand for skilled developers in the world and especially in metropolitan areas it is very hard to find enough team members in one location. As a result, bigger teams normally have team members that work from different locations. With geographical distribution many additional challenges arise for the teams, such as geographical distance, different time zones, cultural differences, and different language skills.
Espinosa et al. conducted a questionnaire and interviews to determine the impacts of geographical distribution in large-scaled agile development. He wanted to determine:

> "How do various types of team knowledge affect coordination in software development? And, how do these effects vary with geographic dispersion?" [44]

The overall result is straightforward: It is more difficult to coordinate tasks across sites compared to coordination within a single site. The distance causes all sorts of problems associated with the distance. However, two fundamental impacts can be distinguished: (1) The communication frequency. This could include discussions in the background surrounding some team members or a short chat about one topic next the coffee machine that improves the team knowledge as opposed to scheduled meetings. (2) The loss of

information through digital communication. Video conferencing is state of the art in geographically distributed teams and they help with communication, but some contextual references are lost through this type of communication. As a result the shared knowledge is different. Based on the interviews it was found out that coordination consists of tree components (1) technical, (2) temporal (e.g. delay) and (3) process. Typically a developer has technical coordination problems. The management normally has temporal or process coordination problems with distributed teams. One way to address these problems are a shared knowledge of tasks, and the team members for the developers, and better task and presence awareness for the management. One way to implement this solution is to increase the use of tools where the knowledge is documented for everyone and direct communication tools with video in order to lose as little information as possible. During the interviews it was discovered that members were less inclined to discuss problems if the other members were distributed geographically and that shared knowledge of the team is more important for team members that are distributed geographically, but that shared knowledge of tasks is more important for co-located team members.

Other researches have come to the same conclusion that geographic distribution in software development has a negative impact (e.g.: Erran (1999) [17], Espinosa et al. (2007) [43], Herbsleb et al. (2003) [56], Ramasubbu et al. (2007) [85]). Physical distance creates barriers that make coordination, communication, and awareness (Cummings et al. (2009) [28]) lead to a longer time to finish a task (Herbsleb et al. (2003) [56]) and to more bugs (Espinosa et al. (2007) [41]). This, in turn, results in worse software with higher costs than normal which cannot compensate with the lower daily rate.

Espinosa (2015-2019)

Espinosa analyzes the impact of distributed teams in different areas such as the temporal distance [42], teams in different time zones [18], and virtual teams [25]. One of his findings is that team members that have worked together in the past know each other and have a relational resource that, in general, improves the team performance, even if the relationship is negative. In 2007, he conducted a field study of geographically distributed software teams to discover the impacts of changed based on the task familiarity and team familiarity on team performance in such an environment [43]. As shown in Figure 3.2 they focused on five hypotheses that have an impact on the team performance.

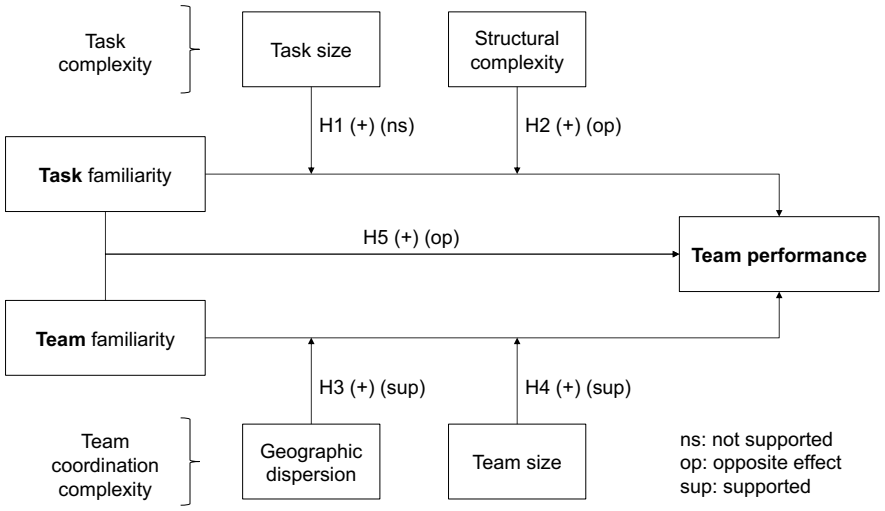


Figure 3.2: Espinosa et al. - Team Performance Model [43]

The results show that, in general, the familiarity is positive for the performance of the team. The second finding was that more complexity (bigger tasks, structural complexity, geographical distribution and bigger teams) had a negative impact on the performance. The more interesting results are: (1) Task familiarity is more beneficial than team familiarity for less structurally complex tasks. (2) The size of tasks are not related to team and task familiarity. Bigger tasks will take longer even if the team is familiar with the task. (3) Team familiarity helps teams to reduce the negative effect of bigger teams. Team building events can help to create a sense of team familiarity. (4) The same counts for geographical distribution. A video conferencing system can help to increase the team familiarity. The final finding is that task familiarity improves team performance more strongly even when team familiarity is poor and vice versa [43].

3.2 Inter-team

Lindsjørn et al. (2018)

Lindsjørn et al. analyzed the influence of small and large agile projects on TWQ. An exploitative survey was conducted with 64 agile teams (31 teams in small projects and 33 teams from large projects) and 320 team members and team leaders. They defined that small projects consist of one or two teams and a large project consists of 10 or more teams [73].

Figure 3.3 shows the relationships between six TWQ facets in grey and team performance in blue and red. In small projects the product quality (ProdQ) correlates more closely with TWQ for team members in blue and team leaders in red than project quality

(ProjQ). Product quality is more central and closer to the quality of teamwork. Project quality is further away and has weaker correlations to TWQ. The same result exists for large projects between the facets and team performance for the team members, but the team leaders product quality has a negative correlation to several facets. TWQ may influence product quality more than project quality in small projects, as team members and team leaders work closely together in small projects with little need to plan ahead, compared to larger teams that need stronger mechanisms to control costs and schedules.

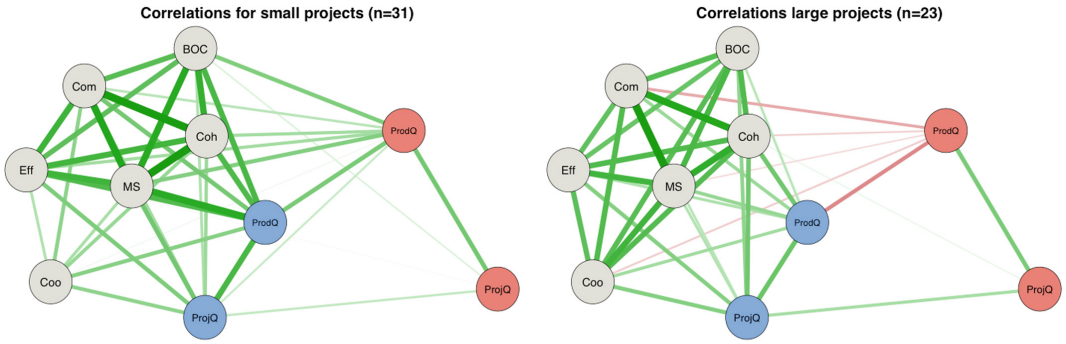


Figure 3.3: TWQ correlations for small and large projects. (Com = Communication, Coo = Coordination, MS = Mutual support, Eff = Effort, Coh = Cohesion, BOC = Balance of member contribution, ProdQ = Product quality and ProjQ = project Quality) [73]

The six facets also have different correlations between small and large projects in some factors. The coordination (Coo) is more positively associated with all other five factors, probably because the larger the project is the more important the coordination for the overall TWQ. The other factors of the six facets change marginally between small and large projects. This is slightly surprising as the authors have assumed that the motivation teamwork aspects (effort, cohesion and balance of member contribution) have a greater influence on smaller projects and the three interaction aspects (communication, coordination and mutual support) have a greater influence on large projects. Unfortunately, there is no explanation for this and further research and more data are needed to make a definitive conclusion.

Scheerer (2017)

SAP SE is, with 96.000 employees [100], one of the biggest software development companies in the world. This is a very competitive industry field with many big customers with special needs where they must adapt their solutions. Scheerer is the *Agile Coach* and *Scrum Master* at SAP where they adapt large-scale agile development. He determined that with agile development there must be a fundamental shift in companies in how they solve complex issues, especially in terms of coordination between teams. He uses the MTS (see Section 2.4) from Mathieu et al. (2001) to find strategies for team coordination in

order to optimize the program performance [94]. This is the only research that could be found which combines large-scale agile development with the MTS. Even if he focused on coordination, the results are still quite interesting. In the paper he discovered a contradiction between the agile policies and key requirements for good communication and coordination. He conducted a case study in a real-life industry setting at SAP SE with ~140 employees in thirteen teams at four locations. He grouped the forms of coordination in three groups: (1) Mechanical coordination - Scheduled or regular coordination with little communication (2) Organic coordination - Coordination through mutual adaptation or feedback through interaction, which can be formal and planned, or informal and spontaneous (3) Cognitive coordination - based on explicit and tacit knowledge of each other. Organic and cognitive coordination between Scrum teams is hardly present due to the explicit Scrum guidelines, such as focusing on the team-related backlog, fixed sprint cycles, and predefined roles. Mechanical coordination is quite limited in agile development, based on the definition of the SM and PO. Both roles have a very limited mandate to control the team. He proposed to change the guidelines and adopt them so that deeper coordination could be established over the teams. This will have a positive impact about the delivery predictability.

4 Case study

This chapter will provide insights into the case study partner in Section 4.2 to answer why and how the case study adapted with nearly 150 people from a mix of waterfall model and Scrum with a big bang to SAFe. The Section 4.3.2 and Section 4.3.1 provide an overview about the team and program setup of the partner and explain why this partner is interesting to analyze. It also describes why the additional factors that are added to the TWQ model are chosen. The information will be than used in the survey evaluation in Section 7.

4.1 Case study design

The case study must provide as much relevant background to the reader as possible to understand the environment and the decisions that are made by the case study partner. To fulfill this, the case study is conducted using the guidelines from Runeson and Höst (2009) [90] and by following and answering the elements of the plan by Robson (2002) [88].

Objective — what to achieve? The case study is an exploratory study that wants to achieve two goals. (1) Get all relevant background information regarding how the large-scale agile transformation was done and worked for the case study partner to document how it could work, what worked well, and what could have been improved. (2) Analyze the team setup and team structure to find relevant KPIs for each team concerning the team members and team performance that will be used as a base for the survey and which can be compared to the survey performance results.

The case — what is studied? The case study partner is described in more detail in Section 4.2. It was observed between September 2018 and March 2019. This was the time shortly before the first PI-Planning until the beginning of the third PI.

Theory — frame of reference To understand the goal of the expiration case study, it is important to have a basic knowledge about agile and Scrum in general and for *SAFe Essential* in more detail. The relevant theory about agile, Scrum, SAFe is described in the Chapters 2 and 3.

Research questions — what to know? From the four research questions that should be answered in the thesis (see Section 1.2) two are based on the case study, especially on the survey.

(1) **Research question 3:** Can the teamwork quality (TWQ) model be applied by the case study partner on the team level and are there any additional significant factors that can be added?
(2) **Research question 4:** Can an influence factor model be applied by the case study partner on program level?

Methods — how to collect data? The data is collected in different ways. **Observations** and **unstructured interviews** were conducted during the *PI-Plannings* where the whole program came together and the transformation could be compared for each team. Additionally there were **semi-structured interviews** with the main roles (*Business Owners*, *System Architect*, *Release Train Engineer*). The historical view of the partner, the team setup, and the performance were done by **document analysis** that was provided by the partner.

Selection strategy — where to seek data? The information was researched by a **case study** in this Chapter. To answer the research questions concerning influence factors for team and program performance, a **survey** was done encompassing the whole program based on the TWQ model, which is described in Section 2.3. The survey questions and the results are presented in Chapter 5.

4.2 Case description

The case study partner: EGP [2] is a German software developing company with nearly 150 employees with offices in different locations in Germany. The partner is developing a software solution based on modern standards. To give an example, some buzzwords include: docker, cloud ready, DDD, progressive HTML5 web-page and microservice architecture.
The reasons why this partner is interesting to analyze and use as a base for the research are:

Timing The research started shortly before the large-scaled agile transformation started for the teams of the partner. This gave a direct and unmodified impression of the implementation of SAFe and not a historical retrospect with information loss.

Size With more than 10 teams and 150 people it counts as very-large scale agile development [37].

Setup There are enough factors that can be analyzed, but not too many that makes it difficult to draw conclusions based on the data. This results in many distributed and non-distributed teams in different sizes. At the same time, most team members have the same background and speak German as their first language.

[2] EGP Gesamtbanksteuerungssysteme GmbH & Co. KG `https://www.egp.finance`

History

After the financial crisis of 2008, many banks were rescued by the government. To ensure that this will not happen again, many new German and European Union regulations were created. The problem that banks face is that the government and the EU require more data from the banks to observe a health indication. At this point two companies, that were partly competitors in some areas, joined together to build a new company in the financial industry to provide a solution for banks. This is a very complex task, based on the many different bank products and the necessity that the calculated numbers must be correct, otherwise they would fail to meet legal requirements.

The company was founded in 2016 by two big players in the German software industry. Each of the two companies counts nearly 7,000 employees. However, one of the companies is more focused on infrastructure and IT operations, while the other company is more of a software consulting or solution provider. The benefit for the new company was that they had access to skilled developers and a running and maintained data center.

Between 2016 and 2018, the first version of the software was developed based on the requirement document and mostly by using the waterfall model. Only some teams started to work independently after the Scrum framework. The whole company started with roughly twenty developers in the initialization phase, but quickly grew to sixty developers. At the end of 2016, they had nearly 100 developers working on the software.

Based on the results in the first two years, the management and stakeholders decided to change to large-scale agile development (SAFe Essential). They decided to do this with a big bang approach by completely transitioning all teams to *SAFe Essential* with their first *PI Planning* in September 2018.

This quote explains the challenge and the timeline quite well:

> "We have started as a speedboat to fulfill a legal requirement and managed it. But at the same time we stared to build a supertanker and tried to overtake the speedboat. Now we are nearly even, but to achieve this we had to drive the supertanker at full speed."

Through the agile approach they wanted to achieve the following benefits:

More flexibility with requirements This is an obvious benefit to change to agile development. This could be bigger requirements that are legal changes that have a bigger impact, but also very small things such as text changes in the UI or reports that should not take months until they are delivered to the end user.

More transparency between the teams With the high number of developers that are working on the same software, that was created in such a short time, a large problem appeared: the dependencies between teams were not clear to everyone based on the requirements definition document and so each team tried to build a component on its own without knowing what must be done in what time frame so that other teams are not blocked. This became clearly crystal in the first *PI Planning*, when, for

the first time, all teams had to identify with each other which dependencies exist to place them on the dependency board (see Figure 4.6).

More transparency for the management and stakeholder In the traditional project flow the requirements were written down and then developed based on the document. There is normally a project controlling what should be taken care of in the progress. In the end the final product is only presented after all teams have developed their requirements and then the management will see the results. This has two very hard impacts that cost money and time to solve. (1) If the developers understand something differently in the written words it is nearly impossible to make it clear until it is finished. (2) The teams normally do not integrate their components with the other components until the end (in between the mock the interfaces) and then it results in additional work to do it. Both challenges could be solved in the traditional development process, but they got automatically solved by implementing SAFe. The current progress and status is transparent in the boards of the teams and shown to the stakeholder in a system demo. With an automated CI/CD pipeline and automated testing there should always be an integrated and running product.

4.3 Large-scaled agile adoption

As scaled agile framework they have chosen *SAFe Essential* because of the market share, compared to other frameworks that bring some benefits with it. The most important criteria for the selection of SAFe was the documentation for the case study partner. This focused especially on the implementation effort and what is documented and already prepared and available, so that all team members can find the most relevant information quite easily. If this is not the case, then all the descriptions for the teams, PO, and SM must be contacted by the case study partner that would be a big effort. This is relevant not only in the initial effort, but also the long term effort and should not be ignored. There will always be change in the program with new developers and if a lesser known or badly documented framework is used it will take longer to onboard the new people. The second plus point of SAFe is the adoption of Scrum inside of SAFe so that teams that have already used Scrum and other team members who are also familiar with Scrum from other projects or the university can easily adopt it. The second point also fits to other models like *Large Scale Scrum* (LeSS), but they are less documented. The comparison and the decision for SAFe was chosen based on the internet research and a small literature research for most decisions. In the beginning LeSS and Essence were compared, but in the very early stage they focused on SAFe. One important point is that the decision was done completely based on their own environment and not on already used large-scale agile frameworks that are used in the company. This is interesting because one of the two owners develops software in the automotive industry, or in the public sector where large-scale agile is used. However, they decided to use SAFe due to some experience exchanged with another subsidiary company in the banking industry that also uses SAFe.

Based on the criteria, the *System Architects* and RTE made decision documents for the management that approved the decision for SAFe. This is also mentioned in many research papers [35] that the decision must come from the top and management commitment and support is required for such large change in the company. The research work was conducted between April 2018 and June 2018. After the final decision, the whole transformation and training was conducted until the first *PI Planning* in September 2018. This is a very short time frame for a full transformation. The time was mainly used to train the SM, PO, and *Solution Product Owner* in Scrum and SAFe theory so that they would know exactly how the process works. They should be the first contact point for the team if they have any questions about the process. The team members only received a crash core with half a day for one day before the first *PI Planning*. If there were specific questions or different understandings, then the team members could contact the RTE for help. If they were not completely sure they could contact an external agile coach that can consult in hard questions or misunderstandings.

Challenges during the large-scale agile adoption:

Make the transition and start working The teams worked during the transition on new features and bugs in the first version of the software and the other parts of the team started to work on the second version. This mix resulted in the fact that the transition caused many problems that could have been minimized if they would have defined an end date where all the tasks must be done, then start the transition, and then start working in an agile world. This also applies to the tasks. Some teams worked until the end with a requirements document in waterfall and at the same time the *Stories* had to be prepared for the first *PI Planning*. This lead to *Stories* that were either wrong or already partially done. This tied up many capacities that could have been invested better.

Different knowledge about Scrum and SAFe Some team members had only heard of Scrum, but never participated in an agile team. They all received the half day agile crash course. In retrospect, the half day was not enough for many team members to understand the whole SAFe process. There are basic questions that must be clear to everyone, such as how *Story Points* are estimated (Complexity $\neq$ working days). All this must be recapped by the SM and PO during the first *Sprints* which cost time and effort.

Measurement of performance The management must be able to get an answer if the progress fits the time line. This is always a very complex task, because not all *Stories* are estimated and *Story Points* are not equal to working days. However, the management only has the funds to pay for a certain amount of working days, so an understanding mus be achieved on whether a finished and running system can be delivered in time and within budget. The RTE role has the responsibility to track the progress and report it to the stakeholders.

Adopt the old organization structure As in most typical organizations, the case study partner had some management and middle management roles that did not fit the SAFe roles. They started by moving the roles from the management role to the Scrum role. Most of the sub project managers became PO or SM but there were also some people who could not handle the "degradation" from their point of view and left the program. However, this was only a small group. The overall project leaders remained based on the ownership of the company. One way or another, the organisation must adopt to SAFe and not the other way around.

Operation, maintenance and problem management The focus of SAFe is to develop new features that extend the software and they have highlights that come with SAFe like Test-Driven-Development with build-in quality. But a software is a living system and it is unreasonable to expect a software without any bugs or problems. The question is: how do you handle the incident and problem management. Do you expect a defined number of incidents in each *Sprint*? If there appear less, then originally expected, than you can add an additional *Story* at the end. What if you have a major incident of a high severity for the end users? Would you stop the *Sprint* for one team or for the whole PI? Those are all difficult questions to answer. They used an initiative for maintenance to at least track the effort and get a feeling how much effort it is.

Overall they are very proud of their transformation. There will always be things that could have been done better, but in the end the transition was done with a big bang and this worked for the teams. They have completely implemented SAFe and will stick to it. The only thing is that they will include the *Portfolio Level* from SAFe in the future to have a better overview. However, it was a good decision to start small and gradually extend.

What they missed in the surroundings of SAFe is a good case study which describes how a transformation really works and what best practices should be used to increase the acceptance of the adoption. The process, the roles, the artifacts, and the meetings are well documented, but how do you get people together and how do you train them is not completely clear. Pictures and texts are impotent, but to get it on the road is a completely different thing. Even in the trainings that are being offered at the SAFe website only very few insights in the real transformation are given. For example: "How do you transform a group of heterogeneous developers that have developed decades in a given and steady waterfall environment?". They have increased the training for this group of people for them to be able catch up with the rest of the program, greatly focusing on the mind.
The second point they miss is a more detailed explanation and tooling on how to control the teams and what KPIs are relevant to make a forecast. They use the current team sizes and the *Solution Product Owners* as expert estimation to estimate complete *Features* or *Epics* in working days.

The fact that SAFe is more of a commercial product than an organization that wants to

help people is not a bad thing. This has the benefit of the focus being to optimize the product and provide enough information that is needed.
For them the most important criteria, if they had to choose a large-scale agile framework again, would be the initial situation. How many developers do you have at what skill level and in what locations? In many cases Scrum would be enough or a small *Scrum of Scrum* will be enough for smaller programs. If necessary, a bigger scaled agile framework should be used, because it brings structure, but it will also bring more complexity which will lead to problems.

4.3.1 Program setup

This section describes the program setup in more detail. It will explain how *SAFe Essential* is implemented and how the teams are located, split, and sized. This information is important to have a better understanding of the case study partner. The team size and the team distribution are used to find out if they have an impact on the TWQ. All numbers and the figures are based on the data of March 2019 at the beginning of PI3. How the teams change between the beginning of PI1 in September 2018 and PI3 in March 2019 is explained in Section 4.3.2.
The case study partner has tried to adopt, as close as possible, *SAFe Essential*, but based on the setup they had and the challenges described in Section 4.2. The result is shown in Figure 4.1 on high level. In total, the whole program consists of 148 people of which 111 (75%) are developers or testers and the other thirty-seven (25%) are management, SM, or PO.

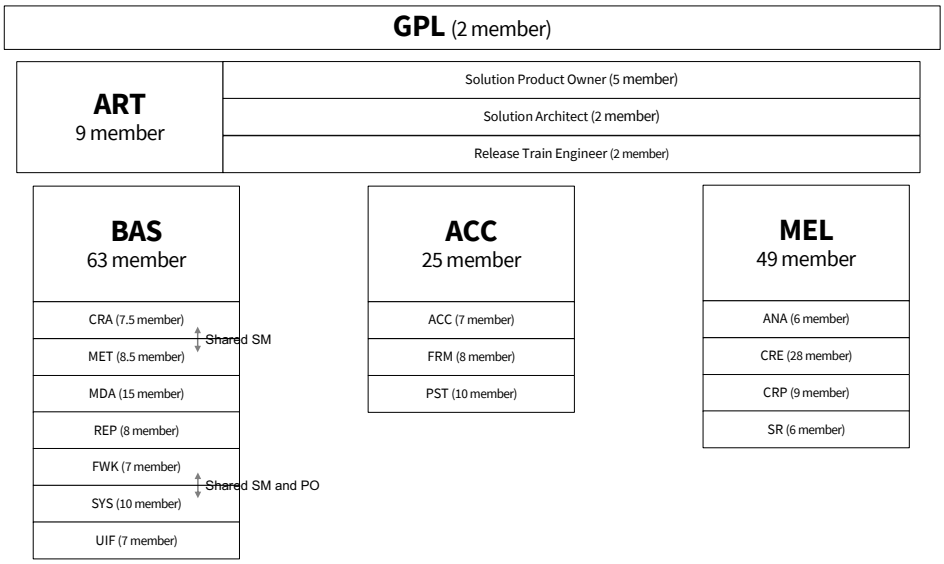


Figure 4.1: Program overview of the case study partner

The overall project lead (GPL stands for "Gesamtprojektleitung" in German) are com-

prised of two people. This is due to the fact that the company is owned by two companies, each being represented by one individual. They are responsible for the business decisions and how the money is spent. Compared to the SAFe roles they are the *Business Owners*. Under the GPL there are three teams that are adopted to the SAFe model. How they are integrated into the program is shown and explained in Figure 4.2. Fourteen teams are combined into three clusters that match from a business point of view. Overall the three clusters are only a virtual split over the teams. There are no borders between the three clusters so that connections and dependencies exist over the whole program. The *PI Planning* is completed with all teams at once. Detailed information about the team size is explained in Section 4.3.2.

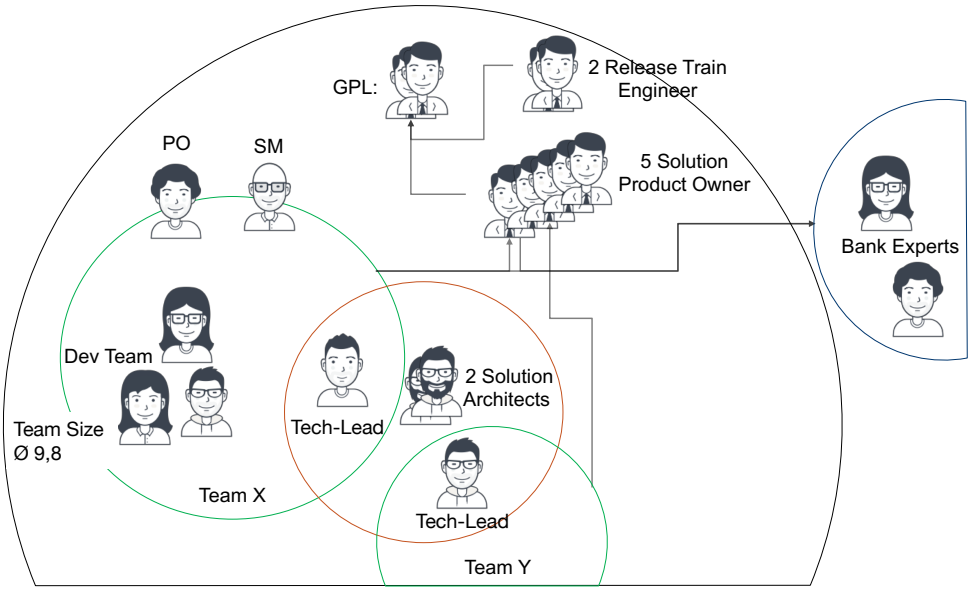


Figure 4.2: Relation between the program members based on the rolls of the case study partner

In Figure 4.2 the program setup is shown again on a different level. The green circle on the left represents the teams. A Scrum (see Section 2.1.2) defined team consists of a *Scrum Master*, a *Product Owner*, and the team members (developers). One or two team members also have the role of a *Tech Lead*. The *Tech Leads* work together with the two *Solution Architects* (in SAFe: System Architect/Engineer, Section 2.2.3) to define interfaces and frameworks. More about the *Tech Leads*, the *Solution Architects*, and architecture process is described in Section 4.3.3. The five *Solution Product Owners* (SAFe: Product and Solution Management) build the bridge between the the customer needs, the legal requirements, and the teams in the three clusters. They also establish the contact between the Scrum teams and the external bank experts if specific explanations or details are needed. This means that direct communication is allowed between the teams and these experts. This differs depending on the team and the *Solution Product Owner*. To support the program, two *Release Train Engineers* (RTE) help the teams in any required way. Additionally they

have the important task to track the progress and the results. These are required by the GPL and the stakeholders to see if the program can be completed in the expected timeline. Besides the shown roles, there are two participants whose task is support the program. The owners support the program with an existing team of *Project Management Office* (PMO) to organize all kinds of tasks such as the *PI Planning* with hotels, rooms, and food. The second thing which is provided by the owners who are experts or colleagues not part of the program, but still help with advice or server management and tools. The tools that are used are industry standard with *Jira* as an issue tracker with the portfolio plug-in, *Confuence* as documentation, and the knowledge system from Atlassian. As a software repository and build system they are using *git* with *Jenkins*. To communicate inside the program with team members they use *Skype* or *zoom.us* for video conferencing.

Since September 2018, they have finished three *Program Increments*. Each of the PIs took 3 months which each had six *Sprints* with a length of two weeks each. The last *Sprint* is the *IP Sprint* that not only allowed the team to finish and document, but also to work on innovation and exploration tasks. Some of the Sprints differed depending on the holidays in Germany which made it quite hard to coordinate. At the beginning of each PI they had a *PI Planning* meeting, as defined in SAFe where all teams meet together in one location. During these two days the main scope should be defined and specially the dependencies between the teams must be made visible for all teams that are related. This meeting should give a short overview over the *PI Planning* and what the main findings from this event are. Figure 4.3 shows the meeting room after the first *PI Planning* day with the team workspaces and their planning process an the walls.

Figure 4.3: PI planning room

They used the agenda schedule plan from SAFe. This means that the GPL presented the current market situation of the product, followed by a presentation of the *Vision*, road map, and goals from the SPO. The next agenda point was the current situation and the goals from the solution architect. All these points were more or less presentations without any real feedback. This changed in the first PI planning for the agenda point "Planning context and lunch" where many questions were asked about the process. This is due to the fact that only three team members had participated in a large-scale agile project so far and that the half day training was not enough for most of the developers. All the questions

were answered by an external consultant who was an expert in agile transformation. In the first *Team Breakouts*, all teams estimate their velocity in total and for each *Sprint*. They also started to plan the goals they want to achieve with the defined *Stories*. In this step, the preparation of the PO makes a big difference if they have prepared the vacation plan for their team, *Stories*, and a rough timeline. Figure 4.4 shows an empty workspace for a team with sheets on the wall for each *Sprint*, the goals and risks, and what cannot be done in the PI. They differentiated *Stories*, *Enabler*, risks, and goals with different colors. This technique helps to see how much time a team spends with preparation or final end to end *Stories*.

Figure 4.4: Prepared team workspace for a PI planning

During the *Team Breakouts* the SM meet twice to check, in a short *Scrum of Scrums*, if all things are done and what tasks are open for the team. In this case the *Stories* are not the focus. The idea is to check all organizational tasks, such as if the definition of done is taken into account or if the team has started to define dependencies. This is checked with a sheet that is shown in Figure 4.5 where all tasks are listed on the left and all teams on the top. This helps the SM to see the progress of each other.

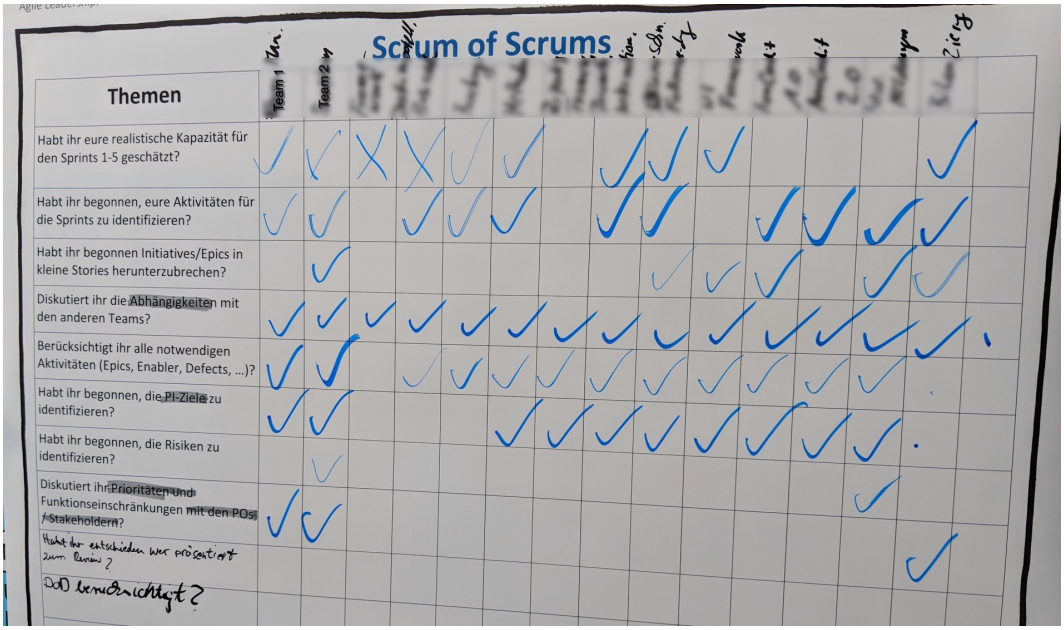


Figure 4.5: Scrum of Scrum status board

In the *Draft Plan Review*, all teams presented their results and a short status. This is the base line for the *Management Review* and *Problem-Solving Meeting* where open points and possible problems are discussed and solutions should be worked out. The first *PI Planning* leads to some shifts between team members, because of a lack of knowledge in some teams.

All management decisions were presented the next morning to the program so that the participants were all up to date. Based on the adjustments the second *Team Breakout* should give the teams the time to finalize their planning. If this is done properly, the team will not need a *Sprint Planning* meeting for the first *Sprint*. With the final objectives, the *Business Owners* assign a *Business Value* for each team objective they can adopt if the time gets short at the end of the PI. Additionally, what should be visible at the end of the meeting are all dependencies between the teams on the *Program Board*. Figure 4.6 shows the comparison between the dependencies after the first *Team Breakout* and after the second one. This figure makes it extremely visible why an on site meeting with all teams makes sense. At this point, it is even more interesting, because the teams already had most dependencies before the start of the first PI. The system team especially had to do a lot of preliminary work for the other teams, because otherwise they are blocked. The board is essential to reduce the complexity between dependencies. Steiner (1972) was one of the first to describe in detail which characteristics of the task structure must be seen as essential conditions for the provision of services in the social context of the group [106]. He has divided the degrees of task complexity and dependencies based on processing into four groups. From (1) additive (parallel and independent) to (2) sequential to (3) reciprocal (sequential with reverse direction) to (4) intensive (all directions). Without

planning the dependencies would be intensive, but with the *Program Board* they are structured and reduced to sequential.

Figure 4.6: Program board with dependencies between the teams at the end of the first day on the left and the second day on the right.

The final plan review and program risks meeting will give an overview about the status of each time. Based on this information and after the program risks have been addressed the teams vote in the *Confidence Vote* if the current PI plan will work out.
If one keeps in mind that many team members had no experience with agile development than overall the *PI Planning* worked very well for the case study partner. This is due to the good preparation of the meeting, the engagement of the team members, and constant help of an external consultant.

SAFe events and artifacts

Tables 4.1 and 4.2 show the artifacts and events that are used by the case study partner according to the definition of SAFe and the different levels. The percentage values show how many of the *Product Owners* and *Scrum Masters* answered in the first survey on the team level with "exists" or "not exists". It is clear that all teams use stories for the *Sprints* to plan the tasks, based on the *Team Backlog*. With the other artifacts at team level, the answers differ. *Iteration Goals* are being used by almost 50% of the teams. Interestingly, not all teams agree on enablers and *PI objectives*, whether they use them or not. On the program level the answers are different depending on the teams. All teams, except CRP, reported that *Features* are used. For the remaining artifacts, it is a mix of answers inside the teams if they exist or not. One reason could be that the wording is not clear to all team members. This is why some believe that the program has a *Program Backlog* and the others think that it is a *Solution Backlog* or *Portfolio Backlog*. Another area where the teams are not in agreement is whether there is a *Vision* or not. They answered either yes or no.

The same applies to the *non-functional requirements*, although there are other teams in this case.

Artifact	Level	Existing	Not Existing
Stories	Team Level	100%	0%
Enabler	Team Level	73%	27%
Iteration Goals	Team Level	47%	53%
Team Backlog	Team Level	100%	0%
Team PI Objectives	Team Level	60%	40%
Features	Program Level	93%	7%
Enabler	Program Level	67%	33%
Program Backlog	Program Level	67%	33%
Program Kanban	Program Level	47%	53%
PI Objectives	Program Level	67%	33%
Architectural Runway	Program Level	47%	53%
Continuous Delivery Pipeline	Program Level	67%	33%
Capabilities	Large Solution Level	27%	73%
Solution Backlog	Large Solution Level	60%	40%
Solution Kanban	Large Solution Level	33%	67%
Epics	Portfolio Level	73%	27%
Strategic themes	Portfolio Level	20%	80%
Portfolio Backlog	Portfolio Level	33%	67%
Nonfunctional Requirements	All Levels	40%	60%
Vision	All Levels	53%	47%

Table 4.1: Adopted SAFe artifacts

All teams agree that their events include: *Iteration Planning*, daily *Stand-up*, *Review*, and *Retrospective*. These are also the typical events organized by every Scrum team. The team responses differ in relation to the *Iteration Execution*, the *Backlog Refinement* and *Innovation, and Planning Iteration*. Here there are teams that have events and some that do not. At the program level, the teams have mostly the same opinion regarding which events exist. Only the answers for the *PO Synch* and the *Community of Practice* are different. For *Inspect and Adapt*, only the GPL thinks that this is part of the program compared to all other teams. Here different viewpoints between the management and the developer teams are to be expected.

Event	Level	Existing	Not Existing
Iteration Planning	Team Level	100%	0%
Iteration Execution	Team Level	67%	33%
Daily Stand-Up	Team Level	100%	0%
Iteration Review	Team Level	93%	7%
Iteration Retrospective	Team Level	93%	7%
Backlog Refinement	Team Level	80%	20%
Innovation and Planning Iteration	Team Level	73%	27%
PI Planning	Program Level	93%	7%
System Demo	Program Level	93%	7%
Scrum of Scrums	Program Level	100%	0%
PO Synch	Program Level	73%	27%
Community of Practice	Program Level	47%	53%
Inspect & Adapt	Program Level	7%	93%

Table 4.2: Adopted SAFe events

All in all many events and artifacts are used in the program of the case study partner. The fact that some artifacts from the *Large Solution* or *Portfolio* category are only partially used or not at all is due to the fact that *SAFe Essential* is used.

4.3.2 Team setup

To interpret the results of the survey correctly, it is important to know the distribution and the team structures. The focus lies on three influencing factors: (1) the team size, (2) the background of the team members in relation to the company affiliation and (3) the geographical distribution of the teams.

Team size

In total the 137 dev. teams are split in 14 teams. The teams are split based on business context which is very near to the bounded context. Some teams have up to two bounded contexts, but most of the teams focus on one. This is one reason why the team size has such a wide spread between the smallest team with 6 members (incl. SM and PO) and the biggest one with 28 members. The average team size is 9.8 which is a little higher than the proposed team size which should be around seven plus or minus two. As shown in Table 4.3 the spread presents with a standard deviation of 5 which is extremely high. This high number is caused by the CRE and the MDA teams. The CRE team would be big enough for three teams and the MDA team has twice the size of a proposed team size. This size was not historically established. At the beginning of the agile transformation there were eighteen teams, which were reduced to the current fourteen teams. There were several reasons for this. At the beginning there was a dedicated test team, which

of course was divided among the individual teams, so that there was an end-to-end responsibility within the teams. There was also a team for parameters and translations, which was also integrated, as both teams performed support tasks. But two other teams, that also tend to perform support tasks, remain in place. There is a team that provides a consistent UI and UI frameworks for other teams. There is also a system team that takes care of the infrastructure, databases, and the CI/CD pipeline. Both teams remained the same over the last six months. This is also true for the MDA team, which started with 16 team members. The team could also be described as an enabler team, as this team works on data modelling and data delivery for the other teams. The central knowledge in a team and the uniform data model are the reasons why the team was not integrated into the other teams. The increased coordination effort between the team for model development and the risk of mistakes, due to lack of overall knowledge, is too high a trade off compared to this team size.

Team	Developers	Tech-Lead	PO	SM	SUM (Members)
ACC	5		1	1	7
FRM	5	1	1	1	8
PST	7	1	1	1	10
ANA	3.5	0.5	1	1	6
CRE	24.5	1.5	1	1	28
CRP	6	1	1	1	9
SR	4		1	1	6
CRA	4	1	1	1.5	7.5
FWK	5	1	0.5	0.5	7
MDA	12	1	1	1	15
MET	6	1	1	0.5	8.5
REP	5	1	1	1	8
SYS	8	1	0.5	0.5	10
UIF	4	1	1	1	7
SUM:	99	12	13	13	137

Table 4.3: Team structure with the number of team members based on the role (March 2019)

On the other hand, the CRE team has grown extremely over this period. In the beginning, the team had only 13 team members, which is already a relatively large team. Nevertheless, the team expanded with resources from other teams, because this is a central module and is essential for the upcoming version which has to handle the large number of still open tasks. Here the trade off is accepted, which brings a large team with it. The big question is: how this team size effects the TWQ and if the investigations of Espinosa et al. (see Chapter 3.1) regarding Task familiarity vs. Team familiarity apply here.

Company affiliation

The second factor that could be relevant for the performance is the background of the team members focused on the company to which they belong. As described in Section 4.2 the company was founded and is owned by two IT companies, but with a completely different focus. As already described, the one owner (following: owner 1) comes from the software consulting and development sector, with activities in various industries. The other owner (following: owner 2) started from the operation of computer centers in the financial sector and in the development of banking software. Depending on the company of the employees different backgrounds arise, which can affect the team performance.

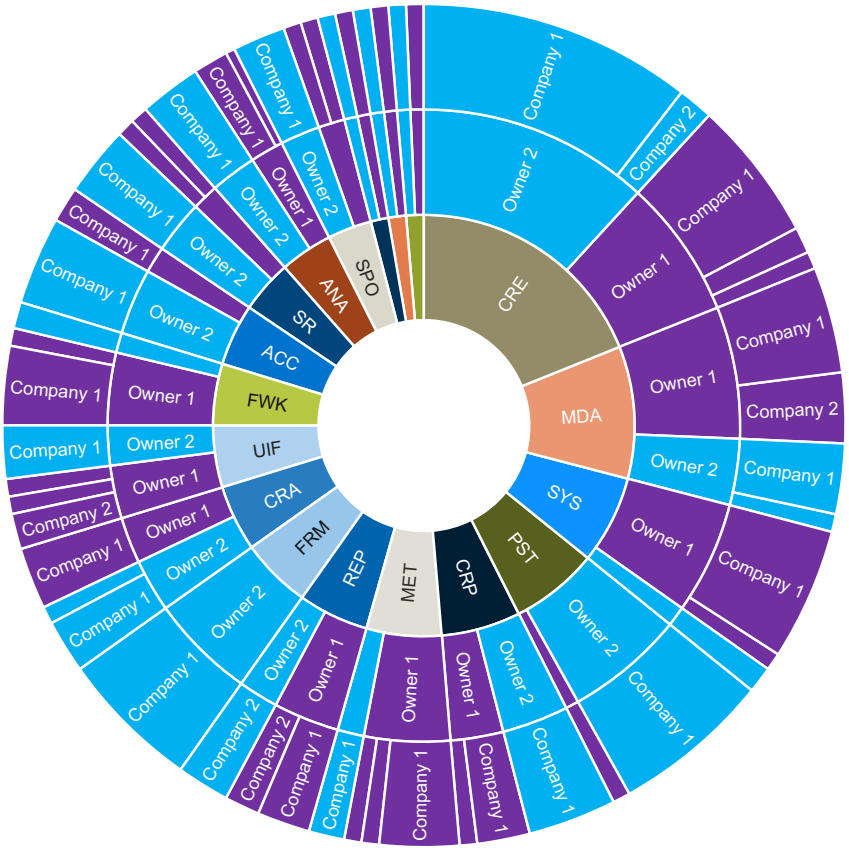


Figure 4.7: Sunburst diagram of all teams based on the number of members that are provided by the two owners (March 2019). The data is available in Table 10.2.

The sunburst diagram (see Figure 4.7) shows, in the inner circle, the teams with the number of team members signalling the size. The middle circle and the color of the circle represents how many team members come from the founding-company or owner one or two. The outer circle adds an additional layer of information. Each owner has multiple sub-companies with different focuses. The hypothesis in this case is based on the differences between the two owners. It is assumed that the employees of the first

owner have a different work flow and approach due to the consulting background as compared to the employees with a focus on operations. This approach is partly based on stereotypes, which always contain some truth. The question here is to analyze whether teams with a mix of both companies have a different team performance than teams from only one company. As shown in the sunburst diagram (see Figure 4.7) most of the teams have a team member mix of both owners, but some teams like the FRM, PST, SYS, and ACC mainly consist of members from one owner.

Geographical dispersion

The geographical dispersion is an additional factor that is relevant for the team performance according to Espinosa et al. (see Chapter 3.1) effecting the team familiarity and a MTS with a negative consequence for information exchange and synchronicity.

Figure 4.8: Sunburst diagram of all teams based and were the team members located (March 2019). The data is available in Table 10.1.

The sunburst diagram (see Figure 4.8) shows the geographical distribution of the teams. In the inner circle the teams are represented and the size represents the number of team members. The outer circle shows the different locations for each team. We assume

that the distance between the locations is irrelevant, since a small distance, for example between Munich and Ismaning (10km), has the same negative effect on communication and cooperation as a large distance. In both cases, it is impossible for the team members to communicate without an additional communication medium or even talk about other topics during the lunch. It can be assumed that the team members only sit together if the location is the same.

The locations with the highest number of team members are Münster with 64 members, Karlsruhe with 21 members, Ismaning with 19 members and Bretten and Frankfurt with each 14 members. The other seven locations have less than 10 team members. Only two teams (FRM and PST) have all team members at the same location. Other teams like the ACC, SR and CRA are mostly located at the same place. However, it is not helpful for the team spirit if all team members besides one are located in one place. Even if one team member can participate only remotely, the whole effort for a video conferencing and screen sharing is inevitable. The teams with the most distribution are the MDA team where the 15 team members are located at 8 different places (max. 4 at one location) and the UIF and SPO teams.

Team dependencies

The last relevant piece of information, besides the company affiliation and the geographical dispersion, is the dependency between the teams. The dependencies between the teams have a large impact on how they are integrated into the program. Figure 4.9 shows the dependencies between the teams in a graph which is based on the survey results (see Section 10.1 Q9). The graph is based on the team member perspective and not enhanced with additional data that also represents the team dependencies such as the dependency board or the code dependencies in the software. The teams: GPL, SPO, SA, and RTE teams are filtered out, because all of them have dependencies to all other teams and the number of connections makes the graph harder to read. The node's color shows the three clusters from Figure 4.1, the node's size is related to the number of indegree edges and the edge color and size represent how many of the survey participants of the team have selected the dependency divided by the number of participants of the team. What becomes immediately obvious in this graph is the large number of dependencies between all teams. The teams can also not be divided based on the three clusters. At no point there is a clear cut between the teams, as for example Maurya (2018) showed in his model (see Figure 2.2.4). This happens partly due to the architecture (details in Section 4.3.3). But if the number of indegrees is compared to the number of outdegrees (see Figure 10.2) for each node than the difference between the BAS cluster and the ACC and MEL clusters show a different image. The BAS cluster have a lot of indegree (node size in Figure 4.9) edges but also a lot of outdegree edges. The ACC and MEL clusters both have some indegree edges but a lot more outdegree edges. A good example is the FRM team hat have 6 indegee edges but also an outdegree edge to nearly all teams.

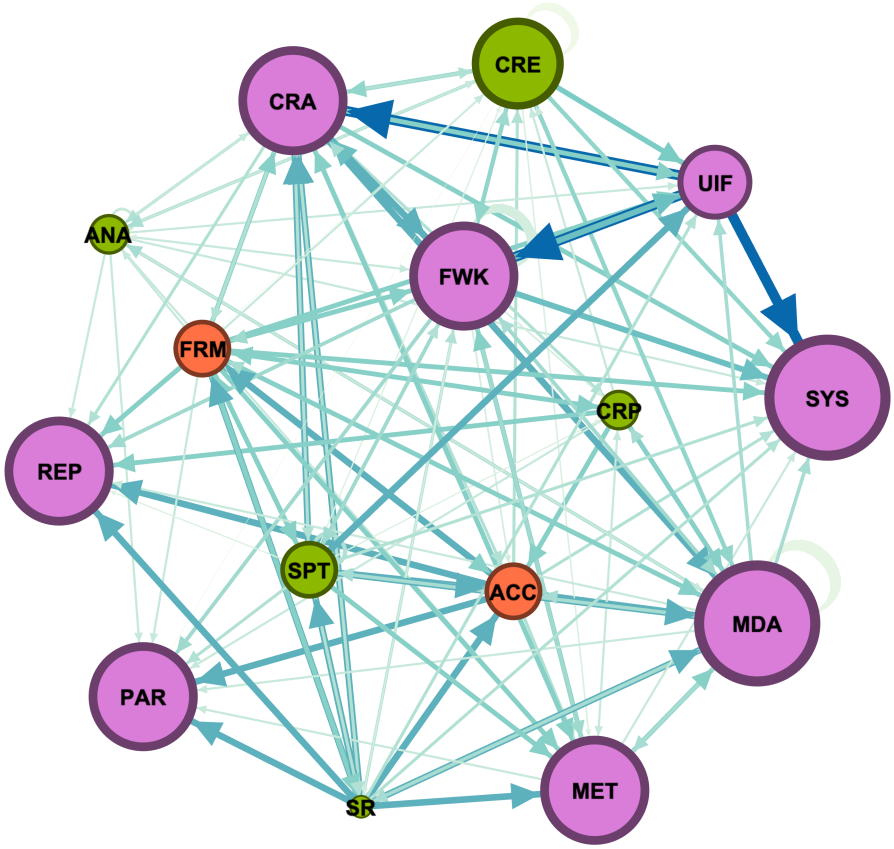


Figure 4.9: Team dependencies without the GPL and ART teams - program overview. Purple dots are BAS teams, green dots are MEL teams and orange dots are ACC teams.

The impact of dependencies to other teams on the TWQ is tested in Chapter 7 in hypotheses 7.

Other performance influence factors from the MTS research of Shuffler are shown in Figure 2.20 such as personal background, language barriers, time-zone, or cultural diversity cannot be fully applied in this case, since the teams in question are more homogeneous. They have similar background, live in Germany, and speak fluent German. Additional information about the survey participants is available in Section 5.1.1.

4.3.3 Architecture

Besides the people, the architecture of an application is essential for fast development with fewer conflicts. It is also the backbone that is set at the beginning, but will remain for the duration of the application life cycle. Bad software architecture can hardly be removed afterwards without big re-factoring efforts. This is the case for traditional development and also for agile development. Even if the idea of agile is that the current issues should be solved, future progress should be kept in mind which means that the developers and the architect team must work hand in hand to develop a stable and maintainable system. As shown in Figure 4.2 the architecture decisions are mainly taken by the two *System Architects* in a *Community of Practical* approach with the *Tech Leads* that represent their teams in a two weekly architecture meeting as SPoCs. In these meetings the current problems and interfaces are discussed so that the teams can communicate via *micro service architecture* with each other, but team internal architecture problems are also discussed so that they can receive advice from another perspective. The team tech leads are chosen by the teams and each team has one or two that are appointed. The decision is made based on the skills. A *Tech Lead* should have (1) development experience in the used environment, (2) DevOps knowledge, (3) experience with *Test Driven Development* (TDD) and automated testing and (4) the ability to work in a team and to communicate. However, all team members are allowed to get in touch with the *System Architect* for small questions and feedback if necessary.
If the *Tech Leads* were to be compared to the *System Architects*, one could say that the *Tech Leads* focus on their team and the external dependencies of their team. Whereas, the *System Architects* are also responsible for the overall system and the scaling of the system, because they are responsible for discussion of the long term road map with the test of the ART team. The second important focus lies on whether the *Architectural Principles* are met and developed. In Section 10.1.1 a table of architectural principles and the implementation status is shown. Nearly all of the requested principles are completely or partially implanted. The most important ones are the systems which is divided into modules that communicate over interfaces or the fact that each component has a clear owner. This is due to the fact that a completely new system is developed and experience and knowledge about the splitting of components and the infrastructure are based on the first developed version. For an architectural model they use *EAM Pattern Catalog* [67] to archive goals like a (1) co-develop domain model with business in a highly integrated fashion, (2) development of new components as libraries or services deployed in containers and (3) easy updates of single components (a detailed list is available in Section 10.1.1). The overall success of the architecture is tracked by the *System Architect* based on defined fitness functions for the most important topics. In the near feature they want to integrate an architecture help status integrated into jQAssistant by visualizing the latest connections between the components. The visualization should be automatically checked or tested against a defined set of rules. Baldrich et al. (2018) demonstrated an approach regarding how a test system can automatically check the architecture based on rules that are easily accessible, readable, and flexible for adjustments [4]. If they are really

readable and understandable for everyone they should exceed the current documentation that is done in a wiki system (Atlassian Confluence) based on *Unified Modeling Language* (UML) and *Architecture Decision Record* (ADR) where also the decisions are written down. The biggest challenges for the *System Architects* in this program are: (1) the massive unclear and volatile non-functional requirements that lead to many redesigns with wasted time which could be reduced had they been steady. One solution, in the future, is to use assumptions and document them, then use them as a base for discussions to be able to faster finalize a documented status that can be used by all teams to start working. (2) The skills and the experience of many developers in the program. This leads to two facts. The first one is that the system is based on the latest software technology (Spring, Docker, single page application with VueJS and PatternFly, ...) where some of the used frameworks are only one or two years old which automatically leads to less documentation and experience. The second factor is the background of the developers. The older colleagues may have worked with Cobalt where they head a completely different use case focus and environment. This can be compensated by investing some of the PI time in training for the developers. (3) A key feature of the software is to calculate financial data correctly based on input data based on the background laws and regulations. All these calculations could not be redeveloped that is why a legacy software is used to calculate some of the data. This legacy software is used at many code parts in different ways and not via an interface. To solve this problem the legacy software will be encapsulated behind an interface which is a more common way to communicate, such as with other components, and that there is only one point that must be adjusted in case of a change. (4) The system will be operated by one of the company owners that have high knowledge in this area. This is helpful in many cases, but a consequence is that they force many decisions that lead to problems in the architecture and could nowadays probably be solved in a more modern way. In addition, it has been observed that the data modeling should not be done by a single team that will provide the data to the rest of the teams. The model should be done by each team as collaborative work. Ideally, the developer of the model and the user should be the same person, which will ensure automatic understanding of what the use case, without having to interpret the requirements of the other teams. Another already implemented finding is, that the test capacities should be integrated into the teams. At the beginning of the program, the testers had their own team responsible for all tests and the automated testing. This caused the same problem as the data model team. Only having a high level overview with no further insight into the teams brings mistakes that could have been avoided.

5 Methodology

This chapter describes how the data that is used in Chapters 6 and 7 was collected. First, in Section 5.1, the survey design and the parts are explained. Afterwards, the response rate based on the team, roles, and other characteristics are shown.

5.1 Questionnaire design

This survey was a differentiated replication of the TWQ questionnaire from Hoegl and Gemuenden (2001) (see Section 2.3) which was already adapted by Dingsøyr in order to compare agile development to traditional development (see Section 3.1). The same questions are used as one part for this survey. With this case study it became necessary to make adjustments because not only the team performance, but also the program performance was analyzed. All questions had to be adjusted because the TWQ model is based on the team level. In order to remain true to the questionnaire, the only change to the questions was that the word "team" was replaced by the word "program" in most cases. In seven questions the word "team" had to be first added to make clear whether the question is related to the team or to the program. The full list with all questions is available in Section 10.1.1.
Figure 5.1 gives an overview concerning the survey structure and the goals the survey should answer. Since the original TWQ questionnaire consisted of 60 questions and these have been doubled, the survey was split into two surveys. The first survey is based on the team Level questionnaire. The second survey is the adaptation of the TWQ questions to program level. In the bottom half of Figure 5.1, one can find the survey components grouped into four areas. The first area deals with general information about the participant. The role is used for two differentiations: (1) How does the evaluation of performance differ between dev. team members, PO, SM, and program roles (e.g.: SPO)? Program roles have to complete the survey several times if they want to perform an evaluation for different teams. (2) Dev. team members are asked about all components (TWQ facets, personal success, team performance) of the TWQ survey. All non-dev. team members are only asked about the team performance, because they are not a direct part of the team. This approach was also followed by Dingsøyr. The experience and company of the participants were required in order to use them as latent variables in the data analysis, if the TWQ measurement model and outcome has a correlation to these factors. The share of participants based on the experience is shown in Section 5.1.1 to compare with the results from other studies. The second area is the main part of the survey. The TWQ questions are based on the original survey by Hoegl and Gemuenden (2001) (see

Section 2.3) for the survey on team level and with the adaption from team to program level for the second survey. The area concerning team information was required to define the participant's current team. With this information the team size and team distribution could be determined. The team dependency question was only part of the team survey to understand how the teams interact with each other and if this could be an outcome factor. The results are shown in Chapter 4.3.2. The last area was only part of the team level survey and meant to collect additional information about the program and how each team has adopted SAFe. The results of how the teams have evaluated the SAFe events and artifacts on team and program level are shown in Chapter 4.3.2.

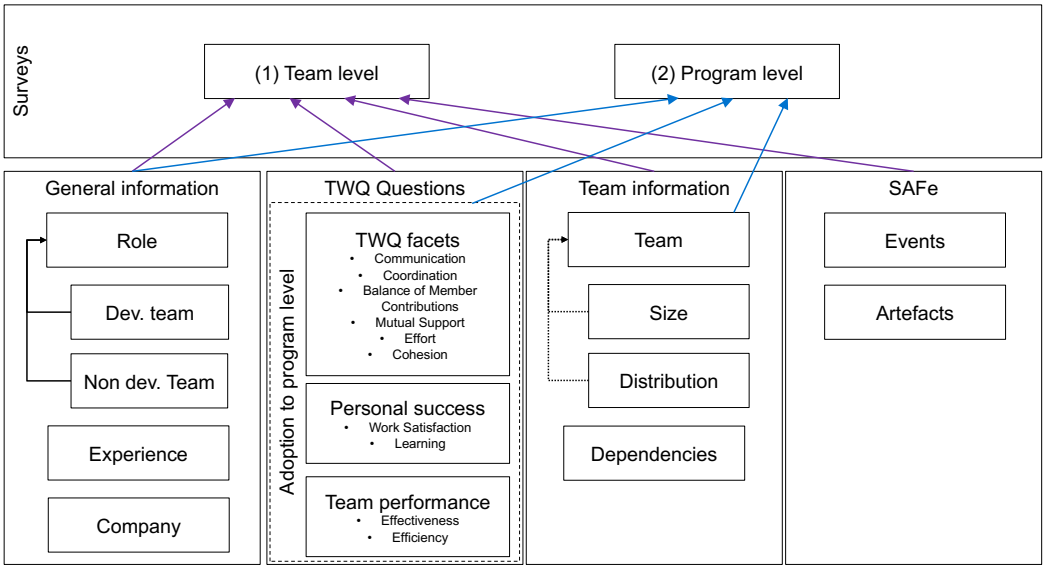


Figure 5.1: Survey overview structure

The surveys were created as an online questionnaire in the software QuestBack [3] and the participants were notified about the survey via e-mail.

5.1.1 Participants

Both survey studies included four groups of respondents: (1) dev. team members, (2) team leader = SM, (3) PO and (4) stakeholders. The maximum number of responses could be 148 if all program members participated. In total, 122 questionnaires were filled out; For the survey on team level 79 participants (54%) completed the full survey (details in Table 5.1).

[3] www.questback.com

Team	Team size	Dev. team	PO	SM	Stakeholders	SUM
GPL	2	0	0	0	1	**1**
RTE	2	1	0	0	2	**3**
SPO	5	0	0	0	1	**1**
SA	2	0	0	0	0	**0**
ACC	7	6	0	1	1	**8**
ANA	6	4	1	1	1	**7**
CRA	7.5	4	1	1	0	**6**
CRE	28	11	0	1	0	**12**
CRP	9	3	0	0	2	**5**
FRM	8	4	0	0	0	**4**
FWK	7	4	0	1	0	**5**
MDA	15	8	0	1	0	**9**
MET	8.5	2	1	1	0	**4**
PST	10	5	0	0	2	**7**
REP	8	0	0	0	0	**0**
SR	6	2	1	1	0	**4**
SYS	10	0	0	0	0	**0**
UIF	7	3	0	0	0	**3**
SUM	**148**	**57**	**4**	**8**	**10**	**79**

Table 5.1: Team level survey participants

On the program level the number was lower with 43 participants (29%) (details in Table 5.2). Participants who did not completely finish the questionnaire were excluded from the analyses. In total 21 team members, 6 SM, 7 PO, and 9 stakeholder responses were used in the analyses which results in a total of 43 valid data sets.

Team	Team size	Dev. team	PO	SM	Stakeholders	SUM
GPL	2	0	0	0	1	**1**
RTE	2	1	0	0	3	**4**
SPO	5	0	0	0	1	**1**
SA	2	0	0	0	0	**0**
ACC	7	2	0	1	0	**3**
ANA	6	2	1	0	1	**4**
CRA	7.5	0	1	1	0	**2**
CRE	28	3	0	0	0	**3**
CRP	9	2	0	1	1	**4**
FRM	8	2	0	1	0	**3**
FWK	7	1	2	0	0	**3**
MDA	15	1	0	0	0	**1**
MET	8.5	0	0	2	0	**2**
PST	10	1	2	0	2	**5**
REP	8	2	0	0	0	**2**
SR	6	3	1	0	0	**4**
SYS	10	0	0	0	0	**0**
UIF	7	1	0	0	0	**1**
SUM	**148**	**21**	**7**	**6**	**9**	**43**

Table 5.2: Program level survey participants

The characteristics of the sample population can be found in Table 5.3 which were required in the general information part of both surveys. This data was also analyzed if they had an impact on the TWQ and PWQ in Chapter 7. Questions about the role, experience, agile experience, and company were mandatory in the survey. The information of one's age and education were optional. The team size, geographical dispersion, and team dependencies were calculated based on the team with data from the case study partner or from the survey itself.

		Team survey		Program survey	
Characteristic	Category	N	Pct.	N	Pct.
Role	Quality Assurance	1	1%	1	2%
	Developer	55	70%	19	44%
	UX Designer	1	1%	1	2%
	Product Owner (PO)	8	10%	7	16%
	Scrum Master (SM)	4	5%	6	14%
	Product/Solution Manager	1	1%	2	5%
	RTE	2	3%	3	7%
	Business Owner	2	3%	1	2%

	GPL	0	0%	1	2%
	Other	5	6%	2	5%
Age	20 - 25 Years	6	8%	2	5%
	26 - 30 Years	3	4%	1	2%
	31 - 35 Years	15	19%	5	12%
	36 - 40 Years	15	19%	3	7%
	41 - 45 Years	5	6%	4	9%
	46 - 50 Years	12	15%	6	14%
	50 - 60 Years	10	13%	5	12%
	>60 Years	1	1%	0	0%
	-	12	15%	17	40%
Experience	1 – 2 Years	6	8%	4	9%
	3 – 5 Years	24	30%	9	21%
	6 – 10 Years	22	28%	12	28%
	11 – 15 Years	8	10%	6	14%
	16 – 20 Years	6	8%	3	7%
	>20 Years	13	16%	9	21%
Agile experience	1 – 2 Years	46	58%	27	63%
	3 – 5 Years	26	33%	13	30%
	6 – 10 Years	7	9%	3	7%
Company	Owner 1	26	33%	12	28%
	Owner 2	35	44%	23	53%
	Freelancer	8	10%	2	5%
	Direct	9	11%	5	12%
	Other	1	1%	1	2%
Education	A-level	3	4%	2	5%
	Bachelor / Diplom	31	39%	9	21%
	Master	25	32%	11	26%
	Professor / PhD	3	4%	1	2%
	Other	4	5%	2	5%
	-	13	16%	18	42%
Team size	2	4	5%	5	12%
	5	1	1%	1	2%
	6	11	14%	8	19%
	7	16	20%	7	16%
	7.5	6	8%	2	5%
	8	4	5%	5	12%
	8.5	4	5%	2	5%
	9	5	6%	4	9%
	10	7	9%	5	12%

	15	9	11%	1	2%
	28	12	15%	3	7%
Team locations	1	11	14%	8	19%
	2	18	23%	10	23%
	3	11	14%	10	23%
	4	15	19%	10	23%
	5	15	19%	4	9%
	8	9	11%	1	2%

Table 5.3: Characteristics of sample population for team and program level

5.1.2 Measures

The questionnaire was sent out after a careful development phase and a pre-test with non-program members. Adjustments were made to improve the clarity of the points. We used existing, validated scales from the literature to measure the factors. For all elements, a 5-point Likert scale from 1 (strongly different) to 5 (strongly consistent) was used. However, the scale was extended with an additional option of "don't know" based on the user feedback on the first day of the survey. Two participants mentioned that they were not able to finish the survey because they did not have enough information. A good example is the question Q64: "The team product proves to be stable in operation". The product is currently under development and has not been released yet, so it is not clear if it is stable of not. For the data analyses the "don't know" answers were filtered out. This means that the mean was calculated with 5 rather than 6 questions, if one of the questions is answered with "don't know". In total only fourteen questions were answered with "don't know" and in no case more than two answers are used from one participant so that the results should be still valid.

6 Data analysis and Processing

This chapter will describe how the collected survey results are analyzed and what techniques and tools are used to do that.

6.1 Modeling framework

The most frequently used approaches to data analysis, and the selected approach from the original paper, are *Principal Component Analysis* (PCA) and *Structural Equation Models* (SEM).
PCA is used to structure, simplify, and illustrate extensive data sets by approximating a large number of statistical variables with a smaller number of meaningful linear combinations. In the survey it is used to group all survey questions for the six TWQ facets and the performance factors to each one variable.
SEM is a statistical model that allows the estimation and testing of correlative relationships between dependent variables and independent variables, and the hidden structures between them. SEM is used to evaluate causal relationships between characteristics. The aim is to investigate hypothetical causal relationships. Similar to regression analysis, the question in which direction and to what extent one or more so-called exogenous variables influence one or more endogenous variables is examined. Similar to factor analysis, it is assumed that the characteristic of interest may not be observable, but rather represents the "latent" basis for the observed behaviour or the expression of opinion or attitude of the interviewee. The latter serves as indicators for the expression of the underlying basic characteristics.
First of all, the basis for path analysis is a hypothesis that states: which characteristics are influenced by variables. Complex contextual structures can be established in this way, whereby characteristics can be simultaneously independent (i.e. influencing) and influenced by others. Path analysis tests this hypothesis by measuring the degree of influence and determining the degree of information of such a model.

The SEM consists of the four elements:

Indicator (item) These are observed variables. Usually the model recommends the use of at least four indicators. For example, an indicator for "intelligence" is the "final grade in the graduation". In Figure 6.1 the indicators of the latent exogenous variable are the six TWQ facets (communication, coordination, balance of member contribution, mutual support, effort, cohesion) on the left side and the indicators of the latent endogenous variables are the work satisfaction, learning, effectiveness,

and efficiency on the right side. The latent exogenous variables ξ1 will be used in Chapter 7 to evaluate what other factors have a direct influence on the TWQ.

Latent variable (factor) The unobserved variable is measured only by its indicators. In Figure 6.1 the latent exogenous variable is the TWQ (for program level: PWQ) and the latent endogenous variables are the team members' success and performance variables.

Measurement model The model is based on a confirmatory factor analysis and models connections between the indicators and the latent variables. In Figure 6.1 the measurement model of exogenous latent variables are the six TWQ facets and the TWQ variable and the measurement model of endogenous latent variables are the team members' success and performance variables with the work satisfaction, learning, effectiveness and efficiency as indicators.

Structural model This is the set of exogenous and endogenous variables and their connections. In Figure 6.1 the TWQ, team members' success, and performance are combined with the structural model.

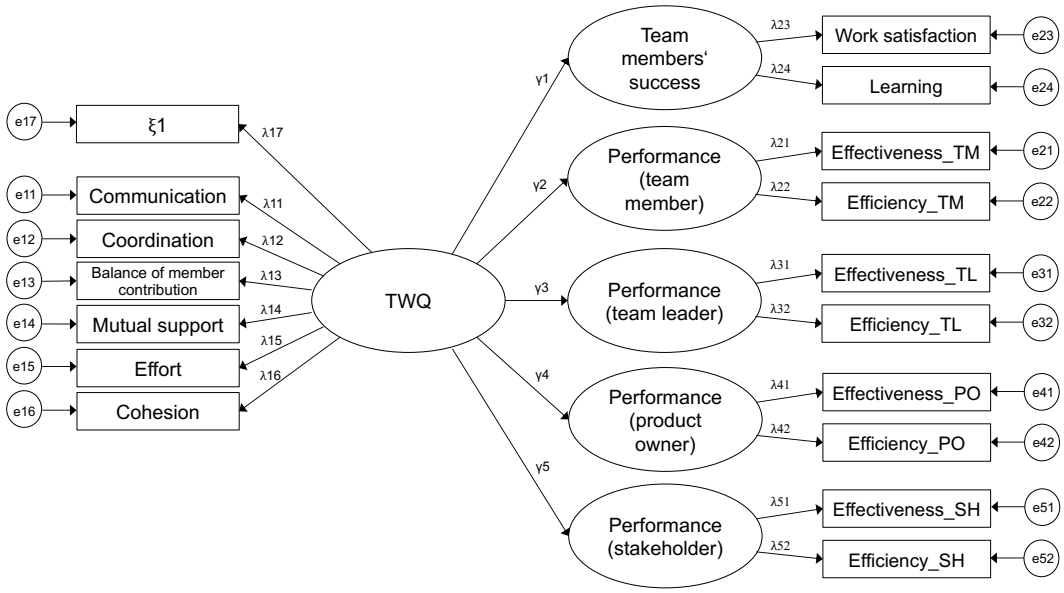


Figure 6.1: SEM model causal model (without residual variables)

6.2 Data analysis

The data analysis was conducted with mainly two tolls: the basic analysis, filtering, and the mapping of answers are completed in Excel; the SEM analysis was conducted in

SmartPLS [4] The data was analyzed according to the seven steps of the defined approach of Backhaus et al. (2008) [3]. (1) Theoretical foundation and hypothesis formation: Logical theoretical considerations about the cause-effect relationships of the relevant variables. The hypotheses are formed according to these theoretical preliminary considerations. Then the hypothesized constructs are connected to each other by means of a hypothesis system [63]. (2) Method selection: The choice of method depends on the type of examination and the research objective. (3) Model formulation: The theoretical considerations are transformed into a linear system of equations. In order to reduce the complexity, the hypothetical correlations are graphically represented by means of a path system. (4) Empirical elevation: To collect empirical data on the basis of which a solution to linear SEM can later be found. (5) Parameter estimation (with the PLS procedure): First, a factor analytical estimation of the factor loads (lambda coefficients) of the exogenous and endogenous measurement model is performed as well as a calculation of the respective factor values. The factor values are then used to perform a regression analysis in which the exogenous variables form the independent variables and the endogenous variables form the dependent variables. (6) Assessment of the estimation results: Then the model structure is checked to determine how well it adapts to the empirical data. Various quality criteria are available for evaluating the estimated results. (7) Modification of the model structure: In all methodological efforts and attempts to achieve a good model fit, the falsification of hypotheses is often even of higher value for theoretical development than the elimination of parameters, because it challenges the theory-building process new. In this context, particular attention should be paid to hypothesis testing and methodology. Following this approach, the survey data is exported and for each participant the mean for each of the six TWQ facets (communication, coordination, balance of member contribution, mutual support, effort, cohesion) and for the work satisfaction, learning, effectiveness, and efficiency are calculated. The reverse coded questions are turned to match the scale for the mean. To check for unengaged responses three techniques are used: (1) check the time to fill out the survey, (2) check the reverse coded items compared to the other answers and (3) the standard deviation is proved to see if participants always answered with the same answer for each question. None of the respondents must be excluded based on these analyses. The results are filtered overall and for each team combined with additional information such as the team name and potential latent exogenous variables (size, age, ...) into a CSV file that was imported into SmartPLS. The missing values for factors are excluded in the analysis. This must be done, because only the development team members fill out the complete survey and the SM and PO could only answer the performance questions. In SmartPLS the data is analyzed with the consistent PLS (PLSc) algorithm. The algorithm performs a correction of reflective constructs' correlations to make results consistent with a factor-model. Factor analyses were performed to determine whether all six TWQ factors refer to the same latent construct. A non-rotational *Principal Component Analysis* (PCA) was performed at the team level using aggregated

[4] Ringle et al. (2015). SmartPLS 3 Boenningstedt: SmartPLS GmbH, `http://www.smartpls.com` [87], [77].

responses from team members. To prove the correctness of the model a model fit is done based on Hu and Bentler (1999) [60] and Browne and Cudeck (1993) [13] analyses is done for both models and the sub models where the validity and reliability (R-square, P-value) of the model is checked.

To compare the different influence factors to the model a multi-group analysis (MGA) (Byrne et al. (2006) [16], and Floh (2006) [46]) is performed on the data, even if the number of data-sets decreased by splitting. With MGA the data-set is split by grouping variable (such as age) and than the model is tested with each data-set. Without the MGA the model can answer if the age has an influence on the TWQ but not if this differs depending on older to younger developers and if other factors are more or less relevant for them.

6.3 Data overview

The survey data results are processed as descried and based on this the following results and KPIs for the data are generated. The data is only shown in this section but not analyzed. The analyzes based on the research questions is done in Chapter 7. First the model fit describes how the integrity of the data and the model are checked and how reliable the data are.

6.3.1 Model fit

For each presented model the following metrics are checked to see if the model is valid. The model fit refers to how well the proposed model correlation is between variables in the data-set. The following thresholds listed determine goodness of fit (by Hu and Bentler (1999) [60]): SRMR should be < 0.09; NFI should be above 0.9; p-value for the model should be > 0.05. Additionally to these metrics SmartPLS calculates reliability and validity metrics for each latent variable: (1) Cronbach's alpha that gives the ratio of observed variance to the variance of test values and is therefore a measure of internal consistency and should be above 0.7, (2) average variance extracted (ave) is a measure of the quality of how a single latent variable explains its indicators and should be above 0.5 and the composite reliability that checks the factor loadings and uniqueness from a factor analysis and should be above 0.7. If the goodness fit measures and the metrics for latent variables are fine the indicators connection to the latent variables are checked by using the t-statistics. It is the ratio of the deviation of the estimated value of a parameter from its hypothetical value to its error rate (higher is better: 99% is 2.58, 95% is 1.96 and 90% is 1.65). Only measurement models are used where the t-statistics is above 1.65. The second measure is for collinearity statistics by the inner and outer *Variance Inflation Factor* (VIF) that represent the ratio of variance in a model with multiple terms, divided by the variance of a model with one term alone where the goal is to be as low as possible. Lower than 3 is a good VIF. It is a measure of the factor by which the variance of a parameter estimator increases when multicollinearity is present.

6.3.2 Team level data

Table 6.1 shows the descriptive statistics for the 16 variables that are used to measure TWQ based on the survey results for the SAFe team members' success, team performance by the team members, team leaders (SM), PO and stakeholder. Each variable is represented as the arithmetic mean, the median and standard deviation of the individual items that comprise the variable. To have an indication of the data quality the reliability factors excess kurtosis (measure of the "tailedness"), skewness (measure of the asymmetry around the mean), outer VIF and Alpha (Cronbach's alpha) are also shown and highlighted in green if they fulfill the criteria and orange if they don't. The kurtosis and skewness are not as expected between -1 and 1 for the performance of the team leader and PO based on the number of participants (see Table 5.2).

	Mean	Median	S.D.	Kurtosis	Skewness	VIF	Alpha
Communication	3.803	3.800	0.554	-0.770	-0.114	2.650	0.764
Coordination	3.733	3.750	0.759	-0.312	-0.155	3.545	0.727
Mutual Support	4.364	4.429	0.527	1.028	-1.091	3.048	0.852
Effort	3.667	3.625	0.747	-0.553	0.139	3.978	0.786
Cohesion	3.706	3.600	0.640	-0.356	0.163	4.340	0.857
Contribution	3.716	3.667	0.731	-0.907	0.081	3.341	0.516
Team member - success							
Work Satisfaction	4.135	4.250	0.666	-0.009	-0.615	3.547	0.908
Learning	4.004	4.000	0.727	0.173	0.773	3.547	0.756
Team member - performance							
Effectiveness	3.794	3.800	0.609	-0.586	-0.274	2.664	0.884
Efficiency	3.294	3.200	0.782	-0.264	0.322	2.664	0.894
Team leader - performance							
Effectiveness	3.945	4.000	0.509	-1.523	-0.222	2.405	0.802
Efficiency	4.000	4.200	0.743	4.410	-1.782	2.405	0.780
PO - performance							
Effectiveness	3.667	3.500	0.544	0.980	1.056	4.339	0.852
Efficiency	3.400	4.000	0.993	-1.235	-1.680	4.339	0.923
Stakeholder - performance							
Effectiveness	4.167	4.300	0.435	-0.327	-0.924	3.633	0.824
Efficiency	4.233	4.400	0.522	-0.465	-0.250	3.633	0.704

Table 6.1: Means. medians. standard deviations. and reliability factors (Excess Kurtosis. Skewness. Outer VIF. Alpha (Cronbach's alpha)) of the variables at the team level

Tables 6.2 and 6.3 show reliability factors and validity metrics for the data. All factors are inside the expected range.

	Saturated model	Estimated model
SRMR	0.064	0.136
NFI	0.708	0.648

Table 6.2: Model fit metrics with the SRMR and NFI on team level

	Alpha	AVE	Composite reliability
TWQ	0.931	0.745	0.946
TM Success	0.927	0.932	0.965
TM Performance	0.840	0.860	0.925
TL Performance	0.888	0.895	0.944
PO Performance	0.811	0.838	0.912
SH Performance	0.924	0.889	0.941

Table 6.3: Construct reliability and validity metrics with the Alpha (Cronbach's alpha), average variance extracted (AVE) and Composite reliability on team level

To compare the impacts depending on the available criteria for the MGA the following spitting is used based on the team setup and the number of teams, team members and survey participants. Table 6.4 shows the list of the case study dev. teams and into which group these are classified for each influence factor. The following factors are analyzed in Chapter 7: (1) team size, (2) team geographically dispersion based on the number of (2.1) locations and how many team members are (2.2) located together, (3) company affiliation based on the two company owners, (4) work experience in software development, (5) agile work experience and (6) number of dependencies to other teams. The table also show cross dependencies that must be taken into account in the analyzes. This is for example the case between inexperienced in agile work experience and small teams.

Team	Team-size	Locations	Distribution	Owner	Agile experience	Dependencies
ACC	Small	Some	Dis. <3/3	Mostly Owner 2	2-4 years	8-11 dep.
ANA	Small	Some	Dis. <=2/3	Half/Half	2-4 years	up to 8 dep.
CRA	Medium	Some	Dis. <=2/3	Half/Half	>4 years	more than 11 dep.
CRE	Large	Many	Dis. <=2/3	Half/Half	2-4 years	8-11 dep.
CRP	Medium	Many	Dis. <=2/3	Half/Half	<=2 years	up to 8 dep.
FRM	Medium	One	Dis. =1	Mostly Owner 2	<=2 years	8-11 dep.
FWK	Small	Many	Dis. <=1/3	Mostly Owner 1	>4 years	more than 11 dep.
MDA	Large	Many	Dis. <=1/3	Half/Half	2-4 years	more than 11 dep.
MET	Medium	Many	Dis. <=1/3	Mostly Owner 1	2-4 years	more than 11 dep.
PST	Large	One	Dis. =1	Mostly Owner 2	2-4 years	8-11 dep.
REP	Medium	Some	Dis. <=2/3	Half/Half	-	more than 11 dep.
SR	Small	Some	Dis. <=2/3	Mostly Owner 2	<=2 years	up to 8 dep.
SYS	Large	Some	Dis. <=2/3	Mostly Owner 1	-	more than 11 dep.
UIF	Small	Many	Dis. <=1/3	Half/Half	2-4 years	8-11 dep.

Table 6.4: List of dev. teams and in what groups/clusters they are assigned for the MGA. For details about the groups see: (1) size in Table 4.3, (2) locations and (3) distribution in Figure 4.8 and Table 10.1, (4) owner in Figure 4.7 and Table 10.2, (5) agile experience in Table 5.3, and (6) dependencies Figure 4.9 and Figure 10.2.

6.3.3 Program level data

Table 6.5 shows the descriptive statistics for the 16 variables on the program level survey. The reliability factors are also shown as in Table 6.6 on program level. For the developers all metrics are acceptable so that the results can be statistically used even if some of the VIF metrics are a bit off. As for the team level the number of team leader and PO participants (see Table 5.2) is a little bit low that is why the metrics for the performance for both groups have only moderate reliability metrics.

	Mean	Median	S.D.	Kurtosis	Skewness	VIF	Alpha
Communication	3.317	3.000	0.609	-0.343	-0.182	4.402	0.778
Coordination	3.492	3.500	0.743	-0.490	-0.608	1.977	0.760
Mutual Support	3.448	3.000	0.758	-0.093	0.301	2.070	0.851
Effort	3.400	3.500	0.838	-1.052	0.004	6.140	0.879
Cohesion	3.317	3.000	0.609	-0.343	-0.182	3.113	0.872
Contribution	3.350	3.000	0.732	0.311	0.287	2.647	0.470
Team member - success							
Work Satisfaction	3.550	3.500	0.776	-0.094	-0.239	2.651	0.964
Learning	3.533	3.500	0.991	0.130	-0.395	2.651	0.839
Team member - performance							
Effectiveness	3.217	3.500	0.873	-1.007	-0.001	3.395	0.942
Efficiency	3.067	3.000	0.987	-0.491	-0.420	3.395	0.915
Team leader - performance							
Effectiveness	3.083	3.000	0.607	1.335	-0.440	3.841	0.826
Efficiency	3.333	4.000	0.745	-0.300	-0.857	3.841	0.866
PO - performance							
Effectiveness	2.714	3.000	0.749	4.447	-1.968	7.857	0.920
Efficiency	2.714	3.000	0.700	7.000	-2.646	7.857	0.876
Stakeholder - performance							
Effectiveness	3.444	3.000	0.926	-0.963	0.197	3.233	0.924
Efficiency	3.000	3.000	1.155	-0.286	0.000	3.233	0.938

Table 6.5: Means, medians, standard deviations, and reliability factors of the variables at the program level

Tables 6.6 and 6.7 show reliability factors and validity metrics for the data. As for the team level also the metrics on program level are inside the expected range. Only the AVE and composite reliability indicator for the stakeholder performance is outside the expected range, but in an acceptable range.

	Saturated Model	Estimated Model
SRMR	0.090	0.091
NFI	0.658	0.654

Table 6.6: Model fit metrics with the SRMR and NFI on program level

	Alpha	AVE	Composite reliability
PWQ	0.912	0.697	0.932
TM Success	0.882	0.893	0.943
TM Performance	0.913	0.920	0.958
TL Performance	0.925	0.930	0.964
PO Performance	0.966	0.967	0.983
SH Performance	0.908	0.222	0.280

Table 6.7: Construct reliability and validity metrics with the Alpha (Cronbach's alpha), average variance extracted (AVE) and Composite reliability on program level

6.3.4 SEM model and factor loadings

Table 6.8 shows a compression of the factor loads of the surveys that are done by Hoegl and Gemuenden (2001) [59], Lindsjørn et al. (2016) [74] and the results based on the survey on team and program level in the case study. Similar ladings are highlighted with a background color. The full SEM model for all four datasets are shown in Figures: (1) Figure 2.16 by Hoegl and Gemuenden, (2) Figure 3.1 by Lindsjørn, Yngve, et al., (3) Figure 7.1 on team level and (4) Figure 7.2 on program level. This shows that the results are comparable with the other two survey results. Even if they are little bit of. Interestingly, the factor loadings are more equal to the Hoegl and Gemuenden results that are done in traditional project environments. In comparison to the Lindsjørn et al. results that compared them in agile environment. But when it comes to the factor loads for the PO and stakeholder the results match the agile results.

	Hoegl and Gemuenden	Lindsjørn et al.	TWQ	PWQ
Communication	0.88	0.84	0.82	0.84
Coordination	0.71	0.47	0.89	0.76
Mutual Support	0.89	0.87	0.84	0.80
Effort	0.82	0.74	0.87	0.91
Cohesion	0.89	0.90	0.92	0.87
Contribution	0.89	0.73	0.85	0.82
Team member				
Success	0.93	1.00	0.79	0.79
Work Satisfaction	0.94	0.90	0.96	0.93
Learning	0.69	0.81	0.97	0.96
Team member				
Performance	0.64	0.68	0.62	0.80
Effectiveness	0.86	0.96	0.95	0.96
Efficiency	0.74	0.75	0.91	0.96
Team leader				
Performance	0.34	0.32	0.22	0.28
Effectiveness	0.74	0.90	0.97	0.97
Efficiency	0.79	0.70	0.92	0.96
PO				
Performance	0.26	0.06	0.15	0.05
Effectiveness	0.80	1.04	0.94	0.98
Efficiency	0.76	0.64	0.89	0.99
Stakeholder (compared to PO results)				
Performance	-	-	0.10	0.10
Effectiveness	-	-	1.00	0.12
Efficiency	-	-	0.88	0.68

Table 6.8: Factor loadings comparison between Hoegl and Gemuenden [59], Lindsjørn et al. [74], TWQ and PWQ based on the survey. The background color show similar loadings between the resultes.

7 Evaluation and results

This chapter combines the related work influence factors on the team and program performance from Chapter 3 and then translates them into hypotheses that will be answered by using the survey results from Chapter 6.

7.1 Is the TWQ model applicable

Before other additional influence factors can be evaluated it must be confirmed that the overall TWQ model is applicable for the case study. This is done on on team level to prove that the existing research results are also applicable for large-scale agile development and that the results are comparable. The second part is to prove that the six facets and the correlation to team performance and personal success from the TWQ model can be also applied to the whole program.

Hypothesis 1 (H1): The TWQ model of Hoegl and Gemuenden is applicable for assessing Team Work Quality in a large-scale agile development program

To test H1 the survey results (see Figure 7.1) on team level are compared with the existing results from Hoegl and Gemuenden (2001) [59] and Lindsjørn et al. (2016) [74] that are shown in Figure 2.16 and Figure 3.1. Table 6.8 shows the factor loadings of all three models to compare them more easily. The factor loadings of TWQ in the three surveys are highly similar. But the facets loadings are more comparable with the Hoegl and Gemuenden results that are done in traditional software projects as compared to the results from Lindsjørn et al. that are done in an agile environment. Between both the largest difference is that the data from the agile survey has a lower loading for coordination (0.47) than the data from the traditional survey (0.71). For the case study partner the loading is now even higher with (0.88). One reason for this may be that the program and the dependencies create more complexity, which becomes relevant for TWQ. The second finding is that the the coefficients in the agile survey are higher for team members' success and performance by the team members. The loadings for the team leader (PO) are nearly equal for all three surveys. But the performance loadings for the POs (0.08) and stakeholders (0.06) are as low as for the agile survey by Lindsjørn et al. (0.06). So the role of the PO and stakeholder in the SAFe program for each team is more comparable with a Scrum team where the team is also in charge for the results and the performance.

Overall the loadings are comparable and this means that the TWQ model is also applicable for team in a large-scale agile program.

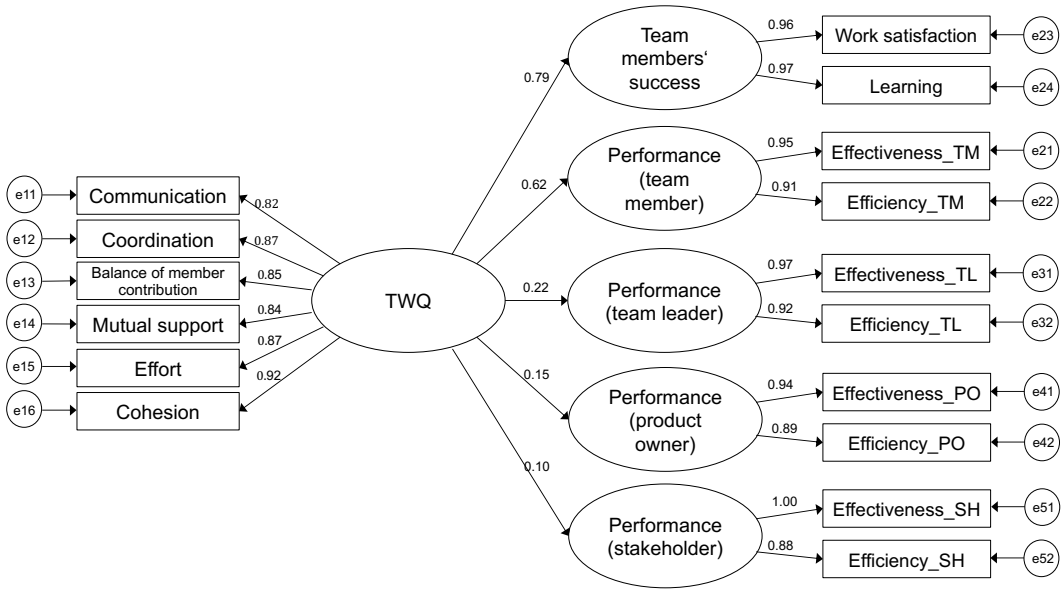


Figure 7.1: Standardized factor loadings and path coefficients for the survey results on team level.

Hypothesis 2 (H2): The model of Hoegl and Gemuenden is applicable for assessing Program Work Quality in a large-scale agile development program

Compared to Hypothesis 1, Hypothesis 2 tests if the TWQ model with the same six facets, the team members' success and the performance can be also applied on the program level. As there are currently no research results available this is done by the results from the adapted TWQ survey. As Table 5.2 shows the number of participants on for the program survey is slightly low, which is nevertheless sufficient (see Table 6.5) for the results to be used. To test H2 the survey results (see Figure 7.2) are compared to the factor loadings of the TWQ model from Hoegl and Gemuenden [59] and Lindsjørn et al. [74]. The factor loadings of the surveys are shown in Table 6.8 and they also highly similar compared to each other and also compared to the TWQ results from the case study. All six facets have a loading higher than 0.70 and the only real difference is the same as for the TWQ case study results compared to the agile environment (0.47) that the coordination is as high (0.72) as for the traditional survey (0.71). Again, the reason may be that the coordination effort for a SAFe program became more important and relevant than for a Scrum team. In contrast, the performance loadings are more similar to agile environments. Although the performance of the team members is very high (0.90) compared to all other results with less than 0.70, the performance of the team leaders (PO) with 0.23 is lower than the other results with more than 0.30.

All in all, the differences are all within the range and it can be confirmed, based on the results, that the TWQ model is applicable not only for teams (H1) within the program, but also for the whole program (H2).

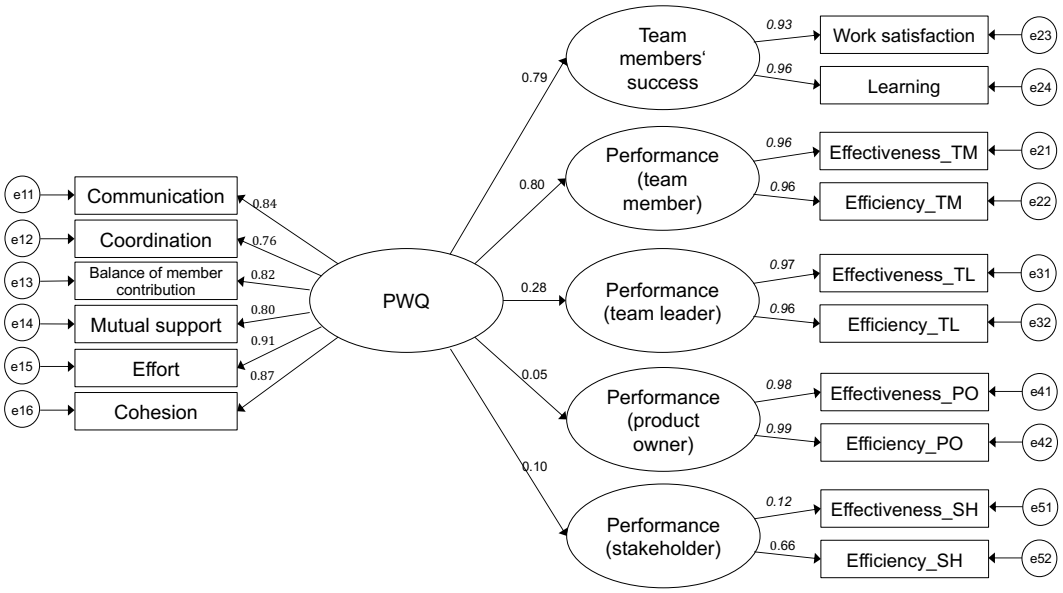


Figure 7.2: Standardized factor loadings and path coefficients for the survey results on program level.

7.2 Additional influencing factors and their impact on TWQ

The next hypotheses will use the collected data of the survey and evaluate whether additional factors have an influence on the performance of the team and how they interfere with the existing factors from the TWQ model.

Using the MGA with SEM, one can test if the proposed relationships are moderated by other influence factors. Bootstrapping offers an efficient test of indirect effects (Preacher, Rucker and Hayes (2007) [84]) as well as a direct test between two or three discrete values of a moderator. The clustering of the teams is done based on the available teams (see Table 4.3) and where the biggest difference is between the results, based on the mean where at least two teams are part of the cluster with valid results.

Hypothesis 3 (H3): Team size has an impact on TWQ

Studies have been conducted (see Figures 2.5, 3.2 and 2.20) to evaluate the correlation between team size and team outcome. These studies come to the same result: the larger the team, the higher the process loss resulting in the productivity of the team sinking. The same results are visible in Figure 7.3 where the mean of the team answers are split in three groups based on the definition of Table 7.1.

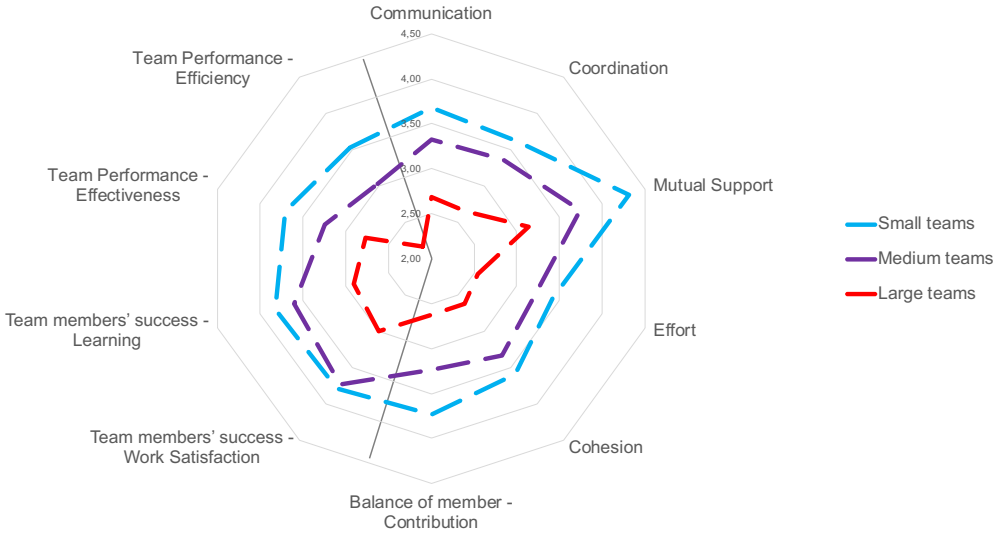


Figure 7.3: Mean results of the team survey on team level grouped by the team size.

Cluster	Condition	Survey participants	Teams	Team member
Small	<= 7	27	5	33
Medium	>7 and <= 9	19	5	41
Large	>9	28	4	63

Table 7.1: Team size cluster definition for MGA with the number of participants, teams and team members for each group.

All factor loadings are shown in Table 7.2. the clusters differ in the following way:

Communication is a bit higher for small (0.85) and medium (0.84) teams than the overall results (0.82), but for large teams the factor loading is 0.55 lower (-0.27).

Coordination has similar results compared to communication. The small (0.91) and medium (0.96) teams are almost equal. But both are higher than the overall factor (0.89) loading. The large teams are lower with 0.70 which is -0.19 less.

Effort differs between all three sizes. The large (0.90) teams nearly match the overall factor loading with 0.87, but the small teams are 0.76 lower, especially compared to the medium teams with 0.99, the highest factor loading overall on effort.

Team members‘ success fluctuate between 0.66 for the medium size teams and 0.88 for the small teams. The large teams (0.71) and the overall (0.79) are in between.

Performance (team member) is nearly the same for the small (0.67) and the large (0.71) teams compared to the overall (0.62) factor loading. However, the loading is

extremely low for the medium sized teams with only 0.20 which means that it has no impact for this group on the TWQ.

	Overall	Small	Medium	Large
Communication	0.82	0.85	0.84	0.55
Coordination	0.89	0.91	0.96	0.70
Mutual support	0.84	0.87	0.82	0.81
Effort	0.87	0.76	0.99	0.90
Cohesion	0.92	0.95	0.93	0.89
Contribution	0.85	0.84	0.85	0.82
Success (TM)	0.79	0.88	0.66	0.71
Performance (TM)	0.62	0.67	0.20	0.71

Table 7.2: Comparison of factor loadings based on the team size.

To sumarize the results, the data shows that smaller teams overall have a better TWQ and outcome on all factors compared to medium and large teams. The medium teams are between the large and the small teams. Interestingly, the factor loadings are especially different in communication and coordination. Other than expected, the loading for coordination is lower for the largest teams than for the small and medium teams. One would expect the opposite behaviour, when the communication and coordination effort would have a greater influence, the bigger the team and the program is. The study from Lindsjørn et al. (2018) compares the TWQ between small and large programs and how the correlation between the factors changes. They show that the program size has no impact on the communication and coordination [73]. In the case of a bigger program one could assume a larger influence in the first moment. The second major deviation is the influence of team members' performance on the TWQ for medium-sized teams. In this case the value is so low that it can be ignored and the team performance has no influence. There is no clear explanation why there is no dependency.

Hypothesis 3 confirms the investigations from various research areas in terms of TWQ and team size. For example, Hoegl (2005) reported that larger teams often suffer from poor TWQ [58], and Gratton and Erickson (2007) reported that when the team size grows to over 20 members, collaboration between members tends to decline [51]. With team size increasing, it becomes increasingly difficult to coordinate interactions between members because of the inherent complexities in interactions (Bradner et al. (2005) [12]).

Hypothesis 4 (H4): Team distribution has an impact on TWQ

As described by Espinosa et al. (2007) (see Section 3.1), distribution of teams creates challenges in the program that can effect the TWQ. To consider the influence, the teams are divided based on the number of locations and the team distribution factor [5] according to the criteria in Table 7.3 (details about the team distribution in Figure 4.8). Figure 7.4 shows the mean results of the survey answers for the teams. The first thing to observe is that the teams working together in one place have very good results in all areas. This is no surprise as it confirms the results from the TWQ and MTS research. In contrast, in the middle of the diagram one can see the teams that are partially distributed have an average score of 1.2 below compared to the teams that work together at one place. This result confirms the scientific findings. Some surprising results are the results of the teams that work in multiple places. The average scores are very good and even exceed the results of the teams in one place.

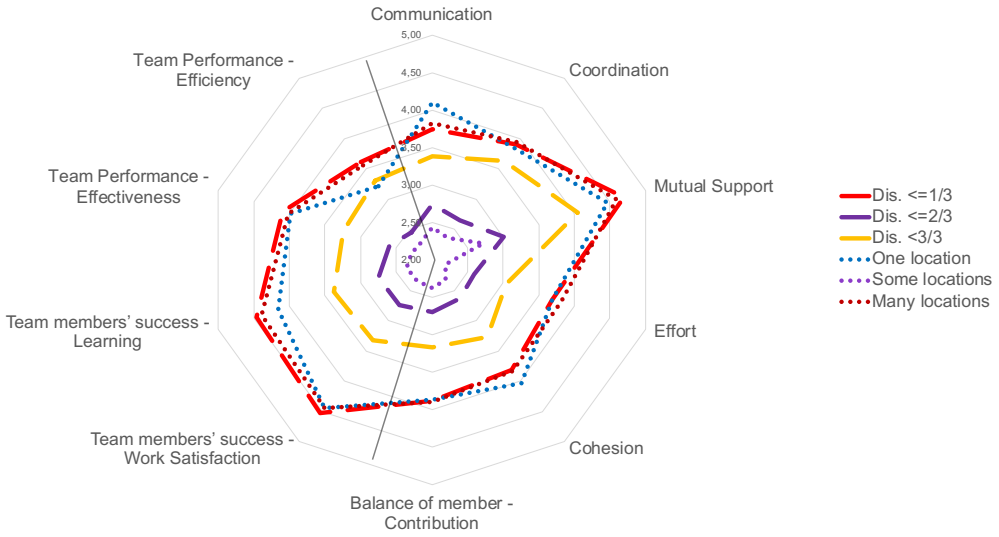


Figure 7.4: Mean results of the team survey on team level grouped by the distribution.

To validate the result, not only is the number of locations taken into account, but also the team distribution factor.

[5] Number of team members working together at one place. This takes into account how many locations there are and how many team members work in relation to the team size per location. Examples: 10 team members at one location = 100%; 10 team members at 10 locations = 10%; 10 team members with 5 persons each at two locations = 50%; 10 team members with the following breakdown: 2,2,2,4 = 28%.

Cluster	Condition	Survey participants	Teams	Team member
One	1 location	11	2	18
Some	<= 3 locations	25	6	44.5
Many	>3 locations	38	6	74.5
Dis. <=1/3	Team distr. <=1/3	21	4	37.5
Dis. <=2/3	Team distr. >1/3 and <=2/3	35	7	74.5
Dis. <3/3	Team distr. >2/3 and <1	7	1	7

Table 7.3: Team distribution cluster definition for MGA with the number of participants, teams and team members for each group.

This also confirms the result. All factor loadings are shown in Table 7.4. the clusters differ in the following way:

Effort has a similar behavior as coordination in the other direction. The teams at one location (0.95), teams with >2/3 members at one location (0.94) and very distributed teams (0.90) are higher than the overall factor loading with 0.87, but the teams that are partly distributed are lower with 0.80.

Contribution factor loading is for teams at mostly one place higher with 0.96 compared to overall (0.85), partly distributed (0.84) and very distributed (0.84).

Team members‘ success has the same result for teams at one place (0.76) and very distributed teams (0.74) compared to the overall factor loading (0.75), but the partly distributed teams have a high factor loading, 0.89, in comparison.

Performance (team member) has a lower factor loading for partly distributed teams with 0.50. In contrast the very distributed teams (0.77) and the mostly at one located teams (0.76) are higher than the overall factor loading with 0.62.

	Overall	>2/3	Some	Many
Communication	0.82	0.88	0.87	0.81
Coordination	0.89	0.90	0.93	0.82
Mutual support	0.84	0.84	0.84	0.83
Effort	0.87	0.94	0.80	0.90
Cohesion	0.92	0.96	0.95	0.92
Contribution	0.85	0.96	0.84	0.84
Success (TM)	0.79	0.82	0.90	0.73
Performance (TM)	0.62	0.76	0.50	0.77

Table 7.4: Comparison of factor loadings based on the team distribution.

The evaluations confirm hypothesis 4. The distribution of teams has an impact on TWQ in most areas. What is particularly interesting is the result that teams in one place and very distributed teams produce very good results. An explanation why completely distributed teams may have such a good result is that digital communication tools and digital documentation are used as common practice. Whereas, communication problems can occur with partially distributed teams, as the majority of the team is located in one place and analog meetings can be held. Obviously, the remote team members are forgotten, which leads to gaps in knowledge. Unfortunately this correlation was not found in any related work. However, this observation was confirmed in the interviews with the RTEs and is also visible in the *Jira* boards of the teams. There are more comments when the teams are completely distributed. Despite the large differences that can be seen between the groups in the survey results in all categories, the factor loadings on the TWQ differ only marginally. The biggest difference is in the performance and here the difference of the two extreme values is only 0.27, and the overall loading is exactly in between.

Hypothesis 5 (H5): Professional experience of team members has an impact on TWQ

The hypothesis can be divided into two parts, since the experience in both software development (Q2) and agile development (Q3) is a part of the survey. Unfortunately, the hypothesis cannot be tested because there are problems with the available data for both parts. The test, which is based on the average experience in software development, has a very similar average result for nearly all teams (sum of average experience divided by the number of team members). Even if one divides the teams by the areas of the survey to observe a wider spread, there are not enough differences to divide the available teams into clusters which can be used for an MGA analysis. The agile experience test has enough different data sets that can be used for evaluation. The mean results are shown in Figure 7.5, based on the breakdown in Table 7.5. The first impression may, however, be somewhat confusing. If one were to interpret the results without reference, then the teams with the lowest (<=2 years) agile experience have the best results and are even slightly better than the teams with the most (>4 years) agile experience. Only teams with two to four years of agile experience are outperformed. However, if one looks at the teams in the context of the other available data in Table 6.4, it becomes immediately clear that both groups: <=2 and >4 agile experience are all teams with the team size: small.

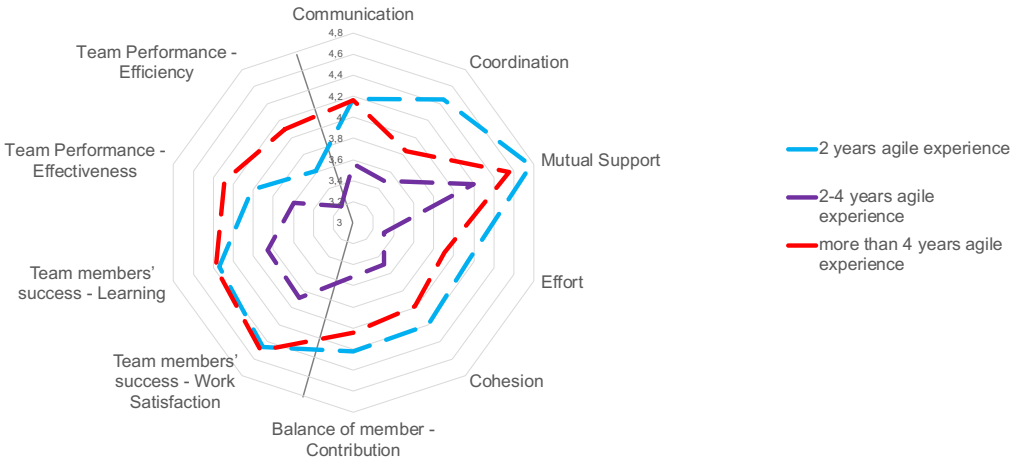


Figure 7.5: Mean results of the team survey on team level grouped by the agile experience if the the team member average.

Cluster	Condition	Survey participants	Teams	Team member
2 years agile experience	Avg. <=2	13	3	23
2-4 years agile experience	>2 and <=4	50	7	81.5
more than 4 years agile exp.	>4	11	2	14.5

Table 7.5: Team agile experience cluster definition for MGA with the number of participants, teams and team members for each group.

Therefore, it is not possible to test if there is a direct correlation between experience in software development and TWQ. In the case of the correlation between agile experience and TWQ, one can even say that there is no correlation, since otherwise the agile experience would have effected the results of hypothesis 3.

Hypothesis H6: Different company affiliations of team members have an impact on TWQ

A specific test based on a possible influence factor for the case study partner is that the company is owned and populated by the employees of the two parent companies that have a different background (see Section 4.3 and Figure 4.7). So this test, in comparing the background, mindset and motivation structure (mentioned in Sections 2.4.1 and 2.2.4) of the teams is based on the assumption that team members from owner one have a consulting company background and team members from owner two have a background based on operation of computer centers. But the teams that include the team members

from owner one have less experience in software development with, on average, 3-5 years compared to the 11-15 years of the employees coming from owner two. However, Figure 7.6 shows directly that, as with team size and distribution, there is an impact here. The observed teams from the clusters are also equally distributed among the clusters of the other hypotheses that there are no cross-dependencies that falsify the results. Based on the data, it is clear that the teams perform better the greater the proportion of team members provided by owner one.

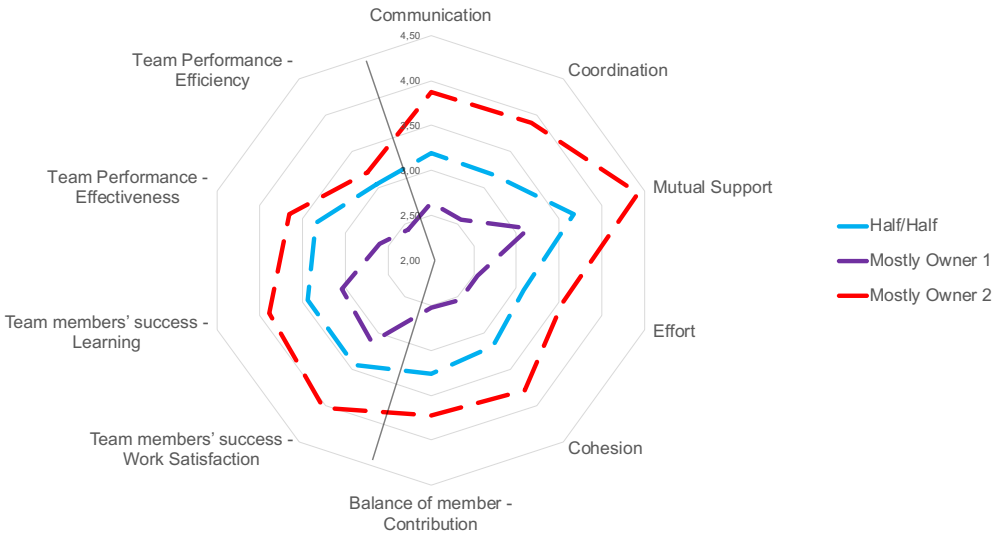


Figure 7.6: Mean results of the team survey on team level grouped by the companies the people come from.

Cluster	Condition	Survey participants	Teams	Team member
Half/Half	<=2/3 of one owner	42	7	80.5
Owner 1	>=2/3 TM of owner 1	9	3	25.5
Owner 2	>=2/3 TM of owner 2	23	4	31

Table 7.6: Team owner cluster definition for MGA with the number of participants, teams and team members (TM) for each group.

The impact from the company affiliation to all factor loadings is shown in Table 7.7. Clusters differ in the following way:

Mutual support has an overall score of 0.84 and the teams originating from mostly owner two (0.81) and teams with mixed members (0.87) have a nearly similar factor loading. In comparison, the teams originating from owner one have a low factor loading with 0.59.

Team members' success show a similar result as mutual support. The factor loadings overall (0.79), owner two (0.78) and half/half (0.82) are very similar. Only the factor loading for teams that mostly come from owner one have a very low loading with 0.47 that is 0.32 lower than the overall result.

	Overall	Owner 1	Half/Half	Owner 2
Communication	0.82	0.77	0.82	0.87
Coordination	0.89	0.92	0.90	0.91
Mutual support	0.84	0.59	0.87	0.81
Effort	0.87	0.88	0.87	0.94
Cohesion	0.92	0.99	0.92	0.95
Contribution	0.85	0.82	0.84	0.95
Success (TM)	0.79	0.47	0.82	0.78
Performance (TM)	0.62	0.63	0.66	0.62

Table 7.7: Comparison of factor loadings based on the team members and which share is provided by which owner.

The hypothesis can therefore be confirmed. There is visible influence of company affiliation in this case study partner which has the direct and linear impact on team performance and TWQ. However, the exact reason for the better performance cannot be exactly proved. The team members of owner one have more professional experience and come from the industrial background. This certainly has an influence on their performance, but the team members of owner two have a mindset that is more in line with the agile approach. These findings were also confirmed during the interviews. However, no interview partner wanted to confirm the correlation between performance and company affiliation. The personal background may also lead to a different interpretation of the survey scale. Based on the available data, the performance ratings correspond to the ratings of the SM, PO and stakeholders. The loadings are very similar between all groups and factors. Only the teams originating from owner one are less dependent on mutual support and personal success of the team members. Both factors have a personal character which might be attributed to the background and mindset of the team members.

Hypothesis H7: The number of dependencies to other teams in the program have an impact on TWQ

Dependencies to other teams cannot be avoided in programs, since the entire program has the same goal and the teams have mutual dependencies. However, the program can do something to limit the effect so that the dependencies do not result in too much influence: (1) split the teams as effectively as possible in order to make dependencies between teams as low as possible (example: Maurya (2018) [80]), (2) plan dependencies between teams as well as in *PI Planning*, and (3) split dependencies between teams as

fairly as possible so that not all teams are dependent on a team that is a bottleneck. The case study partner does seem to have successfully managed these points, since the mean value of the dependencies is 10.4, min is 7, max is 13 and the standard deviation is 2.0. This is reflected in Table 7.8, which shows that there are no large ranges that distinguish the teams in comparable sizes. The interviews show that this results mainly to the use of bounded contexts, which are also visible in Figures 4.9 and 10.2. Figure 7.7 shows the mean values based on the clusters. It is immediately noticeable that all three clusters overlap almost completely and that there are no large differences.

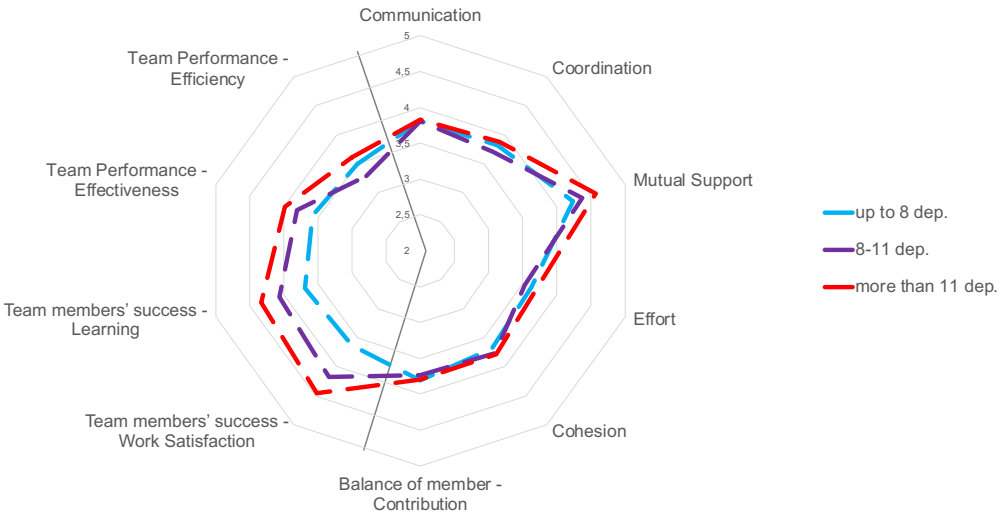


Figure 7.7: Mean results of the team survey on team level grouped by the number of dependencies of the teams.

Cluster	Condition	Survey participants	Teams	Team member
up to 8 dep.	<=8 dependencies	16	3	21
8-11 dep.	>8 and <=11 dep.	34	5	60
more than 11 dep.	>11 dependencies	24	6	56

Table 7.8: Team dependencies cluster definition for MGA with the number of participants, teams, and team members (TM) for each group.

The factor loadings differ completely between clusters. All loadings for teams with less than 8 dependencies, have a loading higher than 0.9 and closer to 1.0 except the loading for performance with 0.77. The loadings between the overall result and the two clusters between 8 and 11 dependencies and with more than 11 dependencies are very similar. All three clusters only differ in effort (overall = 0.87, >8 and <=11 = 0.74, >11 = 0.94). And the value for performance of 0.28 is very low for the cluster with more than 11 dependencies.

	Overall	<=8	>8 and <=11	>11
Communication	0.82	0.98	0.81	0.79
Coordination	0.89	0.99	0.76	0.82
Mutual support	0.84	0.95	0.82	0.74
Effort	0.87	0.98	0.74	0.94
Cohesion	0.92	0.98	0.90	0.95
Contribution	0.85	0.93	0.87	0.83
Success (TM)	0.79	0.97	0.75	0.68
Performance (TM)	0.62	0.77	0.65	0.28

Table 7.9: Comparison of factor loadings based on the number of team dependencies.

This hypothesis cannot be confirmed because the deviations are not visible. Nevertheless, the hypothesis cannot be rejected either, because the range of dependencies between the teams is very low. There are no teams in the dataset which, for example, only have two dependencies or teams with 20 dependencies. Only with a larger spread of values can the hypothesis can be refuted.

Summary of the evaluation results from H3 to H7

The available data confirm hypotheses 3, 4 and 6. Hypothesis 3 confirms that team size has a direct impact on TWQ and performance. Hypothesis 4 confirms that there is a direct correlation between the geographical distribution of teams and performance and TWQ. Teams in one central location perform best, although teams with a complete distribution achieve similar results. Only the teams that are partly located in one central location and the rest of the team in different locations perform worse. For hypotheses 5 and 7, there is not enough data available. However, this does not refute the hypotheses, which can neither be confirmed nor refuted.

Hypotheses	Test	Result
H3	Team size has an impact on TWQ	passed
H4	Team distribution has an impact on TWQ	passed
H5	Professional experience of team members has an impacton TWQ	not enough data
H6	Different company affiliations of team members have animpact on TWQ	passed
H7	The number of dependencies to other teams in the program have an impact on TWQ	not enough different data

Table 7.10: Overview of the hypotheses results

8 Discussion

This chapter summarizes key findings of the master thesis and shows the limitations of the thesis to allow the reader to interpret the results.

8.1 Key findings

A large-scale agile program can be seen as a MTS

A large-scale agile SAFe program matches the defined characteristics of an MTS. This allows the already existing scientific findings of the social sciences to be applied to SAFe programs. There are interesting findings on coordination and leadership and other influencing factors, which have also been confirmed on the basis of the hypotheses. The challenges and success criteria of MTS and large-scale agile development overlap.

A big bang approach for the transformation from traditional development models to large-scale agile development is feasible

The case study partner has proved that it is possible to move from traditional development to large-scale agile development with almost 150 people in one project. The preparation of the transformation is a very important part. All team members have to be trained on the new model. Even if this binds resources, the investment will prove to be worth the effort in time. Naturally, it is helpful to have an additional external coach to have all questions regarding processes and procedures answered in the most efficient manner.

The TWQ model is applicable on a large-scale agile program for each team and for the whole program

The same factors from the TWQ model effect not only traditional and agile teams, they also effect large-scale agile teams. And this applies to not only each individual team but also to the entire program. The influence factors for the teams are even more comparable with a traditional project setup than the influences of a single agile team, because several teams lead to an increased coordination effort. In contrast, the influence of POs and other stakeholders is very low, as with small agile teams.

The smaller the teams the more efficient they are, even if the number of dependencies increase, due to the higher number of teams

The correlation between team size and performance is not surprising. In large-scaled agile programs, a team behaves just like any other team. This effect can also be seen in teams that are already large according to Scrum. The performance between teams up to 7 members, 8 to 9 members and more than 9 members becomes around 10% worse the bigger the teams become. Between teams with 10 members compared to teams with 15 or even 28 members the performance decreases further, even if this percentage is not as large. The team size is always a trade-off between small teams, which need a SM and PO as well as add more dependencies to other teams which also effect performance, and larger teams with less overhead and less dependencies. But hypothesis 7 has shown that there is at least no direct loss in team performance if the number of dependencies increase slightly (range 7 to 13).

The team should work completely distributed if it has to be geographically distributed

If it is not possible for resource reasons that all team members work together as a team at one location, one should take care not to place a larger part of the team at one location and distribute the rest of the team at other locations. This only leads to the formation of a subteam at one location with the other team members not receiving all the information. This leads directly to a worse team performance. When the team needs to be split up, all team members should be split up, so the team is always forced to communicate electronically, which in turn allows all team members to receive the same information. This makes the performance comparable to teams at a single location.

8.2 Limitations

The mixed methods of the presented research design, described in Section 1.3, lead to limitations in all three used methods and the overall limitation of the thesis. The first limitation is the limited time frame of the thesis due to the time frame of only nine months when all relevant evaluations are done. The second limitation is that the influence of the results of the case study and the survey is that it is a single-case study with one partner in Germany. Because of this, the results are very specific to that company with side effects that are probably not mentioned because the impact was not clear and there was no other program to which to compare the results. To mitigate the risk of false results, the survey questions are based and compared to the results of Hoegl and Gemuenden, and Dingsøyr if they have similar results. As for the surveys of Hoegl and Gemuenden, and Dingsøyr, the biggest limitation is the number of responses. All surveys are limited to the number of participants between 100 and 300 which is very low for a SEM analysis. More details about the survey results and the response rate are described in Chapter 5. For the case study exploration and interviews, the potential threats are reduced by the approach

of Runeson and Höst (2009) [90]. If different understandings were raised, then more stakeholders are asked to validate the statement. The literature review that is compiled for challenges and success factors for large-scale agile transformation is limited to the query that is used with the limitation that the main keywords "Performance" or "Success factor" or "Challenge" must be in the tile. No other synonyms are used and based on the number of search results, this is the limitation that had the highest impact on the query. Changing from "Success factor" to "Success" has tripled the number of results. Another limitation is due to the fact that large-scale agile development is quite a new research area and that is why only results after 2010 are taken into account. For the literature review about TWQ and MTS, only a forward and backward search was conducted to find relevant results in Google Scholar. This limits the results regarding two factors: (1) The results must be publicly available and listed in Scholar and (2) the original papers or a reference must be correctly referenced.

9 Conclusion and future work

The last chapter of this thesis presents a short summary of the results that are evaluated based on the research objectives. The second part presents an outlook for further work.

9.1 Summary

This section features conclusions made based on conducted research. The four research questions asked in Section 1.2 will be answered for that purpose.

> **Research question 1:** What positive and negative influence factors exist on the team and program level in literature and which can be mapped to an agile environment?

The literature review summarizes what has already been researched about agile and large-scale agile development and the success factors in it. Based on other research in this area, there are a number of influencing factors with positive and negative effects on the project and teams. The same influences effect both the team (intra-team) and the program level (inter-team), since individual agile teams also have external dependencies. The likelihood increases only for large-scale agile projects due to external influences such as conflicts and dependencies to other teams. The majority of the studies show the same picture: that organization, communication, coordination, leadership, and mindset have an influence on the project. Additionally there are also studies that focus on a specific aspect, such as team split. Chow & Cao (2008) have grouped these factors into five main groups that provide a good overview: organization, people, process, technical, and project factors [20].

> **Research question 2:** Can the multi-team system (MTS) concept be applied for large-scale agile programs in order to adapt the existing research in this area?

The comparison of the theory of large-scale agile development (in particular SAFe) with the characteristics of MTS show that the two concepts overlap. There are small differences in the leadership structure, which is rather rigid in MTS, and the duration of the MTS, which is limited to the period of achievement of a common goal. However, these criteria can be met by a large-scale agile project. A comparison of the existing MTS literature shows that it can also be applied to a large-scale agile project. The same influencing factors from RQ1 exist not only in the agile environment, but also in MTS studies. In

addition, there are further detailed MTS studies related to the dependency between different leadership styles and MTS performance, or the conflict between team goals and MTS goals, which can also be applied to large-scale agile projects.

> **Research question 3:** Can the teamwork quality (TWQ) model be applied by the case study partner on the team level and are there any additional significant factors that can be added?

The TWQ model is applicable for this case study partner and delivers very similar results compared to the existing results of Hoegl and Gemuenden in the traditional development environment and Dingsøyr in agile environments. The result for the SAFe program is a mix of both results. The coordination is relevant, as in traditional development, but the influence of managers and stakeholders behaves more similarly from the agile environment (not significant). Based on the data and the number of teams, the influence of team size, geographical distribution and company affiliation could be confirmed. The influence of the experience in software development and the agile environment and the number of inter team dependencies could not be confirmed or rejected due to lack of significant data.

> **Research question 4:** Can the TWQ model be applied from the team level to the program level?

The adapted TWQ survey on program level confirms, for this case study partner, that the same six facets, performance and personal success can also be applied to a program. It is important to note that this is only one program and therefore only an indication is delivered, since several programs are required to offer direct proof.

9.2 Future work

The three areas which are considered in the master thesis, can be further investigated in detail in further studies.

The first area is to use the analysis of MTS findings to evaluate their impact on SAFe programs and performance. There are hundreds of studies and models from the military area that focus on leadership and coordination and from the civilian area that focus on communications and team set-up.

Hypothesis 1 confirmed that the same factors of the TWQ model can be applied to the entire SAFe program of the case study partner. For this purpose, the TWQ questions were adapted to the program. Until now, however, this result has only been confirmed for this one case study partner. It is necessary to verify the results by conducting the same survey in other large-scaled agile projects. If these other studies also show a similar result, then other influencing factors such as the number of teams or how different languages effect the project can also be analyzed. The comparison of the team results can also be used, since there is a correlation between the team and the program. For example: How does

the number of teams effect the team and program? If it increases the performance of the teams for the team tasks and the TWQ, does it have, at the same time, a negative effect on the program success and the PWQ (see MTS Section 2.21)?

Finally, further influence factors on the performance and TWQ of the participating teams are compared with hypotheses 3 to 7. These hypotheses should be verified in other programs. The hypotheses that cannot be tested because of the data availability might also be confirmed or refuted with more data from other programs. In addition to the five hypotheses, there are other influence factors from the MTS and large-scale agile areas that can be analyzed. To name just a few examples: the number of team changes, the length of collaboration in these teams, the influence of different languages, and many more.

These findings allow the constantly growing number of large-scale agile programs to be further optimized.

10 Appendix

10.1 Survey questionnaire

The following survey questions were used in the online survey at the case study partner to get the results about the TWQ and the program-work quality.

General information

Q1 **Which role description applies to you? (mandatory)**
Welche Rollenbezeichnung trifft auf Sie zu?

- ◯ Gesamtprojektleitung (GPL)
- ◯ Release Train Engineer (RTE)
- ◯ System/Solution Architect/Engineer
- ◯ Product/Solution Manager
- ◯ Business Owner
- ◯ Product Owner (PO)
- ◯ Scrum Master (SM)
- ◯ Entwickler // Developer
- ◯ UX Designer
- ◯ Qualitätskontrolle // Quality Assurance
- ◯ Betrieb Hosting // Operation Hosting
- [a] Anderer / Other

Q2 **How many years have you been working in the field of software development? (mandatory)**
Seit wie vielen Jahren sind Sie im Bereich der Softwareentwicklung taïg?

- ◯ 1 – 2 Jahre / Years
- ◯ 3 – 5 Jahre / Years
- ◯ 6 – 10 Jahre / Years
- ◯ 11 – 15 Jahre / Years
- ◯ 16 – 20 Jahre / Years
- ◯ > 20 Jahre / Years

Q3 **How many years have you been working in the field of agile software development? (mandatory)**
Seit wie vielen Jahren sind Sie im Bereich der agilen Softwareentwicklung taïg?
- ◯ 1 – 2 Jahre / Years
- ◯ 3 – 5 Jahre / Years
- ◯ 6 – 10 Jahre / Years
- ◯ 11 – 15 Jahre / Years
- ◯ 16 – 20 Jahre / Years
- ◯ > 20 Jahre / Years

Q4 **Associated company? (mandatory)**
Zugehöriges Unternehmen?
- ◯ Owner 1 / Company 1
- ◯ Owner 1 / Company 2
- ◯ Owner 1 / Company 3
- ◯ Owner 1 / Company 4
- ◯ Owner 2 / Company 1
- ◯ Owner 2 / Company 2

Q5 **How old are you? (optional)**
Wie Alt sind Sie?
- ◯ 20 – 25 Jahre / Years
- ◯ 26 – 30 Jahre / Years
- ◯ 31 – 35 Jahre / Years
- ◯ 36 – 40 Jahre / Years
- ◯ 41 – 45 Jahre / Years
- ◯ 46 – 50 Jahre / Years
- ◯ 50 – 60 Jahre / Years
- ◯ > 60 Jahre / Years

Q6 **Education? (optional)**
Ausbildung?
- ◯ Abitur – A-level
- ◯ Bachelor – Diplom – Fachausbildung
- ◯ Master – Magister
- ◯ Doktor – Professor – PhD
- [a] Anderer – Other

Q7 **Name and e-mail address for further questions. (optional)**
Name und E-Mail Adresse für Rückfragen .

Team information

Q8 **Team member in? (currently) (mandatory)**
Teammitgleid in? (zum aktuellen Zeitpunkt)

- ◯ GPL
- ◯ RTE
- ◯ SPO
- ◯ SA
- ◯ ACC
- ◯ ANA
- ◯ CRA
- ◯ CRE
- ◯ CRP
- ◯ FRM
- ◯ FWK
- ◯ MDA
- ◯ MET
- ◯ PAR
- ◯ PST
- ◯ REP
- ◯ SR
- ◯ SYS
- ◯ UIF
- ◯ RTE
- ◯ SPO
- ◯ SA

Q9 **To which teams does your team have dependencies? (multiselect)**
Zu welchen Teams hat ihr Team Abhängigkeiten?

- ☐ ACC
- ☐ ANA
- ☐ CRA
- ☐ CRE
- ☐ CRP
- ☐ FRM
- ☐ FWK
- ☐ MDA
- ☐ MET
- ☐ PAR
- ☐ PST
- ☐ REP
- ☐ SR
- ☐ SYS
- ☐ UIF
- ☐ RTE
- ☐ SPO
- ☐ SA

10.1.1 Questions based on the TWQ model

The following questions are based on the questions that used by Hoegl and Gemuenden (2001) [59] for the TWQ model and the same scaling is used. Each of the questions was mandatory and Likert scaled (five-point answer scale). However, the scale was extended with an additional option for "don't know" based on the user feedback on the first day of the survey. Two participants mentioned that they were not able to finish the survey because they have nor information. A good example is the question Q64: "The team product proves to be stable in operation". The product is currently under development and not released yet, so it is not clear if it is stable of not.

Scale:
◯ Strongly disagree // trifft nicht zu
◯ Disagree // trifft eher nicht zu
◯ Neither agree nor disagree // teils-teils
◯ Agree // trifft eher zu
◯ Strongly agree // trifft zu
◯ (added) Don't know // kann ich nicht beurteilen

Communication

Q10	Team	**There is frequent communication within the team** Es findet eine regelmäßige Kommunikation innerhalb des Teams statt
	Program	**There is frequent communication within the program** Es findet eine regelmäßige Kommunikation innerhalb des Programms statt
Q11	Team	**The team members communicate often in spontaneous meetings, phone conversations, etc.** Die Teammitglieder kommunizieren oft in spontanen Meetings, Telefonaten, etc.
	Program	**The program members communicate often in spontaneous meetings, phone conversations, etc.** Die Programmmitglieder kommunizieren oft in spontanen Meetings, Telefonaten, etc.
Q12	Team	**The team members communicate mostly directly and personally with each other** Die Teammitglieder kommunizieren meist direkt und persönlich miteinander
	Program	**The program members communicate mostly directly and personally with each other** Die Programmmitglieder kommunizieren meist direkt und persönlich miteinander
Q13	Team	**There are mediators in the team communication through whom much communication is conducted** Es gibt Mediatoren in der Team Kommunikation, durch diese die Kommunikation begleitet wird
	Program	**There are mediators in the program communication through whom much communication is conducted** Es gibt Mediatoren in der Programm Kommunikation, durch diese die Kommunikation begleitet wird

Q14	Team	**Relevant ideas and information relating to the teamwork is shared openly by all team members** Relevante Ideen und Informationen zur Teamarbeit werden von allen Teammitgliedern offen ausgetauscht
	Program	**Relevant ideas and information relating to the teamwork in the programm is shared openly by all program members** Relevante Ideen und Informationen zur Teamarbeit im Programm werden von allen Programmmitgliedern offen ausgetauscht
Q15	Team	**Important information is kept away from other team members in certain situations** Wichtige Informationen werden in bestimmten Situationen von anderen Teammitgliedern ferngehalten
	Program	**Important information is kept away from other program members in certain situations** Wichtige Informationen werden in bestimmten Situationen von anderen Programmmitgliedern ferngehalten
Q16	Team	**In the team there are conflicts regarding the openness of the information flow** Im Team gibt es Konflikte um die Offenheit des Informationsaustausches
	Program	**In the program there are conflicts regarding the openness of the information flow** Im Programm gibt es Konflikte um die Offenheit des Informationsaustausches
Q17	Team	**The team members are happy with the timeliness in which they receive information from other team members** Die Teammitglieder sind zufrieden mit der Aktualität, mit der sie Informationen von anderen Teammitgliedern erhalten
	Program	**The program members are happy with the timeliness in which they receive information from other program members** Die Programmmitglieder sind zufrieden mit der Aktualität, mit der sie Informationen von anderen Programmmitgliedern erhalten

Q18	Team	**The team members are happy with the precision of the information they receive from other team members** Die Teammitglieder sind zufrieden mit der Qualität der Informationen, die sie von anderen Teammitgliedern erhalten
	Program	**The program members are happy with the precision of the information they receive from other program members** Die Programmmitglieder sind zufrieden mit der Qualität der Informationen, die sie von anderen Programmmitgliedern erhalten
Q19	Team	**The team members are happy with the usefulness of the information they receive from other team members** Die Teammitglieder sind zufrieden mit dem Mehrwert der Informationen, die sie von anderen Teammitgliedern erhalten
	Program	**The program members are happy with the usefulness of the information they receive from other program members** Die Programmmitglieder sind zufrieden mit dem Mehrwert der Informationen, die sie von anderen Programmmitgliedern erhalten

Coordination

Q20	Team	**The work done on subtasks within the team is closely harmonized** Die Arbeit an Teilaufgaben im Team ist eng aufeinander abgestimmt
	Program	**The work done on subtasks within the program is closely harmonized** Die Arbeit an Teilaufgaben im Programm ist eng aufeinander abgestimmt
Q21	Team	**There are clear and fully comprehended goals for subtasks within our team** Es gibt klare und vollständig verstandene Ziele für Teilaufgaben in unserem Team
	Program	**There are clear and fully comprehended goals for subtasks within our program** Es gibt klare und vollständig verstandene Ziele für Teilaufgaben in unserem Programm

Q22	Team	**The goals for subtasks are accepted by all team members** Die Ziele für die Teilaufgaben werden von allen Teammitgliedern akzeptiert
	Program	**The goals for subtasks are accepted by all program members** Die Ziele für die Teilaufgaben werden von allen Programmmitgliedern akzeptiert
Q23	Team	**There are conflicting interests in our team regarding subtasks/subgoals** In unserem Team gibt es Interessenkonflikte bei Teilaufgaben/Unterzielen
	Program	**There are conflicting interests in our program regarding subtasks/subgoals** In unserem Programm gibt es Interessenkonflikte bei Teilaufgaben/Unterzielen

Mutual Support

Q24	Team	**The team members help and support each other as best they can** Die Teammitglieder helfen und unterstützen sich gegenseitig so gut sie können
	Program	**The program members help and support each other as best they can** Die Programmmitglieder helfen und unterstützen sich gegenseitig so gut sie können
Q25	Team	**If team conflicts come up, they are easily and quickly resolved** Treten Konflikte im Team auf, lassen sie sich leicht und schnell lösen
	Program	**If program conflicts come up, they are easily and quickly resolved** Treten Konflikte im Programm auf, lassen sie sich leicht und schnell lösen
Q26	Team	**Discussions and controversies in the team are conducted constructively** Diskussionen und Kontroversen im Team werden konstruktiv geführt
	Program	**Discussions and controversies in the program are conducted constructively** Diskussionen und Kontroversen im Programm werden konstruktiv geführt

Q27	Team	**Suggestions and contributions of team members are respected** Vorschläge und Beiträge von Teammitgliedern werden berücksichtigt
	Program	**Suggestions and contributions of program members are respected** Vorschläge und Beiträge von Programmmitgliedern werden berücksichtigt
Q28	Team	**Suggestions and contributions of team members are discussed and further developed** Vorschläge und Beiträge von Teammitgliedern werden diskutiert und weiterentwickelt
	Program	**Suggestions and contributions of program members are discussed and further developed** Vorschläge und Beiträge von Programmmitgliedern werden diskutiert und weiterentwickelt
Q29	Team	**The team is able to reach consensus regarding important issues** Das Team ist in der Lage, einen Konsens über wichtige Themen zu erzielen
	Program	**The program is able to reach consensus regarding important issues** Die Programmmitglieder sind in der Lage, einen Konsens über wichtige Themen zu erzielen
Q30	Team	**The team cooperate well** Das Team arbeitet gut zusammen
	Program	**The program cooperates well** Die Programmmitglieder arbeiteten gut zusammen

Effort

Q31	Team	**Every team member fully pushes the teamwork** Jedes Teammitglied treibt die Teamarbeit voran
	Program	**Every program member fully pushes the teamwork in the program** Jedes Programmmitglied treibt die Programmarbeit voran
Q32	Team	**Every team member makes the teamwork their highest priority** Jedes Teammitglied macht die Teamarbeit zu seiner obersten Priorität
	Program	**Every program member makes the teamwork in the program their highest priority** Jedes Programmmitglied macht die Teamarbeit im Programm zu seiner obersten Priorität

Q33	Team	**The team put(s) much effort into the teamwork** Das Team hat viel Mühe in die Teamarbeit gesteckt
	Program	**The program put(s) much effort into the teamwork in the program** Die Programmmitglieder haben viel Mühe in die Teamarbeit im Programm gesteckt
Q34	Team	**There are conflicts regarding the effort that team members put into the teamwork** Es gibt Konflikte bezüglich der Leistung, welche die Teammitglieder in die Teamarbeit einbringen
	Program	**There are conflicts regarding the effort that program members put into the teamwork in the program** Es gibt Konflikte bezüglich der Leistung, welche die Programmmitglieder in die Teamarbeit im Programm einbringen

Cohesion

Q35	Team	**The teamwork is important to the team** Die Teamarbeit ist wichtig für das Team
	Program	**The teamwork in the program is important to the program members** Die Teamarbeit im Programm ist wichtig für das Programm
Q36	Team	**It is important to team members to be part of the team** Es ist wichtig, für die Teammitglieder, Teil des Teams zu sein
	Program	**It is important to program members to be part of the program** Es ist wichtig, für die Programmmitglieder, Teil des Programms zu sein
Q37	Team	**The team does not see anything special in this teamwork** Das Team sieht in der Teamarbeit nichts Besonderes
	Program	**The program members do not see anything special in this teamwork in the program** Die Teammitglieder sehen in der Teamarbeit im Programm nichts Besonderes
Q38	Team	**The team members are strongly attached to the team** Die Teammitglieder sind stark mit dem Team verbunden
	Program	**The program members are strongly attached to the program** Die Programmmitglieder sind stark mit dem Programm verbunden

Q39	Team	**All team members are fully integrated in the team** Alle Teammitglieder sind vollständig in das Team integriert
	Program	**All program members are fully integrated in the program** Alle Programmmitglieder sind vollständig in das Programm integriert
Q40	Team	**There were many personal conflicts in the team** Es gab viele persönliche Konflikte im Team
	Program	**There were many personal conflicts in the program** Es gab viele persönliche Konflikte im Programm
Q41	Team	**There is mutual sympathy between the members of the team** Es besteht gegenseitige Sympathie zwischen den Mitgliedern des Teams
	Program	**There is mutual sympathy between the members of the program** Es besteht gegenseitige Sympathie zwischen den Mitgliedern des Programms
Q42	Team	**The team sticks together** Das Team hält zusammen
	Program	**The program members stick together** Das Programm hält zusammen
Q43	Team	**The members of the team feel proud to be part of the team** Die Mitglieder des Teams sind stolz darauf, Teil des Teams zu sein
	Program	**The members of the program feel proud to be part of the program** Die Mitglieder des Programms sind stolz darauf, Teil des Programms zu sein
Q44	Team	**Every team member feels responsible for maintaining and protecting the team** Jedes Teammitglied fühlt sich verantwortlich für die Aufrechterhaltung und den Fortbestand des Teams
	Program	**Every program member feels responsible for maintaining and protecting the program** Jedes Programmmitglied fühlt sich verantwortlich für die Aufrechterhaltung und den Fortbestand des Programms

Balance of member - Contribution

Q45	Team	**The team recognizes the specific characteristics (strengths and weaknesses) of the individual team members** Das Team erkennt die spezifischen Eigenschaften (Stärken und Schwächen) der einzelnen Teammitglieder
	Program	**The program members recognize the specific characteristics (strengths and weaknesses) of the individual program members** Die Programmmitglieder erkennen die spezifischen Eigenschaften (Stärken und Schwächen) der einzelnen Programmmitglieder
Q46	Team	**The team members contribute to the achievement of the team's goals in accordance with their specific potential** Die Teammitglieder tragen entsprechend ihrer spezifischen Fähigkeiten zur Erreichung der Teamziele bei
	Program	**The program members contribute to the achievement of the program's goals in accordance with their specific potential** Die Programmmitglieder tragen entsprechend ihrer spezifischen Fähigkeiten zur Erreichung der Programmziele bei
Q47	Team	**Imbalance of member contributions cause conflicts in our team** Das Ungleichgewicht der Beiträge der Teammitglieder führt zu Konflikten in unserem Team
	Program	**Imbalance of member contributions cause conflicts in our program** Das Ungleichgewicht der Beiträge der Programmmitglieder führt zu Konflikten in unserem Programm

Team members' success - Work Satisfaction

Q48	Team	**So far, the team can be pleased with its work** Bisher kann das Team mit seiner Arbeit zufrieden sein
	Program	**So far, the program members can be pleased with its work** Bisher können die Programmmitglieder mit der Arbeit zufrieden sein

Q49	Team	**The team members gain from the collaborative teamwork** Die Teammitglieder profitieren von der gemeinsamen Teamarbeit
	Program	**The program members gain from the collaborative teamwork in the program** Die Programmmitglieder profitieren von der gemeinsamen Teamarbeit im Programm
Q50	Team	**The team members will like to do this type of collaborative work again** Die Teammitglieder werden diese Art der Zusammenarbeit gerne wiederholen
	Program	**The program members will like to do this type of collaborative work again** Die Programmmitglieder werden diese Art der Zusammenarbeit gerne wiederholen
Q51	Team	**We are able to acquire important know-how through this teamwork** Durch diese Teamarbeit sind wir in der Lage, wichtiges Know-how zu erwerben
	Program	**We are able to acquire important know-how through this teamwork in the program** Durch diese Teamarbeit im Programm sind wir in der Lage, wichtiges Know-how zu erwerben

Team members' success - Learning

Q52	Team	**We consider this teamwork as a technical success** Wir betrachten diese Teamarbeit als echten technischen Erfolg
	Program	**We consider this teamwork in the program as a technical success** Wir betrachten diese Teamarbeit im Programm als echten technischen Erfolg
Q53	Team	**The team learn important lessons from this teamwork** Das Team kann aus dieser Teamarbeit wichtige Erkenntnisse ziehen
	Program	**The program members learn important lessons from this teamwork in the program** Die Programmmitglieder können aus dieser Teamarbeit im Programm wichtige Erkenntnisse ziehen

Q54	Team	**Teamwork promotes one personally** Teamarbeit fördert jeden persönlich
	Program	**Teamwork in the program promotes one personally** Teamarbeit im Programm fördert jeden persönlich
Q55	Team	**Teamwork promotes one professionally** Teamarbeit fördert einen professionell
	Program	**Teamwork in the program promotes one professionally** Teamarbeit im Programm fördert einen professionell

Team Performance - Effectiveness

Q56	Team	**Going by the results, this teamwork can be regarded as successful** Ausgehend von den Ergebnissen kann das Teamwork als erfolgreich bewertet werden
	Program	**Going by the results, this teamwork in the program can be regarded as successful** Ausgehend von den Ergebnissen kann das Teamwork im Programm als erfolgreich bewertet werden
Q57	Team	**All demands of the customers are satisfied in the team** Alle Anforderungen der Kunden an das Team sind erfüllt
	Program	**All demands of the customers are satisfied in the program** Alle Anforderungen der Kunden an das Programm sind erfüllt
Q58	Team	**From the company's perspective, all team goals are achieved** Aus Unternehmenssicht werden alle Teamziele erreicht
	Program	**From the company's perspective, all program goals are achieved** Aus Unternehmenssicht werden alle Programmziele erreicht
Q59	Team	**The performance of the team advances our image to the customer** Die Leistung des Teams fördert unser Image beim Kunden
	Program	**The performance of the program advances our image to the customer** Die Leistung des Programms fördert unser Image beim Kunden

Q60	Team	**The teamwork result is of high quality** Das Ergebnis der Teamarbeit ist von hoher Qualität
	Program	**The teamwork in the program result is of high quality** Das Ergebnis der Teamwork im Programm ist von hoher Qualität
Q61	Team	**The customer is satisfied with the quality of the teamwork result** Der Kunde ist mit der Qualität der Arbeitsergebnisse im Team zufrieden
	Program	**The customer is satisfied with the quality of the teamwork results in the program** Der Kunde ist mit der Qualität der Arbeitsergebnisse im Programm zufrieden
Q62	Team	**The team is satisfied with the teamwork result** Das Team ist mit dem Ergebnis der Teamarbeit zufrieden
	Program	**The program members are satisfied with the teamwork results in the program** Die Programmitglieder sind mit dem Ergebnis des Teamworks im Programm zufrieden
Q63	Team	**The product produced in the team, requires little rework** Das im Team produzierte Produkt erfordert wenig überarbeitung
	Program	**The product produced in the program, requires little rework** Das im Programm produzierte Produkt erfordert wenig überarbeitung
Q64	Team	**The team product proves to be stable in operation** Das Produkt des Teams erweist sich als stabil im Betrieb
	Program	**The program product proves to be stable in operation** Das Produkt des Programms erweist sich als stabil im Betrieb
Q65	Team	**The team product proves to be robust in operation** Das Produkt des Teams erweist sich als robust im Betrieb
	Program	**The program product proves to be robust in operation** Das Produkt des Programms erweist sich als robust im Betrieb

Team Performance - Efficiency

Q66	Team	**The company is satisfied with how the teamwork progresses** Das Unternehmen ist mit dem Fortschritt der Teamarbeit zufrieden
	Program	**The company is satisfied with how the teamwork in the program progresses** Das Unternehmen ist mit dem Fortschritt der Teamarbeit im Programm zufrieden
Q67	Team	**Overall, the team works in a cost-efficient way** Insgesamt arbeitet das Team kosteneffizient
	Program	**Overall, the program works in a cost-efficient way** Insgesamt arbeitet das Programm kosteneffizient
Q68	Team	**Overall, the team works in a time-efficient way** Insgesamt arbeitet das Team zeiteffizient
	Program	**Overall, the program works in a time-efficient way** Insgesamt arbeitet das Programm zeiteffizient
Q69	Team	**The team is within schedule** Das Team ist im Zeitplan
	Program	**The program is within schedule** Das Programm ist im Zeitplan
Q70	Team	**The team is within budget** Das Team liegt im Rahmen des Budgets
	Program	**The program is within budget** Das Programm liegt im Rahmen des Budgets

SAFe events

Q71 **In your opinion, which SAFe events are adapted?**
Welche SAFe Events gibt es ihrer Meinung nach?

Answers	Existing Gibt es	Not Existing Gibt es nicht
Iteration Planning (Team Level)	◯	◯
Iteration Execution (Team Level)	◯	◯
Daily Stand-Up (Team Level)	◯	◯
Iteration Review (Team Level)	◯	◯
Iteration Retrospective (Team Level)	◯	◯
Backlog Refinement (Team Level)	◯	◯
Innovation and Planning Iteration (Team Level)	◯	◯
PI Planning (Program Level)	◯	◯
System Demo (Program Level)	◯	◯
Scrum of Scrums (Program Level)	◯	◯
PO Synch (Program Level)	◯	◯
Community of Practice (Program Level)	◯	◯
Inspect & Adapt (Program Level)	◯	◯

SAFe artefacts

Q72 **In your opinion, which SAFe artefacts are adapted?**
Welche SAFe Artefakte gibt es ihrer Meinung nach?

Answers	Existing Gibt es	Not Existing Gibt es nicht
Iteration Planning (Team Level)	◯	◯
Iteration Execution (Team Level)	◯	◯
Daily Stand-Up (Team Level)	◯	◯
Iteration Review (Team Level)	◯	◯
Iteration Retrospective (Team Level)	◯	◯
Backlog Refinement (Team Level)	◯	◯
Innovation and Planning Iteration (Team Level)	◯	◯
PI Planning (Program Level)	◯	◯
System Demo (Program Level)	◯	◯
Scrum of Scrums (Program Level)	◯	◯
PO Synch (Program Level)	◯	◯
Community of Practice (Program Level)	◯	◯
Inspect & Adapt (Program Level)	◯	◯

Program setup

Area	Team	Location	Number of members
GPL	GPL	Ismaning	1
		Münster	1
ART	SA	Karlsruhe	1
		Frankfurt	1
	RTE	Ismaning	1
		Münster	1
	SPO	Bretten	1
		Karlsruhe	2
		Frankfurt	1
		Münster	1
P-Acc	ACC	Frankfurt	1
		Münster	6
	FRM	Münster	8
	PST	Münster	10
P-Mel	ANA	Bretten	1
		Frankfurt	0.5
		Münster	4.5
	CRE	Karlsruhe	1
		Frankfurt	4.5
		Ismaning	1
		Leinfelden-Echterdingen	1
		Münster	20.5
	CRP	Cologne	1
		Frankfurt	2
		Münster	5
		Stuttgart	1
	SR	Bretten	1
		Ismaning	1
		Münster	4
XP-Bas	CRA	Karlsruhe	5.5
		Cologne	2
	FWK	Bretten	2
		Karlsruhe	0.5
		Ismaning	2.5
		Munich	2

	MDA	Bretten	2
		Karlsruhe	3
		Cologne	1
		Eschborn	1
		Essen	2
		Frankfurt	1
		Ismaning	4
		Munich	1
	MET	Bretten	2
		Karlsruhe	3.5
		Cluj-Napoca (Romania)	1
		Ismaning	2
	REP	Bretten	4
		Cologne	2
		Münster	2
	SYS	Karlsruhe	1.5
		Frankfurt	2
		Ismaning	6.5
	UIF	Bretten	1
		Karlsruhe	3
		Cluj-Napoca (Romania)	1
		Frankfurt	1
		Münster	1

Table 10.1: Program structure of the case study partner in March 2019 based on the teams and the locations

Area	Team	Owner	Sub-Company	Number of members
ART	SA	Owner 1	Company 1	1
		Owner 2	Company 1	1
	RTE	Owner 1	Company 1	1
		Owner 2	Company 1	1
	SPO	Owner 1	Company 1	1
			Company 3	1
		Owner 2	Company 1	3
GPL	GPL	Owner 1	Company 1	1
		Owner 2	Company 1	1
P-Acc	ACC	Owner 1	Company 1	2
		Owner 2	Company 1	5
	FRM	Owner 2	Company 1	8

	PST	Owner 1	Company 1	1
		Owner 2	Company 1	9
P-Mel	ANA	Owner 1	Company 1	2
			Company 3	0.5
		Owner 2	Company 1	3.5
	CRE	Owner 1	Company 1	8
			Company 2	1
			Company 3	1.5
		Owner 2	Company 1	15.5
			Company 2	2
	CRP	Owner 1	Company 1	3
			Company 3	1
		Owner 2	Company 1	5
	SR	Owner 1	Company 1	1
			Company 2	1
		Owner 2	Company 1	4
XP-Bas	CRA	Owner 1	Company 1	3.5
		Owner 2	Company 1	3
			Company 2	1
	FWK	Owner 1	Company 1	4.5
			Company 2	1
		Owner 2	Company 1	1.5
	MDA	Owner 1	Company 1	6
			Company 2	4
		Owner 2	Company 1	4
			Company 2	1
	MET	Owner 1	Company 1	4.5
			Company 2	1
			Company 4	1
		Owner 2	Company 1	2
	REP	Owner 1	Company 1	3
			Company 2	2
		Owner 2	Company 2	3
	SYS	Owner 1	Company 1	7.5
			Company 2	1
		Owner 2	Company 1	1.5
	UIF	Owner 1	Company 1	1
			Company 2	2
			Company 4	1
		Owner 2	Company 1	3

Table 10.2: Program structure of the case study partner in March 2019 based on the teams and the company the members are provided

Architecture goals

The List 10.1 of goals is based on an interview in January 2019 with one of the case study partners system architect. This is not a complete list, but is will provide a good overview on which they are focused.

1. Break down complexity by forcing the system to be composed of manageable parts with clear boundaries (bounded contexts)
2. Development of new components as libraries or services deployed in containers
3. Where possible, integration of existing components using a common integration model; wrapping existing components into a service (and container)
4. Co-develop domain model with business in a highly integrated fashion
5. Horizontal scalability to accommodate large amounts of data and provide real or near time analytic possibilities
6. Provide means to easily build simple standard UIs and support building complex UIs without break in UI architecture
7. Easy updates of single components
8. Built to run (decentralized logging, log analytic, monitoring baked in)
9. Built to be tested automatically, including UIs
10. Data/Metadata/Model Lineage baked in

Figure 10.1: Architecture goals from the case study partner in January 2019

Architecture principles

Table 10.3 is based on an interview in January 2019 with one of the case study partners system architect based on the architecture principles catalogue version 2 from the chair of Software Engineering for Business Information Systems (SEBIS) (Technical University of Munich).

Architecture principle	Status

Applications do not cross business function boundaries	completely implemented
Applications rely on One technology stack	partially implemented
Changes to IT systems are only made in response to business needs	completely implemented
Components have a clear owner	completely implemented
IT systems are standardized and reused throughout the organization	completely implemented
IT systems are preferably open source	completely implemented
IT systems communicate through services	completely implemented
Application development is standardized	partially implemented
Vertical system section	completely implemented
Buy application when it does not provide a significant competitive advantage in your core business	not planned
One codebase tracked in revision control, many deploys	partially implemented
Explicitly declare and isolate dependencies	partially implemented
Strictly separate build and run stages	completely implemented
Keep development, staging, and production as similar as possible	partially implemented
Maximize benefit to the enterprise, concede own preferences for the greater benefit of the entire enterprise	not planned
Common vocabulary and data definitions	partially implemented
Build application as independently as possible from the underlying technology	partially implemented
Control technical diversity	completely implemented
Loose coupling of systems or services	completely implemented
The system must be divided into modules that provide interfaces	completely implemented
Modules must be implemented as separate processes, containers, or virtual machines to maximize independence	completely implemented
The system must have two clearly separated levels of architectural decisions: Macro and Micro Architecture	not planned
Communication must use a limited set of protocols like RESTful HTTP or messaging	completely implemented
Each module must have its own independent continuous delivery pipeline	completely implemented
Operations such as configuration, deployment, log analysis, tracing, monitoring, and alerting should be standardized	partially implemented
Modules must be resilient, they must compensate unavailable modules or communication problems	partially implemented

Table 10.3: Implementation status of architecture principles from the case study partner in January 2019

Original TWQ survey questions

Questions that are used by Hoegl and Gemuenden (2001) for the TWQ model [59].

Communication (10 questions): (1) There was frequent communication within the team. (2) The team members communicated often in spontaneous meetings, phone conversations, etc. (3) The team members communicated mostly directly and personally with each other. (4) There were mediators through whom communication was conducted (R). (5) Project-relevant information was shared openly by all team members. (6) Important information was kept away from other team members in certain situations (R). (7) In our team, there were conflicts regarding the openness of the information flow (R). (8) The team members were happy with the timeliness in which they received information from other team members. (9) The team members were happy with the precision of the information received from other team members. (10) The team members were happy with the usefulness of the information received from other team members.

Coordination (4 questions): (11) The work done on subtasks within the project was closely harmonized. (12) There were clear and fully comprehended goals for subtasks within our team. (13) The goals for subtasks were accepted by all team members. (14) There were conflicting interests in our team regarding subtasks/subgoals (R).

Balance of Member Contributions (3 questions): (15) The team recognized the specific potentials (strengths and weaknesses) of individual team members. (16) The team members were contributing to the achievement of the team's goals in accordance with their specific potential. (17) Imbalance of member contributions caused conflicts in our team (R).

Mutual Support (6 questions): (18) The team members helped and supported each other as best as they could. (19) If conflicts came up, they were easily and quickly resolved. (20) Discussions and controversies were conducted constructively. (21) Suggestions and contributions of team members were respected. (22) Suggestions and contributions of team members were discussed and further developed. (23) Our team was able to reach consensus regarding important issues.

Effort (4 questions): (24) Every team member fully pushed the project. (25) Every team member made the project their highest priority. (26) Our team put much effort into the project. (27) There were conflicts regarding the effort that team members put into the project (R).

Cohesion (10 questions): (28) It was important to the members of our team to be part of this project. (29) The team did not see anything special in this project (R). (30) The team members were strongly attached to this project. (31) The project was important to our team. (32) All members were fully integrated in our team. (33)

There were many personal conflicts in our team (R). (34) There was personal attraction between the members of our team. (35) Our team was sticking together. (36) The members of our team felt proud to be part of the team. (37) Every team member felt responsible for maintaining and protecting the team.

Effectiveness (10 questions): (38) Going by the results, this project can be regarded as successful. (39) All demands of the customers have been satisfied. (40) From the company's perspective, all project goals were achieved. (41) The performance of our team advanced our image to the customer. (42) The project result was of high quality. (43) The customer was satisfied with the quality of the project result. (44) The team was satisfied with the project result. (45) The product required little rework. (46) The product proved to be stable in operation. (47) The product proved to be robust in operation.

Efficiency (5 questions): (48) From the company's perspective one could be satisfied with how the project progressed. (49) Overall, the project was done in a cost-efficient way. (50) Overall, the project was done in a time-efficient way. (51) The project was within schedule. (52) The project was within budget.

Work Satisfaction (3 questions): (53) After this project, the team members could draw a positive balance for themselves overall. (54) The team members have gained from the collaborative project. (55) The team members would like to do this type of collaborative work again.

Learning (5 questions): (56) We were able to acquire important know-how through this project. (57) We see this project as a technical success. (58) Our team learned important lessons from this project. (59) Teamwork promotes one personally. (60) Teamwork promotes one professionally.

(R) = reverse coded item

Original TWQ survey results by Hoegl and Gemuenden

TWQ	Items	Mean	S.D.	Alpha
Communication	10	4.20	0.44	0.94
Coordination	4	4.04	0.59	0.85
Balance of Member Cont.	3	4.08	0.50	0.72
Mutual Support	7	4.13	0.55	0.93
Effort	4	3.91	0.59	0.94
Cohision	10	3.89	0.56	0.97
Effectiveness-TM	10	3.91	0.59	0.91
Efficiency-TM	5	3.76	0.77	0.86
Work Satisfaction	3	4.01	0.55	0.79
Learning	5	4.06	0.46	0.76

Table 10.4: TWQ survey results - Number of items, means, standard deviations (S.D.) and reliabilities [59]

Team dependency graph

Compared to Figure 4.9 the node size is related to the outdegree edges in Figure 10.2.

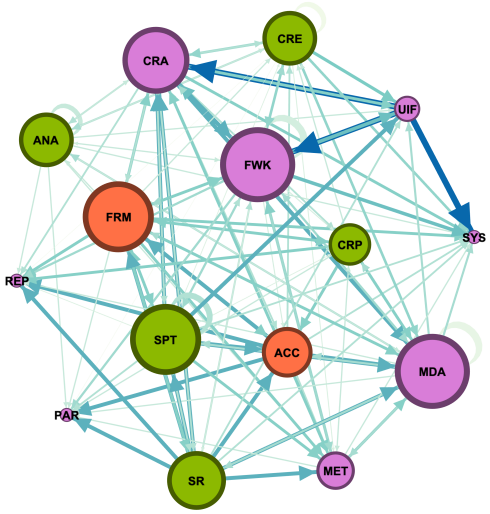


Figure 10.2: Team dependencies without the GPL and ART teams - program overview (outdegree)

Bibliography

[1] S. Ambler and M. Lines. "Scaling agile software development tactically: Disciplined agile delivery at scale." In: *Disciplined Agile Consortium, Tech. Rep.* (2016).

[2] M. Aoyama. "Web-based agile software development." In: *IEEE software* 15.6 (1998), pp. 56–65.

[3] K. Backhaus, B. Erichson, W. Plinke, R. Weiber, et al. *Multivariate analysemethoden: eine anwendungsorientierte einführung*. Vol. 11. Springer, 2006.

[4] L. R. Baldrich, M. Reichert, M. Zimoch, and J. Scheible. "jQAssistant: A QA Tool for Definition and Validation of Software Architecture Rules." PhD thesis. Ulm University, 2018.

[5] T. S. Bateman, H. O'Neill, and A. Kenworthy-U'Ren. "A hierarchical taxonomy of top managers' goals." In: *Journal of Applied Psychology* 87.6 (2002), p. 1134.

[6] K. Beck, M. Beedle, A. Van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, et al. "Manifesto for agile software development." In: (2001).

[7] K. Beck and E. Gamma. *Extreme programming explained: embrace change*. addison-wesley professional, 2000.

[8] S. Bibb and J. Kourdi. *A Question of Trust: The Crucial Nature of Trust in Business, Work and Life - And How to Build It*. Marshall Cavendish, 2007. ISBN: 9781904879213.

[9] P. M. Blau and W. R. Scott. "1962Formal Organizations." In: *San Francisco: Chandler* (1962).

[10] B. Boehm. "Get ready for agile methods, with care." In: *Computer* 35.1 (2002), pp. 64–69.

[11] B. Boehm and R. Turner. "Management challenges to implementing agile processes in traditional development organizations." In: *IEEE software* 22.5 (2005), pp. 30–39.

[12] E. Bradner, G. Mark, and T. D. Hertel. "Team size and technology fit: Participation, awareness, and rapport in distributed teams." In: *IEEE Transactions on Professional Communication* 48.1 (2005), pp. 68–77.

[13] M. W. Browne, R. Cudeck, et al. "Alternative ways of assessing model fit." In: *Sage focus editions* 154 (1993), pp. 136–136.

[14] C. V. Bullen and J. F. Rockart. "A primer on critical success factors." In: (1981).

[15] T. L. C. B.V. *LeSS Framework Overview*. `https://less.works/less/framework/index.html`. 2019 (accessed March 23, 2019).

[16] B. M. Byrne and S. M. Stewart. "Teacher's corner: The MACS approach to testing for multigroup invariance of a second-order structure: A walk through the process." In: *Structural Equation Modeling* 13.2 (2006), pp. 287–321.

[17] E. Carmel. "Global software teams: collaborating across borders and time zones." In: (1999).

[18] E. Carmel, J. A. Espinosa, et al. "Timeshifting—The Mother of All Solutions for Time Zone Differences." In: *World Scientific Book Chapters* (2018), pp. 99–109.

[19] M. Ceschi, A. Sillitti, G. Succi, and S. De Panfilis. "Project management in plan-based and agile companies." In: *IEEE software* 22.3 (2005), pp. 21–27.

[20] T. Chow and D.-B. Cao. "A survey study of critical success factors in agile software projects." In: *Journal of systems and software* 81.6 (2008), pp. 961–971.

[21] P. Coad, J. d. Luca, and E. Lefebvre. *Java modeling color with UML: Enterprise components and process with Cdrom*. Prentice Hall PTR, 1999.

[22] L. Cocco, K. Mannaro, G. Concas, and M. Marchesi. "Simulating kanban and scrum vs. waterfall with system dynamics." In: *International Conference on Agile Software Development*. Springer. 2011, pp. 117–131.

[23] A. Cockburn. *Crystal clear: a human-powered methodology for small teams*. Pearson Education, 2004.

[24] A. Cockburn and J. Highsmith. "Agile software development, the people factor." In: *Computer* 34.11 (2001), pp. 131–133.

[25] D. Cogburn, M. Hine, J. A. Espinosa, and A. Santuzzi. "Virtual Teams, Organizations, and Networks." In: *Proceedings of the 52nd Hawaii International Conference on System Sciences*. 2019.

[26] D. Cohen, M. Lindvall, and P. Costa. "An introduction to agile methods." In: *Advances in computers* 62.03 (2004), pp. 1–66.

[27] J. W. Creswell and V. L. P. Clark. *Designing and conducting mixed methods research*. Sage publications, 2017.

[28] J. N. Cummings, J. A. Espinosa, and C. K. Pickering. "Crossing spatial and temporal boundaries in globally distributed projects: A relational model of coordination delay." In: *Information Systems Research* 20.3 (2009), pp. 420–439.

[29] D. Cyr, M. Head, H. Larios, and B. Pan. "Exploring human images in website design: a multi-method approach." In: *MIS quarterly* (2009), pp. 539–566.

[30] T. A. De Vries, J. R. Hollenbeck, R. B. Davison, F. Walter, and G. S. Van Der Vegt. "Managing coordination in multiteam systems: Integrating micro and macro perspectives." In: *Academy of Management Journal* 59.5 (2016), pp. 1823–1844.

[31] L. A. DeChurch, C. S. Burke, M. L. Shuffler, R. Lyons, D. Doty, and E. Salas. "A historiometric analysis of leadership in mission critical multiteam environments." In: *The Leadership Quarterly* 22.1 (2011), pp. 152–169.

[32] J. Desjardins. *TECHNOLOGYHow Many Millions of Lines of Code Does It Take?* `https://www.visualcapitalist.com/millions-lines-of-code/`. 2017 (accessed March 16, 2019).

[33] D. J. Devine, L. D. Clayton, J. L. Philips, B. B. Dunford, and S. B. Melner. "Teams in organizations: Prevalence, characteristics, and effectiveness." In: *Small group research* 30.6 (1999), pp. 678–711.

[34] C. A. DiazGranados, A. J. Dunning, M. Kimmel, D. Kirby, J. Treanor, A. Collins, R. Pollak, J. Christoff, J. Earl, V. Landolfi, et al. "Efficacy of high-dose versus standard-dose influenza vaccine in older adults." In: *New England Journal of Medicine* 371.7 (2014), pp. 635–645.

[35] K. Dikert, M. Paasivaara, and C. Lassenius. "Challenges and success factors for large-scale agile transformations: A systematic literature review." In: *Journal of Systems and Software* 119 (2016), pp. 87–108.

[36] T. Dingsøyr and N. B. Moe. "Towards principles of large-scale agile development." In: *International Conference on Agile Software Development*. Springer. 2014, pp. 1–8.

[37] T. Dingsøyr, N. B. Moe, T. E. Fægri, and E. A. Seim. "Exploring software development at the very large-scale: a revelatory case study and research agenda for agile method adaptation." In: *Empirical Software Engineering* 23.1 (2018), pp. 490–520.

[38] T. Dingsøyr, K. Rolland, N. B. Moe, and E. A. Seim. "Coordination in multi-team programmes: An investigation of the group mode in large-scale agile software development." In: *Procedia Computer Science* 121 (2017), pp. 123–128.

[39] P. F. Drucker. *Management: Tasks, responsibilities, practices*. truman talley Books, 1986.

[40] K. M. Eisenhardt. "Building theories from case study research." In: *Academy of management review* 14.4 (1989), pp. 532–550.

[41] J. A. Espinosa, N. Nan, and E. Carmel. "Do gradations of time zone separation make a difference in performance? A first laboratory study." In: *International Conference on Global Software Engineering (ICGSE 2007)*. IEEE. 2007, pp. 12–22.

[42] J. A. Espinosa, N. Nan, and E. Carmel. "Temporal distance, communication patterns, and task performance in teams." In: *Journal of Management Information Systems* 32.1 (2015), pp. 151–191.

[43] J. A. Espinosa, S. A. Slaughter, R. E. Kraut, and J. D. Herbsleb. "Familiarity, complexity, and team performance in geographically distributed software development." In: *Organization science* 18.4 (2007), pp. 613–630.

[44] J. A. Espinosa, S. A. Slaughter, R. E. Kraut, and J. D. Herbsleb. "Team knowledge and coordination in geographically distributed software development." In: *Journal of management information systems* 24.1 (2007), pp. 135–169.

[45] B. M. Firth, J. R. Hollenbeck, J. E. Miles, D. R. Ilgen, and C. M. Barnes. "Same page, different books: Extending representational gaps theory to enhance performance in multiteam systems." In: *Academy of Management Journal* 58.3 (2015), pp. 813–835.

[46] A. Floh and H. Treiblmaier. "What keeps the e-banking customer loyal? A multigroup analysis of the moderating role of consumer characteristics on e-loyalty in the financial service industry." In: *A Multigroup Analysis of the Moderating Role of Consumer Characteristics on E-Loyalty in the Financial Service Industry.(March 26, 2006)* (2006).

[47] S. Fraser, A. Martin, R. Biddle, D. Hussman, G. Miller, M. Poppendieck, L. Rising, and M. Striebeck. "The role of the customer in software development: the XP customer-fad or fashion?" In: *Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications*. ACM. 2004, pp. 148–150.

[48] J. R. Galbraith. "Matrix organization designs How to combine functional and project forms." In: *Business horizons* 14.1 (1971), pp. 29–40.

[49] H. H. Gerth. *Essays in sociology*. 1952.

[50] Google. *Google Scholar Cites - Teamwork quality and the success of innovative projects: A theoretical concept and empirical evidence*. `https://scholar.google.fi/scholar?cites=10586756228164766748&as_sdt=2005&sciodt=0,5&hl=en`. 2019 (accessed February 10, 2019).

[51] L. Gratton and T. J. Erickson. "Eight ways to build collaborative teams." In: *Harvard business review* 85.11 (2007), p. 100.

[52] J. W. Guthrie Jr, H. A. Priest Jr, and E. Salas Jr. "The continued evolution of team research: A theoretical model of performance in multiteam systems." In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*. Vol. 49. 3. SAGE Publications Sage CA: Los Angeles, CA. 2005, pp. 582–585.

[53] J. R. Hackman. *Groups that work and those that don't*. E10 H123. Jossey-Bass, 1990.

[54] J. R. Hackman and R. J. Hackman. *Leading teams: Setting the stage for great performances*. Harvard Business Press, 2002.

[55] J. Hackman. *The design of work teams. Inj. w. lorsch (ed.), Handbook of organizational behavior (pp. 315-342)*. 1987.

[56] J. D. Herbsleb and A. Mockus. "An empirical study of speed and communication in globally distributed software development." In: *IEEE Transactions on software engineering* 29.6 (2003), pp. 481–494.

[57] M. A. Hitt, B. W. Keats, and S. M. DeMarie. "Navigating in the new competitive landscape: Building strategic flexibility and competitive advantage in the 21st century." In: *Academy of Management Perspectives* 12.4 (1998), pp. 22–42.

[58] M. Hoegl. "Smaller teams–better teamwork: How to keep project teams small." In: *Business Horizons* 48.3 (2005), pp. 209–214.

[59] M. Hoegl and H. G. Gemuenden. "Teamwork quality and the success of innovative projects: A theoretical concept and empirical evidence." In: *Organization science* 12.4 (2001), pp. 435–449.

[60] L.-t. Hu and P. M. Bentler. "Cutoff criteria for fit indexes in covariance structure analysis: Conventional criteria versus new alternatives." In: *Structural equation modeling: a multidisciplinary journal* 6.1 (1999), pp. 1–55.

[61] D. R. Ilgen, J. R. Hollenbeck, M. Johnson, and D. Jundt. "Teams in organizations: From input-process-output models to IMOI models." In: *Annu. Rev. Psychol.* 56 (2005), pp. 517–543.

[62] V. Inc. *Version One: The 12th Annual State of Agile Report (2017)*. `https://explore.versionone.com/state-of-agile/versionone-12th-annual-state-of-agile-report`. 2017 (accessed January 31, 2019).

[63] K. G. Jöreskog. "Testing structural equation models." In: *Sage focus editions* 154 (1993), pp. 294–294.

[64] H. F. Kaiser. "A note on the equamax criterion." In: *Multivariate behavioral research* 9.4 (1974), pp. 501–503.

[65] B. Kaplan and D. Duchon. "Combining qualitative and quantitative methods in information systems research: a case study." In: *MIS quarterly* (1988), pp. 571–586.

[66] M. B. Ken Schwaber. *Agile Software Development with Scrum* -. London: Prentice Hall, 2001. ISBN: 978-0-130-67634-4.

[67] P. Khosroshahi, M. Hauder, A. Schneider, and F. Matthes. "Enterprise architecture management pattern catalog version 2. 0." In: *Tech. Rep.* (2015).

[68] G. Kim, J. Humble, P. Debois, and J. Willis. *The DevOps Handbook:: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*. IT Revolution, 2016.

[69] R. Knaster and D. Leffingwell. *SAFe 4.5 Distilled: Applying the Scaled Agile Framework for Lean Software and Systems Engineering*. Addison-Wesley Professional, 2018.

[70] D. Leffingwell. *Agile software requirements: lean requirements practices for teams, programs, and the enterprise*. Addison-Wesley Professional, 2010.

[71] D. Leffingwell. *Scaling software agility: best practices for large enterprises*. Pearson Education, 2007.

[72] B. Linders et al. *Getting Value out of Agile Retrospectives-A Toolbox of Retrospective Exercises*. Lulu. com, 2014.

[73] Y. Lindsjørn, G. R. Bergersen, T. Dingsøyr, and D. I. Sjøberg. "Teamwork Quality and Team Performance: Exploring Differences Between Small and Large Agile Projects." In: *International Conference on Agile Software Development*. Springer. 2018, pp. 267–274.

[74] Y. Lindsjørn, D. I. Sjøberg, T. Dingsøyr, G. R. Bergersen, and T. Dybå. "Teamwork quality and project success in software development: A survey of agile development teams." In: *Journal of Systems and Software* 122 (2016), pp. 274–286.

[75] E. A. Locke and G. P. Latham. *A theory of goal setting & task performance.* Prentice-Hall, Inc, 1990.

[76] G. Lohan, M. Lang, and K. Conboy. "Having a customer focus in agile software development." In: *Information Systems Development*. Springer, 2011, pp. 441–453.

[77] P. B. Lowry and J. Gaskin. "Partial least squares (PLS) structural equation modeling (SEM) for building and testing behavioral causal theory: When to choose it and how to use it." In: *IEEE transactions on professional communication* 57.2 (2014), pp. 123–146.

[78] M. A. Marks, L. A. DeChurch, J. E. Mathieu, F. J. Panzer, and A. Alonso. "Teamwork in multiteam systems." In: *Journal of Applied Psychology* 90.5 (2005), p. 964.

[79] J. Mathieu, M. A. Marks, and S. J. Zaccaro. "Multi-team systems." In: *International handbook of work and organizational psychology* 2.2 (2001).

[80] S. Maurya. *Categorise Your Agile Teams To Manage Dependencies and Avoid Overlapping Issues & Conflicts*. `https://we-are.bookmyshow.com/categorise-your-agile-teams-to-manage-dependencies-and-avoid-overlapping-issues-conflicts-a566be68180e`. 2018 (accessed March 23, 2019).

[81] S. Misra and M. Omorodion. "Survey on agile metrics and their inter-relationship with other traditional development metrics." In: *ACM SIGSOFT Software Engineering Notes* 36.6 (2011), pp. 1–3.

[82] B. Mullen and C. Copper. "The relation between group cohesiveness and performance: An integration." In: *Psychological bulletin* 115.2 (1994), p. 210.

[83] NASA. *Original Apollo 11 Guidance Computer (AGC) source code for the command and lunar modules*. `https://github.com/chrislgarry/Apollo-11/`. 2019 (accessed March 16, 2019).

[84] K. J. Preacher, D. D. Rucker, and A. F. Hayes. "Addressing moderated mediation hypotheses: Theory, methods, and prescriptions." In: *Multivariate behavioral research* 42.1 (2007), pp. 185–227.

[85] N. Ramasubbu and R. K. Balan. "Globally distributed software development project performance: an empirical analysis." In: *Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*. ACM. 2007, pp. 125–134.

[86] M. Ringelmann. "Recherches sur les moteurs animes: Travail de l'homme [Research on 424 animate sources of power: The work of man]." In: *Ann l'Inst Natl Agron 2nd Ser* 425 (1913), p. 12.

[87] C. M. Ringle, S. Wende, and J.-M. Becker. *SmartPLS 3*. `http://www.smartpls.com`. 2015.

[88] C. Robson. *Real world research: A resource for social scientists and practitioner-researchers*. Wiley-Blackwell, 2002.

[89] W. W. Royce. "Managing the development of large software systems: concepts and techniques." In: *Proceedings of the 9th international conference on Software Engineering*. IEEE Computer Society Press. 1987, pp. 328–338.

[90] P. Runeson and M. Höst. "Guidelines for conducting and reporting case study research in software engineering." In: *Empirical software engineering* 14.2 (2009), p. 131.

[91] I. Scaled Agile. *About Scaled Agile Framework and History*. https://www.scaledagileframework.com/about/. 2019 (accessed January 31, 2019).

[92] I. Scaled Agile. *Scaled Agile Framework - Case Studies*. https://www.scaledagileframework.com/case-studies/. 2019 (accessed January 31, 2019).

[93] I. Scaled Agile. *Scaled Agile Framework - Essential SAFe Poster*. https://www.scaledagileframework.com/posters/. 2019 (accessed January 31, 2019).

[94] A. Scheerer et al. *Coordination in Large-Scale Agile Software Development*. Springer, 2017.

[95] K. Schwaber. *Agile project management with Scrum*. Microsoft press, 2004.

[96] K. Schwaber and J. Sutherland. "The scrum guide." In: *Scrum Alliance* 21 (2011).

[97] scrumguides.org. *Definition of Scrum*. https://www.scrumguides.org/scrum-guide.html. 2019 (accessed January 30, 2019).

[98] Scrum.org. *Nexus Framework Poster*. https://www.scrum.org/resources/nexus-framework-poster. 2019 (accessed March 23, 2019).

[99] scrum.org. *The Scrum Framework*. https://www.scrum.org/resources/scrum-framework-poster. 2019 (accessed January 30, 2019).

[100] S. SE. *About SAP SE*. https://www.sap.com/corporate/en.html. 2019 (accessed February 17, 2019).

[101] M. L. Shuffler, M. Jiménez-Rodrıguez, and W. S. Kramer. "The science of multiteam systems: A review and future research agenda." In: *Small Group Research* 46.6 (2015), pp. 659–699.

[102] M. L. Shuffler, W. S. Kramer, D. R. Carter, A. L. Thayer, and M. A. Rosen. "Leveraging a team-centric approach to diagnosing multiteam system functioning: The role of intrateam state profiles." In: *Human Resource Management Review* 28.4 (2018), pp. 361–377.

[103] M. G. Software. *Daily Scrum Meeting*. https://www.mountaingoatsoftware.com/agile/scrum/meetings/daily-scrum. 2019 (accessed January 18, 2019).

[104] M. G. Software. *SAFe Metrics*. https://www.scaledagileframework.com/metrics. 2019 (accessed February 10, 2019).

[105] H. N. (statista.com). *German: Mangel an IT-Experten wird größer*. `https://de.statista.com/infografik/16584/zu-besetzende-it-stellen-in-der-deutschen-gesamtwirtschaft/`. 2019 (accessed March 16, 2019).

[106] I. D. Steiner. "Group process and productivity (social psychological monograph)." In: (2007).

[107] A. L. Stinchcombe. "Bureaucratic and craft administration of production: A comparative study." In: *Administrative science quarterly* (1959), pp. 168–187.

[108] H. Takeuchi and I. Nonaka. "The new new product development game." In: *Harvard business review* 64.1 (1986), pp. 137–146.

[109] A. Tashakkori and C. Teddlie. *Sage handbook of mixed methods in social & behavioral research*. Sage, 2010.

[110] O. Turel and C. Bart. "Board-level IT governance and organizational performance." In: *European Journal of Information Systems* 23.2 (2014), pp. 223–239.

[111] Ö. Uludağ, M. Kleehaus, X. Xu, and F. Matthes. "Investigating the role of architects in scaling agile frameworks." In: *2017 IEEE 21st International Enterprise Distributed Object Computing Conference (EDOC)*. IEEE. 2017, pp. 123–132.

[112] L. Vijayasarathy and D. Turk. "Drivers of agile software development use: Dialectic interplay between benefits and hindrances." In: *Information and Software Technology* 54.2 (2012), pp. 137–148.

[113] J. Vom Brocke, A. Simons, B. Niehaves, K. Riemer, R. Plattfaut, A. Cleven, et al. "Reconstructing the giant: on the importance of rigour in documenting the literature search process." In: *Ecis*. Vol. 9. 2009, pp. 2206–2217.

[114] E. Weimar, A. Nugroho, J. Visser, A. Plaat, M. Goudbeek, and A. P. Schouten. "The Influence of Teamwork Quality on Software Team Performance." In: *arXiv preprint arXiv:1701.06146* (2017).

[115] K. Weinkauf, M. Högl, and H. G. Gemünden. "Zusammenarbeit in innovativen Multi-Team-Projekten: Eine theoretische und empirische Analyse." In: *Schmalenbachs Zeitschrift für betriebswirtschaftliche Forschung* 56.5 (2004), pp. 419–435.